The ABC of Software Engineering Research

Faiz Ahmed

Christian-Albrecht University of Kiel, Germany
stu225473@mail.uni-kiel.de

Abstract. This paper presents the ABC framework, which is adapted from the taxonomy developed by McGrath and his colleagues in the social sciences, and seeks to provide guidance to help researchers select an appropriate research strategy that aligns with the goals of their research. The ABC framework contributes to the discourse on research methodology in software engineering by offering an alternative, holistic view that positions eight archetypal research strategies. First of all, I will explain what is ABC framework all about and then will discuss the metaphors for each strategy and their inherent limitations and potential strengths. Finally, some limitations of this framework.

Keywords: GSE · Metaphors · Requirements Engineering Research · SE.

1 Introduction

Physics, biology, and medicine have well-refined public explanations of their research processes. Even in simplified form, these provide guidance about what counts as ”good research” both inside and outside the field [23]. Software engineering does not have this sort of well-understood guidance. Generally speaking, software engineering researchers seek better ways to develop and evaluate software. They are motivated by practical problems, and key objectives of the research are often quality, cost, and timeliness of software products. But they rarely write explicitly about their paradigms of research and their standards for judging quality of results. A number of attempts to characterize software engineering research have contributed elements of the answer, but they do not yet paint a comprehensive picture.

However, more recently, software engineering researchers have adopted numerous research methods, approaches, and techniques from other fields, partly driven by calls for more evidence-based practice and empirical research. The ABC framework is one of the research result of software engineering researchers, which offers eight archetypal research strategies of SE research.

2 Research Methods in Software Engineering

Any research proposal in computer science needs to be validated properly to check it claims, improvements and also its applicability in practice. To conduct such a validation, either the proposal has to be proven formally or empirical methods have to be used to gather evidence. The fact that formal proofs are only seldom possible in software engineering (SE), has lead researchers to emphasise the role of empirical methods, which are traditionally more common in disciplines like physics, social sciences, medicine, or psychology.

Several different empirical methods are known. Quantitative methods try to measure a certain effect, while qualitative methods search for the reasons of an observed effect. Examples for quantitative methods are experiments, case studies, field studies, surveys and meta-studies. Examples for qualitative methods are interviews and group discussions [27]. The ABC framework uses both quantitative and qualitative methods to explain different research strategies and their applications.

3 Dimensions of Research Strategies

Two important dimensions of a research strategy are the level of obtrusiveness and generalizability. These two dimensions are the axes in the ABC framework.

3.1 Obtrusiveness

The first dimension is concerned with how *obtrusive* the research is: to what extent does a researcher "intrude" on the research setting, or simply make observations in an unobtrusive way. Researchers who wish to exert more control in a research study usually have to intrude in the research setting, for example, to manipulate some variables. Several authors have argued that the level of control that a researcher can exert while conducting a study is a key concern. One concern often raised in the context of qualitative methods such as ethnographic and case study research is a lack of control.

3.2 Generalizability

A second key concern that authors have expressed is the level of generalizability of research findings [25]. This has been a recurring concern in software engineering research, in particular in the context of case studies. Exploratory case studies, and other types of field studies, are limited in that the researcher cannot draw any statistically generalizing conclusion from such research. However, such generalization of findings is not the goal of such studies- instead, exploratory case studies and other types of case studies aim to develop understanding rather than generalization of findings across different settings.

4 THE ABC FRAMEWORK FOR RESEARCH STRATEGIES

The term ABC refers to the research goal that strives for generalizability over Actors(A) and precise measurement of their Behavior (B), in a realistic Context (C). Following Table 1 shows some examples of Actors, Behaviors and Context in SE research.

Table 1. Examples of Actors, Behavior, and Context in Software Engineering Research

Actors	Managers, software engineers, users, software systems, software development artifacts incl. defects, tools, techniques, prototypes
Behavior	System behavior (e.g., reliability, performance, and other quality attributes), software engineers' behavior and antecedents such as productivity, motivation, and intention
Context	Industrial settings, organizations, software projects, development teams, software laboratory, classroom, meeting rooms

The remaining section presents a framework that provides a holistic overview of eight different research strategies used in software engineering research. To operationalize the framework, some exemplar studies are identified by the authors [25] that adopted the archetypal research strategies in two SE research areas (global software engineering and requirements engineering).

