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.

1 Introduction

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). The ABC framework uses two dimensions widely considered to be key in research design: the level of obtrusiveness of the research and the generalizability of research findings.

Most software engineering research is concerned with the same three components [32]:

- Actors (A), which includes software professionals, software systems, and their users
- Their behavior (B) (e.g., coordination among developers; developer productivity and antecedents, e.g., motivation; systems' performance and other quality attributes)
- The context (C) of a specific system or organization

2 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.

2.1 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. Parnas lamented the lack of experimental design in current empirical software development research and how observations from a small number of "uncontrolled case studies" do not contribute to scientific knowledge [24] .

2.2 Generalizability

A second key concern that authors have expressed is the level of generalizability of research findings [32]. 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.

3 THE ABC FRAMEWORK FOR RESEARCH STRATEGIES

This section presents a framework that provides a holistic overview of eight different research strategies. The term ABC framework (Figure 1), was originally devised by McGrath and his colleagues for the social sciences [17–19, 27]. The authors interpreted, tailored, and operationalized the framework for a software engineering research context [32]. Specifically, they refined the terminology used in the framework for - for example, in the social sciences the term ”actor” refers to person. Whereas in SE, actors may represent professionals, applications and systems, or their users. To operationalize the framework, they identified exemplar studies that adopted the archetypal research strategies in two SE research areas (global software engineering and requirements engineering).

