The ABC of Software Engineering Research

Faiz Ahmed

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

Abstract. In this paper,I present the ABC framework, which is adapted from the taxonomy developed by McGrath and his colleagues in the social sciences, and seek 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. I will discuss metaphors for each strategy and their inherent limitations and potential strengths.

Keywords: SE · Extreme Programming · Global Software Engineering.

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.

2 Dimensions of Research Strategies

Two important dimensions of 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.

2.2 Generalizability

A second key concern that authors have expressed is the level of gen–eralizability of research findings. This has been a recurring concern in software engineering re–search, in particular in the context of case studies—captured succinctly by one referee: "Case study reports are [. . .] limited, because they report a single case."

3 THE ABC FRAMEWORK FOR RESEARCH STRATEGIES

This section presents a framework that provides a holistic overview of eight different research strategies. The framework, which we have termed the ABC framework (Figure 1), was originally devised by McGrath and his colleagues for the social sciences [1–4].

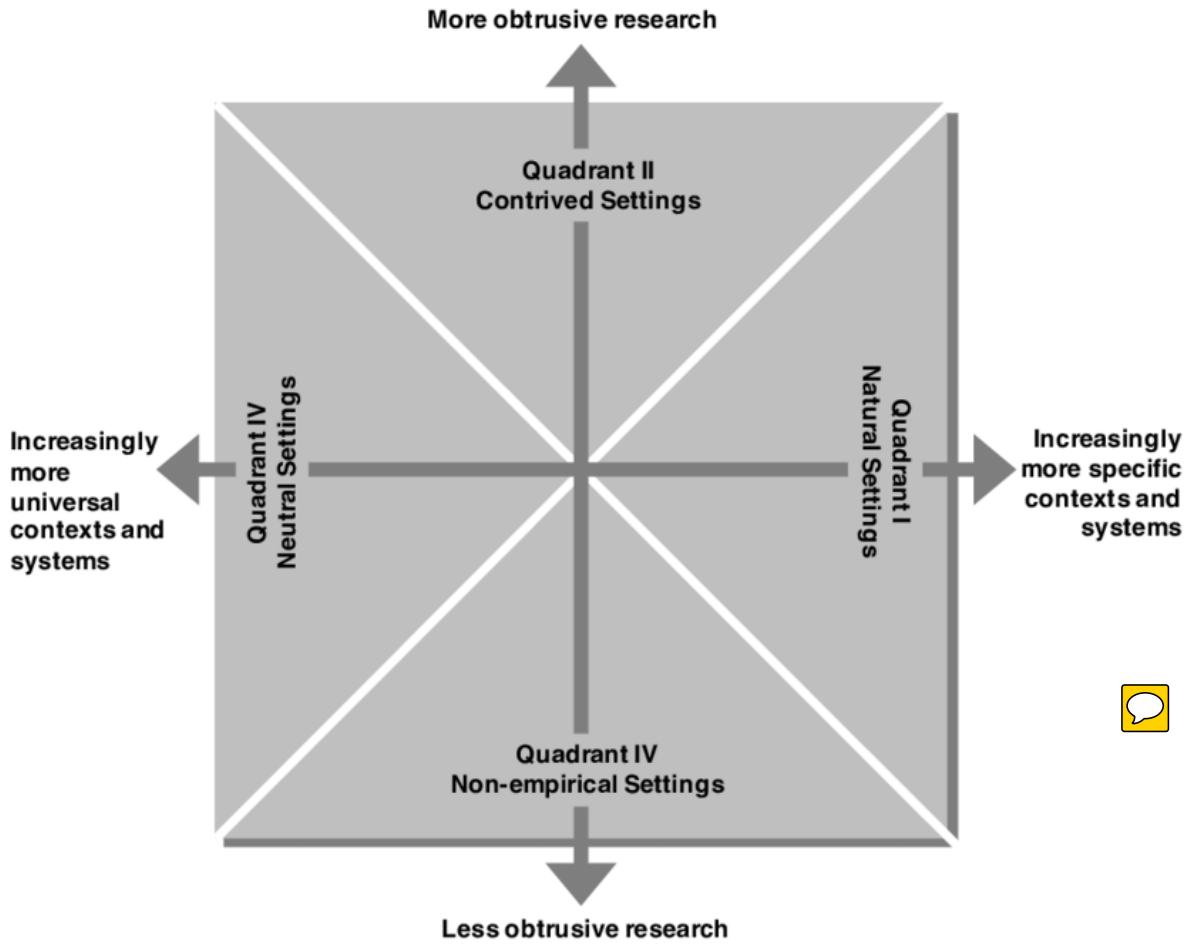

Fig. 1. The ABC framework: eight research strategies as categories of research methods for software engi-neering (adapted from Runkel and McGrath [5]). Quadrants I to IV represent different research settings

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.

One example of a field study in SE is an ethnographic study of XP (Extreme Programming) by Sharp et al [6] . This study did not aim to "control" any variables, nor to change anything in the case study setting. Rather, the authors stated that their "motivation is to gain insight into the culture and community of an agile method." The metaphor we chose for field studies is the *jungle*.

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.

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 natu-ral.

I borrow Runkel and McGrath's metaphor, who compared an experimental simulation to a greenhouse [6]. 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.

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 [6]. 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 [6] used to set it clearly apart from the greenhouse (experimental simulation, representing a more continuous flow of events) and nature reserve (field experiment)

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.
I liken a judgment study to 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.

3.6 Sample Studies

Also in Quadrant III is the *sample study* strategy, which 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 Quadrant III in Figure 1.

A metaphor for the sample study strategy is a *referendum* (though we 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 [7].
As a metaphor, we propose 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.