The two key dimensions *obtrusiveness* and *generazability* of ABC framework already discussed in the previous section (3). The first dimension (the x-axis in Figure 1) is that of generalizability or universality: research strategies can result in findings that are either *specific* to a particular context of system or more *generalizable*. The second dimension (the y-axis in Figure 1) is the level of "obstrusiveness" and refers to the degree to which a research settings is manipulated or instrumented to conduct research. The remainder of this section presents each of the eight research strategies of ABC framework (also depicted in the Figure 1).

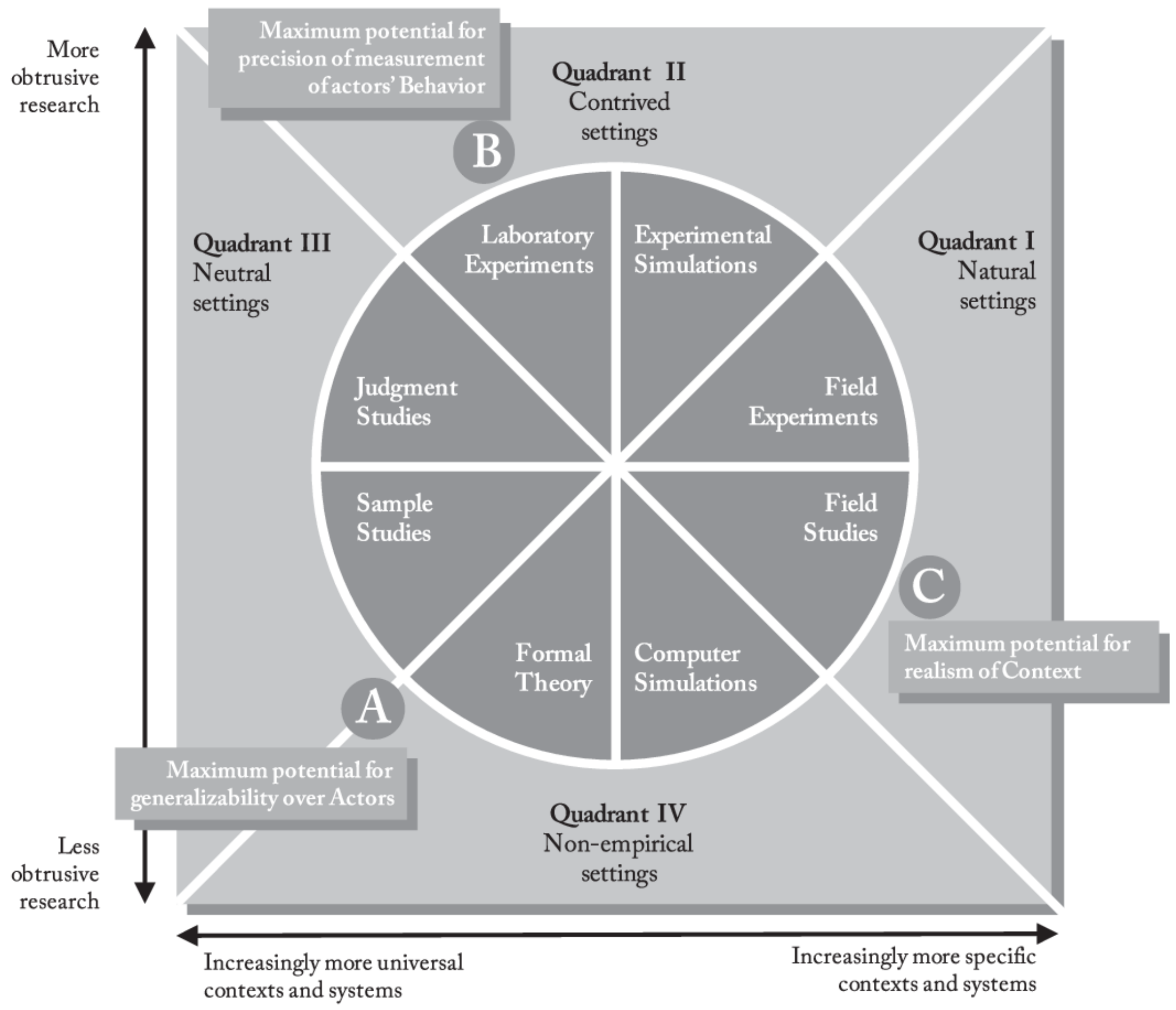

Fig. 1. The ABC framework [25]: eight research strategies as categories of research methods for software engineering (adapted from Runkel and McGrath [6]). Quadrants I to IV represent different research settings.