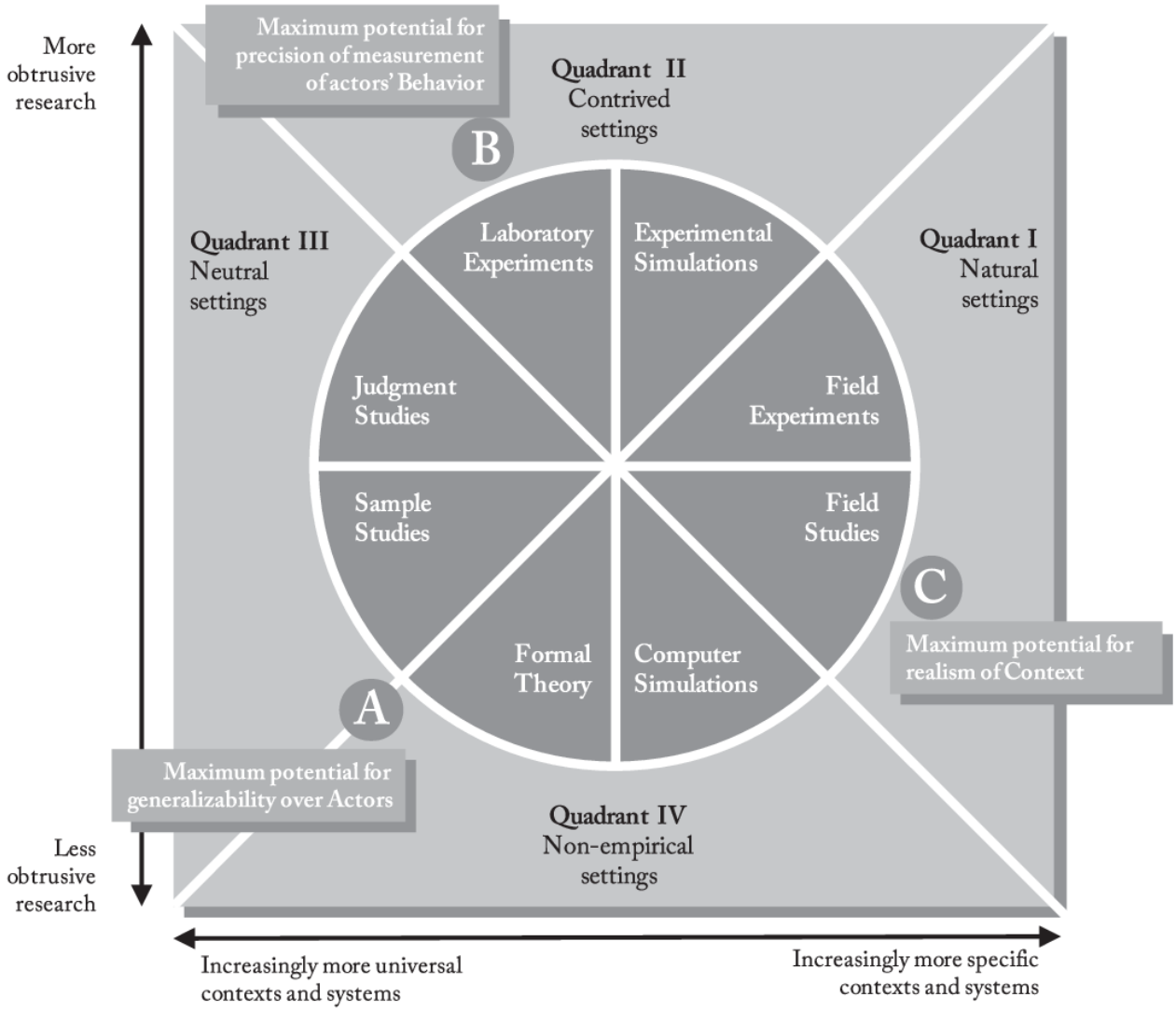

Fig. 1. The ABC framework [32]: eight research strategies as categories of research methods for software engineering (adapted from Runkel and McGrath [7]). Quadrants I to IV represent different research settings.

The two key dimensions *obtrusiveness* and *generazability* of ABC framework already discussed in previous section (2). The first dimension (the x-axis in Figure 1) is that of generalizability of 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). For each strategy the authors of the article [32] discussed a metaphor that, in their opinion, conveys the essence

of that strategy and which may help in distinguishing the eight strategies. For example, they adopted the ”jungle” metaphor from this article’s opening quote to represent field studies.

3.1 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).

They chose *jungle* as metaphor 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.

3.2 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.

They suggest a *nature reserve* as a metaphor 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.

3.3 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* [29]. 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* [29]. 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.

3.4 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 [29]. 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 [29] used to set it clearly apart from the greenhouse (experimental simulation, representing a more continuous flow of events) and nature reserve (field experiment).

An example of a laboratory experiment is a study on pair programming [6]. For this study, 295 Java consultants with varying levels of seniority were hired to participate. The task at hand was to make changes to two alternative Java systems with different levels of complexity. It is clear that the study setting was contrived; the 295 participants would not have undertaken these tasks without explicitly being requested (and paid). The change tasks themselves were also carefully designed as part of the study.

3.5 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).

3.6 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 (see Table 3). 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.

3.7 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 [31]. 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, 13].

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.

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

3.8 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 [29]. In a 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 [20].

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.

4 APPLICABILITY OF THE ABC FRAMEWORK TO SOFTWARE ENGINEERING RESEARCH

To illustrate the use of different research strategies in software engineering research, They presented examples from two different research areas: Global Software Engineering (GSE) and Requirements Engineering (RE) [32]. They selected these topics because they are well-established and important research areas within the software engineering literature, evidenced by their dedicated conferences(ICSGSE, RE). For both areas, they selected exemplar studies that clearly illustrate the eight research strategies. They emphasize that they do not intend to cast any judgment on the cited examples. However, for each example study, they discuss some inherent limitations that are a consequence of the selected research strategy—rather than due to poor research design decisions of the authors.

4.1 Research Strategies in Global Software Engineering Research

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. Table 1, discusses examples of different research strategies used in global software engineering research.

Table 1: Examples of Different Research Strategies Used in Global Software Engineering Research [32].

Strategy	Authors	Objective	Study Setting	Study Procedure	Findings
Field Study	Herbsleb and Grinter 1999 [10]	To understand why geographically distributed development is difficult to coordinate.	*Natural*: Two ocations of a division of Lucent Technologies	18 interviews, archival sources, documents.	Coordination mechanisms; barriers to informal communication;
Experimental Simulation	Bos et al.2004 [2]	To investigate the impact of colocation, coaching.	*Natural*: Alcatel's Switching and Routing business unit.	Comparison of inspections in colocated and distributed teams;	Colocating peer reviews improves defect detection;
Experimental Simulation	Bos et al. 2004 [21]	To study the effect of colocation, the presence of multiple sites within a large company.	Contrived: online multiplayer game (the Shape Factory simulation environment).	13 simulation sessions with 5 rounds each; 10 players per session, 130 participants in total.	Colocated participants collaborated more with each other than with telecommuters.