3.8 Computer Simulations

The eighth research strategy is *computer simulation*, also 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 [6]. In a computer simulation of a real-world phenomenon or setting, everything is represented symbolically and created artificially.
As a metaphor, we 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, we present examples from two different research areas: Global Software Engineering (GSE) and Requirements Engineering (RE). Table 1, discusses examples of Different Research Strategies Used in Global Software Engineering Research.

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.

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

Table 1. Examples of Different Research Strategies Used in Global Software Engineering Research.

Strategy	Authors	Objective	Study Setting	Study Procedure	Findings
Field Study	Herbsleb and Grinter 1999 [8]	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 [10]	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 [9]	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.
Laboratory Experiment	Babar et al.2008 [16]	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 [13]	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 [14, 15]	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, we illustrate the different research strategies with a second research area: Requirements Engineering. 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 [17]. The following Table 2 shows the Distribution of Research Strategies of the Analyzed Sample.

Table 2. Distribution of Research Strategies of the Analyzed Sample

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

4.3 Analysis of a Sample of Studies

Our analysis above is based on a convenience sample; we selected studies that exemplify the eight research strategies. In order to demonstrate a wider applicability of the framework, we analyzed all articles published in 2017 in Springer's Empirical Software Engineering journal.

5 DISCUSSION AND CONCLUSION

5.1 Metaphors and Research Settings in Software Engineering

or each research strategy, we presented a metaphor representing that strategy within its research setting (see Section 3), some of which were proposed by others [4, 18].

5.2 The A, B, and C of Software Engineering Research

The ABC framework positions eight archetypal research strategies that SE researchers can use,in particular for what we have termed knowledge-seeking studies (see Section 1). The framework is based on earlier work in the social sciences [1–5], which we have adapted and operationalized for a software engineering context.

5.3 Limitations of the ABC Framework

Given the socio-technical nature of the software engineering field, we argue that the framework, though rooted in the social sciences, is also useful to SE research. We have demonstrated the fit of the ABC framework on two worked examples in Section 4. In addition, we analyzed a sample of 75 articles published in 2017 in Empirical Software Engineering; we classi-fied all knowledge-seeking studies using the ABC framework. Notwithstanding the fit of the ABC framework, some limitations of the framework should be borne in mind.

5.4 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. In this article, we adopt a framework from the social sciences that we have labeled the ABC framework—as such, it contributes to the literature on research methodology for software engineering.

References

1. J. E. McGrath. 1964. Towards a "Theory of Method" for research on organizations. In New Perspectives in Organization Research, W. Cooper, H. Leavitt, and M. Shelly (Eds.). John Wiley & Sons, New York, 533–556
2. J. E. McGrath. 1984. Groups: Interaction and Performance. Prentice-Hall.
3. J. E. McGrath. 1994. Methodology matters: Doing research in the behavioral and social sciences. In Readings in Human-Computer Interaction: Toward the Year 2000, Ronald M. Baecker (Ed.). Morgan Kaufmann, 152–169.
4. P. J. Runkel and J. E. McGrath. 1972. Research on Human Behavior: A Systematic Guide to Method. Holt, Rinehart and Winston.
5. P. J. Runkel and J. E. McGrath. 1972. Research on Human Behavior: A Systematic Guide to Method. Holt, Rinehart and Winston
6. H. Sharp and H. Robinson. 2004. An ethnographic study of XP practice. Empir. Software Eng. 9, 4 (2004), 353–375
7. K. Stol and B. Fitzgerald. 2015. Theory-oriented software engineering. Sci. Computer Program. 101 (2015), 79–98
8. J. D. Herbsleb and R. E. Grinter. 1999. Splitting the organization and integrating the code: Conway's law revisited. In Proc. International Conference on Software Engineering. 85–95.
9. N. Bos, N. Sadat Shami, J. S. Olson, A. Cheshin, and N. Nan. 2004. 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). ACM, New York, 429–436.
10. C. Ebert, C. H. Parro, R. Suttels, and H. Kolarczyk. 2001. Better validation in a world-wide development environment. In Proc. 7th International Software Metrics Symposium (METRICS'01).
11. C. L. Iacovou and R. Nakatsu. 2008. A risk profile of offshore-outsourced development projects. Commun. ACM 51, 6 (2008), 89–94.
12. J. Ma, J. Li, W. Chen, R. Conradi, J. Ji, and C. Liu. 2008. A state-of-the-practice study on communication and coordination between Chinese software suppliers and their global outsourcers. Software Process Improve. Pract. 13, 3 (2008), 233–247.
13. J. A. Espinosa and E. Carmel. 2003. The impact of time separation on coordination in global software teams: A conceptual foundation. Software Process. Improve. Pract. 8, 4 (2003), 249–266
14. D. Raffo and S.-O. Setamanit. 2005. A simulation model for global software development project. In Proc. 6th International Workshop on Software Process Simulation and Modeling (ProSim'05). Fraunhofer IRB Verlag.
15. S.-O. Setamanit, W. Wakeland, and D. Raffo. 2007. Using simulation to evaluate global software development task allocation strategies. Software Process Improve. Pract. 12 (2007), 491–503.
16. M. A. Babar, B. Kitchenham, and R. Jeffery. 2008. Comparing distributed and face-to-face meetings for software architecture evaluation: A controlled experiment. Empir. Software Eng. 13, 1 (2008), 39–62.
17. D. T. Ross. 1977. Guest editorial: Reflections on requirements. IEEE Trans. Software Eng. 3, 1 (1977), 2–5.
18. E. R. McLean. 1973. Comments on empirical studies of management information systems by Richard L. Van Horn. Data Base 5, 2 (1973), 181–182.