4.1 Field Studies

Field study refers to any research conducted in a specific, real-world setting to study a specific software engineering phenomenon. The field study strategy is located in Quadrant I (Figure 1), representing natural settings. Field studies are unobtrusive in that a researcher does not actively control or change any parameters or variables. That is, there is no deliberate modification of the research setting. Field studies are used to develop a deep understanding of phenomena in specific, concrete, and realistic settings—the specific setting may refer (among others) to a particular system, organization, or team of individuals. For this reason, a field study offers maximum potential for capturing a realistic context (indicated by the “C” marker in Figure 1).

The metaphor *jungle* is used for *filed studies*. A jungle is usually an undisturbed setting, so wildlife can be observed in its natural habitat—but the researcher is expected to not disturb the natural setting as this would lead to different behavior, which in turn reduces the realism of the context.

4.2 Field Experiments

A *field experiment* refers to an experimental study conducted in a natural setting with a high degree of realism (similar to a field study), but in this strategy the researcher manipulates some properties or variables in the research setting so as to observe an effect of some kind—this manipulation educes the level of realism compared to a field study. The realistic research setting exists independent of the researcher, which distinguishes it from the contrived research setting in a laboratory experiment. Common settings for field experiments in SE include a specific software development organization or team or a deployed production system.

Nature reserve as a metaphor is used for a field experiment setting. In a nature reserve, flora and fauna can still thrive as normal, but the reserve facilitates the conduct of research, for example, by placing fences so as to separate the wildlife into different treatment groups and evaluate the effects of those treatments. The study’s goal was to investigate the reproducibility of software projects.

4.3 Experimental Simulations

The *experimental simulation* strategy is one of two strategies in Quadrant II—the quadrant that represents contrived research settings. In an experimental simulation, the behaviors of actors (e.g., developers, users, or software systems) that a researcher aims to observe and measure are natural, but the setting within which these occur has been specifically created for the purposes of the study—that is, without the study, the setting would not exist. Thus, the term “simulation” in this context refers to the artificially created setting that aims to recreate a concrete type of setting in which the “experiment” or observed behavior takes place. The level of obtrusiveness is higher than that of a field experiment, because the simulation requires a *contrived* setting.

Runkel and McGrath’s metaphor compared an experimental simulation to a *greenhouse* [22]. A greenhouse is built to simulate a certain setting, optimized for certain characteristics, for example, to grow fruits that require much sunlight and high temperatures, which otherwise could not be grown in cold climates. Compared to a nature reserve (i.e., field experiments), a greenhouse is a far more contrived setting and less realistic, but it gives a researcher potentially more control over what happens inside. An alternative metaphor is a *flight simulator* [22]. While the specific series of events cannot be fully controlled (it depends on user input), the behavior of participants (e.g., pilots in training) can be carefully monitored and analyzed.

4.4 Laboratory Experiments

The *laboratory experiment* is the second strategy in Quadrant II. It differs from field experiments, which are positioned in Quadrant I and thus study phenomena in their natural context, whereas laboratory experiments are set in a contrived setting. A laboratory experiment is characterized by a high potential to neutralize any confounding factors and extraneous conditions [22]. Consequently, laboratory experiments allow a researcher to exercise maximum precision of measurement of behavior on the studied object—this is indicated by the Quadrant II in Figure 1.

A useful metaphor for laboratory experiment is a *test tube*, which Runkel and McGrath [22] used to set it clearly apart from the greenhouse (experimental simulation, representing a more continuous flow of events) and nature reserve (field experiment).

4.5 Judgment Studies

A *judgment study* involves gathering empirical data from a group of participants who are asked to judge or rate behaviors, to respond to a request or "stimulus" offered by a researcher, or to discuss a given topic of interest. Judgment studies rely on systematic sampling rather than representative sampling and should involve experts, appropriately informed to respond to a certain question or stimulus. The goal of a judgment study is to seek generalizability over the responses, rather than generalizability to a population of actors.

The metaphor for judgment study is a *courtroom*, in which a panel of participants (the jury) are carefully and systematically selected. In a courtroom, evidence is presented (a *stimulus*) and eventually the jury returns a verdict. The setting itself (i.e., the courtroom) is only manipulated to the extent that it aims to be neutral and not distract the participants from the matter at hand (i.e., the case).