Table 1 – Continue

Strategy	Authors	Objective	Study Setting	Study Procedure	Findings
Laboratory Experiment	Babar et al.2008 [15]	To study the impact of groupware support on the quality of software architecture.	*Contrived:* experimental tasks part of assessed course tasks.	Controlled experiment, AB/BA crossover design; 32 teams of 3 participants.	Quality of deliverables from the distributed meeting groups was good.
Judgment Study	Iacovou and Nakatsu 2008 [11]	To investigate risk factors for offshore-outsourcing software development.	*Neutral:* systematically selected panel of experts at a variety of organizations.	Delphi study, 15 experts, 3 rounds: identification; rating;	25 risk factors that could influence the success of an offshore-outsourced project.
Sample Study	Ma et al.2008 [12]	To investigate 3 issues in software development by Chinese software suppliers.	*Neutral:* questionnaire sent out to companies by email.	Random sample of 2,000 from a database of approx.	Language not a major obstacle; email used for development issues.
Formal Theory	Espinosa and Carmel 2003 [8]	To develop a conceptual foundation for future research on GSE.	*Nonempirical:* desk research without any direct empirical observations	Theorizing and conceptualization based on Coordination Theory	A model of coordination costs due to time differences in dispersed software teams.
Computer Simulation	Setamanit et al. 2007 [25, 28]	To evaluate the choice of task allocation strategy.	*Nonempirical:* GSD simulator with which the researchers can model several factors.	For each of 3 strategies: 5 replications for each design point.	1,920 runs in total. Increasing overlap of work hours contributes to shorter.

4.2 Research Strategies in Requirements Engineering Research

In this section, they illustrate the different research strategies with a second research area: Requirements Engineering [32]. 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 [26].

Table 2: Examples of Different Research Strategies Used in Requirements Engineering Research [32].

Strategy	Authors	Objective	Study Setting	Study Procedure	Findings
Field Study	Damian and Zowghi [4]	To investigate the impact of distributed stakeholders on RE activities in GSD.	*Natural*: first author spent 7 months on-site at an organization.	Document study, observation, interviews.	Four major problems and 8 specific challenges related to requirement engineering activities.

Table 2 – Continue

Strategy	Authors	Objective	Study Setting	Study Procedure	Findings
Experimental Simulation	Lauesen and Vinter [14]	To identify a cost-effective way to avoid requirement defects.	*Natural*: company staff and researcher collaborated on-site, using real products to evaluate new approaches.	Action research, data include defect reports, time spent, usability issues, timeliness ofthe project, product sales.	Several conclusions, incl.: scenarios and early usability testing are beneficial techniques;
Experimental Simulation	Lerch et al. [9]	To identify the computer support needs of automation staff in a large organization.	*Contrived*: simulation environment with experimental stimuli that were previously collected.	3 distinct groups of 3 people each to gauge how level of expertise and circumstances affect behavior. Participants received training.	Insights into different information needs and search strategies and decision-making strategies depending on users' expertise. Insights on performance with feed-back/feedforward.
Laboratory Experiment	Porter et al. [1]	To investigate the hypothesis that scenario-based inspections are more effective than ad hoc inspections.	*Contrived*: classroom laboratory exercise with graduate students doing a course in formal methods	Measurement of effect of detection methods (ad hoc, checklist,scenario) on 4 dependent variables incl. fault detection rate.	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	Daneva and Ahituv [5]	To evaluate a set of 12 practices based on feedback by ERP practitioners.	*Neutral:* dedicated meeting room with seating around a table.	10 consultants from 7 ERP services firms, selected based on their interest and expertise.	All 12 practices were observed by several of the panel experts.
Sample Study	Neill and Laplante [22]	To investigate the state of practice of requirements engineering in industry.	*Neutral:* web-based questionnaire.	22 questions; participants drawn from a Penn State School of Graduate Professional Studies database.	Findings include organization and participant characteristics (various domains);