4.6 Sample Studies

The *sample study* strategy aims to achieve generalizability over a certain population of *actors*, whether these are software professionals, software systems, or artifacts of the development process. The sample study is one of two strategies with the potential to maximize generalizability to a population—this is indicated by the marker "A" in Figure 1. The sample study is one of the most common strategies in SE research. When the sample consists of human respondents, data is usually collected through questionnaires that can be administered in hard copy or through websites. In SE research, the sample study is also used quite frequently to study large sets of software development artifacts or projects, in particular open-source software projects, which are easy to access.

A metaphor for the sample study strategy is a *referendum* (though they admit this only suggests samples of human participants, not development artifacts). In a referendum, usually a limited set of questions is presented to a large group of people, who are invited to respond—typically, only a sample actually responds.

4.7 Formal Theory

Formal theory is one of two strategies in Quadrant IV that have no *empirical setting*. Formal theory[1] is a strategy that aims at a high level of universality so that the resulting theory or framework can be applied under a wide array of circumstances, although most theories have boundaries outside of which they do not apply [24]. Formal theory and its role in software engineering research have received considerable attention in recent years, including a series of workshops and special issues in journals [3, 12].

As a metaphor that developing formal theory is akin to solving a *jigsaw puzzle*. Solving a jigsaw puzzle can be done as a solitary or team effort, it happens in a nonempirical context(e.g., at a large table to fit all the pieces), and the goal is to "fit" all pieces together.

4.8 Computer Simulations

The eighth research strategy is *computer simulation*, positioned in Quadrant IV. The goal of a computer simulation is to create a symbolic replica of a certain type of concrete system that can be executed by a computer [22]. In a

[1] Not to be confused with formal methods that are used, for example, to develop formal program specifications.

computer simulation of a real-world phenomenon or setting, everything is represented symbolically and created artificially. Where studies in natural settings (Quadrant I) can be costly or even impossible to conduct as they may involve a higher level of engagement from participants, computer simulations (which take place in a nonempirical setting) "are like virtual laboratories where hypotheses about observed problems can be tested" before they are implemented in real-world systems [16].
As a metaphor, they liken a computer simulation to a *forecasting system*, such as those used in weather prediction, which employ complex mathematical models of the atmosphere and oceans. Such systems are programmed to do a very specific thing based on a set of preprogrammed rules.

5 APPLICABILITY OF THE ABC FRAMEWORK TO SOFTWARE ENGINEERING RESEARCH

There are two well-established and important research areas within the software engineering literature called *Software Engineering (GSE)* and *Requirements Engineering (RE)* [25], where we can use the eight different research strategies.

5.1 Research Strategies in Global Software Engineering Research (GSE)

GSE has been actively studied since the 1990s, and research has focused primarily on the challenges associated with distributed development, whether as a result of offshoring or outsourcing strategies.

Field Study. A key issue in GSE is that of coordination of distributed teams as geographical, temporal, and sociocultural distances give rise to a variety of challenges. To investigate why coordination of distributed teams is so difficult and how associated challenges manifest in a real-world context, Herbsleb and Grinter conducted an in-depth case study at one division of an organization with teams in the United Kingdom and Germany [9]. The study concludes with a number of lessons learned; for example, Herbsleb and Grinter suggested bringing people from different locations together early on in a project.

Field Experiment. To ensure a high level of software quality, organizations can implement so-called validation activities, such as code inspections, peer review, and testing. In order to understand the impact of geographical distance on software quality when conducting such activities in a distributed fashion, Ebert et al. [2] conducted a field experiment. The authors found that colocating peer reviews improves defect detection; coaching within the project reduces cost of rework; teamwork and continuous build in the development process improves global project management.

Experimental Simulation. Bos and colleagues [2] conducted an *experimental simulation* using an online multi-player game to study the effect of colocation, the presence of multiple sites within a large company, collaboration across multiple sites, and the influence of social networks in these collaborations. Finally, they argued that colocated individuals would outperform isolated members of the team.

Laboratory Experiment. To study the impact of groupware support on the quality of software architecture evaluation deliverables Babar et al.2008 [14] conducted *laboratory experiment*. Using 32 teams of three undergraduate students each, Babar and colleagues found that the quality of the scenario profiles developed by the distributed teams using groupware tool support were, in fact, of higher quality than those developed by colocated teams.