Table 2 – Continue

Strategy	Authors	Objective	Study Setting	Study Procedure	Findings
Formal Theory	Nguyen and Shanks [23]	To develop an understanding of the role of creativity in RE.	*Nonempirical:* no empirical observations, but derivation of a conceptual framework from literature.	General creativity literature, requirements engineering creativity literature	A theoretical framework that offers RE researchers a basis to incorporate creativity in RE methods and techniques.
Computer Simulation	Höst et al. [16]	To investigate bottlenecks and overload in RE processes	*Nonempirical:* a discrete event simulator was implemented SDL	Four simulation scenarios with different parameter values to model different circumstances	Two ways were identified to avoid congestion: (1) increase number of staff or productivity, (2) decrease the rate of new requirements.

4.3 Analysis of a Sample of Studies

Their analysis above is based on a convenience sample; they selected studies that exemplify the eight research strategies. In order to demonstrate a wider applicability of the framework, they analyzed all articles published in 2017 in Springer's Empirical Software Engineering journal. The following Table 3 shows the distribution of research strategies of the Analyzed Sample.

Table 3. Distribution of Research Strategies of the Analyzed Sample [32].

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

5 DISCUSSION AND CONCLUSION

5.1 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 [32]. In this paper, eight metaphors used 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 [32], 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 [29]. 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.

5.2 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 4. 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 [30] and Wieringa et al. [33] may be a better fit for positioning studies within a given research program, in particular for solution seeking studies.

5.3 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. Arisholm, H. Gallis, T.D., Sjøberg, D.I.K.: Evaluating pair programming with respect to system complexity and programmer expertise. IEEE Trans. Softw. Eng. **33**, 65–86 (02 2007)
7. E. J. Webb, D. T. Campbell, R.D.S., Sechrest, L.: Research on Human Behavior: A Systematic Guide to Method. Rand-McNally (1972)
8. 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)
9. 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)
10. 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)
11. Iacovou, C.L., Nakatsu, R.: A risk profile of offshore-outsourced development projects. commun. ACM 51 **6**, 89–94 (2008)
12. 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)
13. 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)
14. Lauesen, S., Vinter., O.: Preventing requirement defects: An experiment in process improvement. Requir.Eng. p. 37–50 (06 2001)
15. 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)
16. 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)
17. McGrath, J.E.: Towards a "Theory of Method" for research on organizations. In: New Perspectives in Organization Research. pp. 533–556. W. Cooper, H. Leavitt, and M. Shelly (Eds.). John Wiley & Sons, New York (1964)
18. McGrath, J.E.: Groups: Interaction and Performance. Prentice-Hall (1984)
19. McGrath, J.E.: Methodology matters: Doing research in the behavioral and social sciences. In: Readings in Human-Computer Interaction: Toward the Year 2000. pp. 152–169. Ronald M. Baecker (Ed.). Morgan Kaufmann (1994)
20. 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)
21. N. Bos, N. Sadat Shami, J.S.O.A.C., Nan, N.: In-group/out-group effects in distributed teams: An experimental simulation. In: Proc. International Conference on Computer-Supported Cooperative Work and Social Computing (CSCW'04). pp. 429–436. ACM, New York (2004)
22. Neill, C.J., Laplante, P.: Requirements engineering: The state of the practice. IEEE Software **29**, 40–45 (06 2003)
23. Nguyen, L., Shanks, G.: A framework for understanding creativity in requirements engineering. Inform. Software Tech **51**, 655–662 (3 2008)
24. Parnas, D.L.: Point/counterpoint: Empirical research in software engineering: A critical view. IEEE Software **26**, 56–59 (06 2009)
25. 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)
26. 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)
27. Runkel, P.J., McGrath, J.E.: Research on Human Behavior: A Systematic Guide to Method. Holt, Rinehart and Winston (1972)
28. 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)
29. Sharp, H., Robinson, H.: An ethnographic study of XP practice. Empir. Software Eng. **9**, 353–375 (07 2004)
30. Shaw, M.: What makes good research in software engineering? Int. J. Software Tools Technol. Transf. **4**, 1–7 (01 2002)
31. Stol, K., Fitzgerald, B.: Theory-oriented software engineering. Sci. Computer Program **101**, 79–98 (2015)
32. 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
33. 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