Judgment Study. Iacovou and Nakatsu 2008 [10] set out “to produce a set of project risks that specifically applies to offshore outsourcing”. In particular, they adopted the judgment study strategy, implemented as a Delphi study “to solicit and analyze the input of the expert panelists” [10]. The study identified 25 risk factors, which were ranked in importance according to the panel. The analysis indicated a statistically significant and high level of consensus regarding the risk factors.

Sample Study. Ma et al.2008 [11] investigated how well projects outsourced to Chinese suppliers actually performed in terms of language barriers, communication, and overtime work. To that end, they adopted the sample study strategy, using a questionnaire for data collection. Finally, they concluded that language not a major obstacle; email used for development issues and face-to-face meetings to discuss management issues/requirements; reasons for overtime are requirement changes and underestimation of effort.

Formal Theory. To investigate the implications of temporal distance for distributed teams, Espinosa and Carmel presented a theoretical discussion from different perspectives [7], leading to a model that affords a unified view of coordination challenges in time-separated contexts, which provides a conceptual foundation

Computer Simulation. To investigate, Setamanit et al. conducted a *computer simulation* evaluate the choice of task allocation strategy and its impact on project duration [19, 21]. The researchers first investigated the “ideal” circumstances, without any communication and cultural barriers. Finally, they concluded that increasing overlap of work hours contributes to shorter duration for module-based and phase-based strategy, and a longer project duration for FTS strategy.

5.2 Research Strategies in Requirements Engineering Research

This section will explain that, how we can use the different research strategies with a second research area: Requirements Engineering(RE). RE has long been one of the core areas within software engineering research, with a first special issue in IEEE Transactions on Software Engineering in 1977 [20].

To investigate these new challenges, Damian and Zowghi conducted a *field study* at one large multi-site organization [4]. One of the authors spent several months on-site at the case organization to develop an in-depth understanding of the specific context. Data were collected through document study, observation, and interviews. The authors report on the specifics of the context, including the organizational structure and collaboration technologies.

Lauesen and Vinter [13] conducted a *experimental seimulation* to “find cost-effective ways to avoid requirement defects in the products”. Their study setting was *natural* and found several conclusions, incl.: ”scenarios and early usability testing are beneficial techniques”.

The findings of *experimental simulation* by Lerch et al. [8] was ”Insights into different information needs and search strategies and decision-making strategies depending on users’ expertise. Insights on performance with feedback/feedforward”.

To investigate the hypothesis that scenario-based inspections are more effective than ad hoc inspections Porter et al. [1] conducted a *laboratory experiment* and concluded with 4 key findings, incl.: scenario-based method leads to higher fault detection rate than other methods; scenario reviewers were more effective at detecting faults the scenarios were designed to uncover.

Judgment Study was adopted by Daneva and Ahituv [5], to evaluate a set of 12 practices based on feedback by ERP practitioners. To do this 10 consultants from 7 ERP services firms, selected based on their interest and expertise. This study validated all 12 practices.

Neill and Laplante [17] adopted the *sample study* strategy by conducting an industry survey to assess the state of practice of RE. Data were collected through an online questionnaire with 22 questions. From the 1,519 invited persons, 194 completed the survey. Findings include organization and participant characteristics (various domains; participants held variety of positions); software development life cycle model (agile, waterfall,etc.); RE techniques.

To develop an understanding of the role of creativity in RE Nguyen and Shanks [18] developed a theoretical framework. The framework development draws on the wider creativity research literature and the literature on creativity in the RE field. The resulting framework consists of five elements: product, process, domain, people, and context.

Höst et al. [15] adopted the *computer simulation* strategy to investigate bottlenecks and overload in RE processes.The authors concluded that "congestion" in the process can be avoided, either by increasing the number of staff or productivity or by limiting the rate of new requirements through, for example, requirements prioritization.

5.3 Analysis of a Sample of Studies

To demonstrate a wider applicability of the framework Stol et al. [25] analyzed all articles published in 2017 in Springer's Empirical Software Engineering journal. Their analysis shows that all studies can be mapped to the ABC framework. Table 2 shows that Sample Studies (n = 38) and Laboratory Experiments (n = 30) are most widely used. Field Studies were reported in 10 articles, and only a few articles reported Experimental Simulations and Judgment Studies. No instances of Field Experiments, Formal Theories, or Computer Simulations in this particular sample.

Table 2. Distribution of Research Strategies of the Analyzed Sample [25].

Strategy	Counts
Field Study	10
Experimental Simulation	0
Experimental Simulation	4
Laboratory Experiment	30
Judgment Study	3
Sample Study	38
Formal Theory	0
Computer Simulation	0

6 DISCUSSION AND CONCLUSION

6.1 Metaphors and Research Settings in Software Engineering

Metaphors are widely used in the software engineering literature and practice; indeed, the term "software engineering" itself can be considered a metaphor. Metaphors are powerful pedagogical devices that can help to convey the essence of a term or entity. However, they are usually limited in the extent to which they can fully capture the similarity with the real-world entity they represent [25]. In this paper, eight metaphors used are so far.
Field studies used *jungle*, as it allows a researcher to investigate a real and concrete instance of a phenomenon. The proposed metaphor for *field experiments* is *nature reserve*—like a jungle, it represents a natural setting, yet it allows a researcher to intrude on the setting by making changes to the "living circumstances" of some inhabitant. In the *experimental simulation* strategy, the researcher's borrowed the *greenhouse* metaphor [25], where a researcher has a controlled environment that simulates a particular type of concrete setting. The *laboratory experiment* strategy is characterized by an even higher level of intrusion by the researcher. That's why *test tube* metaphor used [22]. Both *judgment studies* and *sample studies* are conducted in neutral settings. Here *courtroom* and *referendum* metaphors were introduced for *judgment studies* and *sample studies* respectively. Finally, the *formal theory* and *computer simulation* strategies are nonempirical strategies. The development of formal theory was compared with to solving a *jigsaw puzzle*. On the other hand, a computer simulation is very similar to *forecasting systems*, such as used for weather prediction and stock markets.

6.2 Limitations of the ABC Framework

The ABC framework, though rooted in the social sciences, is also useful to SE research. This article demonstrated the fit of the ABC framework on two worked examples in Section 5. Though this framework gives of hope of selecting research strategies, the following limitations are not also ignorable.

First, the ABC framework does not provide guidance for specific methods, Instead, the ABC framework provides a holistic overview of different research strategies, each with specific characteristics, positioned along two key dimensions that facilitates a systematic comparison. As a researcher selects a specific method (such as a controlled experiment or an ethnography, for example), the researcher still needs to consider a variety of research design considerations that are specific to those methods. This is true also during operationalization of a study: for example, a sample study can involve human respondents or using a sample of software process artifacts (typically gathered through repository mining).

Second, the ABC framework presents archetypal research strategies, which is to say that other researchers may design studies that do not fall cleanly within one of the octants of the framework. There are certain research methods that represent a compromise between different strategies.

Finally, the ABC framework is an alternative view to software engineering research, not necessarily the best view. Previous classifications such as by Shaw [23] and Wieringa et al. [26] may be a better fit for positioning studies within a given research program, in particular for solution seeking studies.

6.3 Conclusion

The software engineering research community has made considerable progress in terms of the quality of studies that seek knowledge and understanding. Over the last several decades, the community has reflected on the way it conducts research and the methods to do that research. There are numerous guidelines for employing specific methods, and the variety of research methods has increased without a doubt. Nevertheless, there is still some confusion about terminology, and the various overviews of research methods are a “mixed bag” in that the various methods identified have not been carefully positioned in relation to one another. A holistic view of the landscape of research strategies to generate new knowledge and understanding has been missing so far in the SE research community. The ABC framework represents a holistic overview of eight archetypal research strategies that is oriented around two key dimensions: the level of obtrusiveness of a study and the generalizability of the findings. Finally, the framework also clarifies the inherent weaknesses and potential strengths of the research strategies.

References

1. A. A. Porter, L.G.V., Basili, V.R.: Comparing detection methods for software requirements inspections: A replicated experiment. IEEE Trans. Software Eng. **21**, 363–575 (06 1995)
2. C. Ebert, C. H. Parro, R.S., Kolarczyk, H.: Better validation in a world-wide development environment. In: Proc. 7th International Software Metrics Symposium (METRICS’01) (2001)
3. D. I. K. Sjøberg, T. Dybå, B.C.D.A., Hannay, J.E.: Building theories in software engineering. In: Guide to Advanced Empirical Software Engineering. Forrest Shull, Janice Singer, and Dag I. K. Sjøberg (Eds.). Springer-Verlag London Limited (2008)
4. Damian, D.E., Zowghi, D.: Re challenges in multi-site software development organisations. IEEE Trans. Softw. Eng. p. 149–160 (08 2003)
5. Daneva, M., Ahituv, N.: A focus group study on inter-organizational erp requirements engineering practices. In: Proc. 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement. ACM (2010)
6. E. J. Webb, D. T. Campbell, R.D.S., Sechrest, L.: Research on Human Behavior: A Systematic Guide to Method. Rand-McNally (1972)
7. Espinosa, J.A., Carmel, E.: The impact of time separation on coordination in global software teams: A conceptual foundation. Software Process. Improve. Pract **8**, 249–266 (04 2003)

8. F. J. Lerch, D.J.B., Harter, D.E.: Using simulation-based experiments for software requirements engineering. Ann. Software Eng. **03**, 345–366 (01 1997)
9. Herbsleb, J.D., Grinter, R.E.: Splitting the organization and integrating the code: Conway's law revisited. In: Proc. International Conference on Software Engineering. pp. 85–95 (1999)
10. Iacovou, C.L., Nakatsu, R.: A risk profile of offshore-outsourced development projects. commun. ACM 51 **6**, 89–94 (2008)
11. J. Ma, J. Li, W.C.R.C.J.J., Liu, C.: A state-of-the-practice study on communication and coordination between chinese software suppliers and their global outsourcers. Software Process Improve. Pract. **13**, 233–247 (03 2008)
12. K. Stol, M.G., Jacobson, I.: Introduction to the special section—general theories of software engineering: New advances and implications for research. IEEE Trans. Softw. Eng. **20**, 176–180 (2016)
13. Lauesen, S., Vinter., O.: Preventing requirement defects: An experiment in process improvement. Requir.Eng. p. 37–50 (06 2001)
14. M. A. Babar, B.K., Jeffery, R.: Comparing distributed and face-to-face meetings for software architecture evaluation: A controlled experiment. Empir. Software Eng **13**, 39–62 (01 2008)
15. M. Höst, B. Regnell, J.N.o.D.J.N., Nyberg, C.: Exploring bottlenecks in market-driven requirements management processes with discrete event simulation. J. Sys. Software **59**, 323–332 (3 2001)
16. Müller, M., Pfahl, D.: Simulation methods. In: Guide to Advanced Software Engineering. F. Shull, J. Singer, and D. I. K. Sjøberg (Eds.). Springer-Verlag London Limited (2008)
17. Neill, C.J., Laplante, P.: Requirements engineering: The state of the practice. IEEE Software **29**, 40–45 (06 2003)
18. Nguyen, L., Shanks, G.: A framework for understanding creativity in requirements engineering. Inform. Software Tech **51**, 655–662 (3 2008)
19. Raffo, D., Setamanit, S.O.: A simulation model for global software development project. In: Proc. 6th International Workshop on Software Process Simulation and Modeling (ProSim'05). pp. 429–436. Fraunhofer IRB Verlag (2005)
20. Ross, D.T.: Comparing distributed and face-to-face meetings for software architecture evaluation: A controlled experiment. IEEE Trans. Software Eng **3**, 2–5 (02 1973)
21. S.-O. Setamanit, W.W., Raffo, D.: Using simulation to evaluate global software development task allocation strategies. Software Process. Improve. Pract p. 491–503 (12 2007)
22. Sharp, H., Robinson, H.: An ethnographic study of XP practice. Empir. Software Eng. **9**, 353–375 (07 2004)
23. Shaw, M.: What makes good research in software engineering? Int. J. Software Tools Technol. Transf. **4**, 1–7 (01 2002)
24. Stol, K., Fitzgerald, B.: Theory-oriented software engineering. Sci. Computer Program **101**, 79–98 (2015)
25. Stol, K.J., Fitzgerald, B.: The abc of software engineering research. ACM Trans. Softw. Eng. Methodol. **27**(3) (Sep 2018). https://doi.org/10.1145/3241743, https://doi.org/10.1145/3241743
26. Wieringa, R., Maiden, N., Mead, N., Rolland, C.: Requirements engineering paper classification and evaluation criteria: A proposal and a discussion. Requir. Eng. **11**(1), 102–107 (Dec 2005). https://doi.org/10.1007/s00766-005-0021-6, https://doi.org/10.1007/s00766-005-0021-6
27. Wilhelm Hasselbring, S.G.: Research Methods in Software Engineering. Trustworthy Software Systems (2006)