A holistic overview of software engineering research strategies

Klaas-Jan Stol, Brian Fitzgerald

K. Stol and B. Fitzgerald (2015) A Holistic Overview of Software Engineering Research Strategies. Proceedings of the 3rd International Workshop on Conducting Empirical Studies in Industry, co-located with ICSE '15. Florence, Italy.

A Holistic Overview of Software Engineering Research Strategies

Klaas-Jan Stol and Brian Fitzgerald
Lero—the Irish Software Research Centre
University of Limerick, Ireland
klaas-jan.stol@lero.ie, bf@lero.ie

***Abstract*—Empirical research studies are the principal mechanism through which the software engineering research community studies and learns from software engineering practice. The focus on empirical studies has increased significantly in the past decade, more or less coinciding with the emergence of evidence-based software engineering, an idea that was proposed in 2004. As a consequence, the software engineering community is familiar with a range of empirical methods. However, while several overviews exist of popular empirical research methods, such as case studies and experiments, we lack a 'holistic' view of a more complete spectrum of research methods. Furthermore, while researchers will readily accept that all methods have inherent limitations, methods such as case study are still frequently critiqued for the lack of control that a researcher can exert in such a study, their use of qualitative data, and the limited generalizability that can be achieved. Controlled experiments are seen by many as yielding stronger evidence than case studies, but these can also be criticized due to the limited realism of the context in which they are conducted. We identify a holistic set of research methods and indicate their strengths and weaknesses in relation to various research elements.**

I. Introduction

The importance of conducting empirical research has been increasingly highlighted in recent years [1]. To conduct empirical research, the software engineering research community has adopted several research methods, approaches and techniques from other fields. Traditionally, 'empirical research' in software engineering has been assumed to involve a strong emphasis on quantitative and experimental research [1]–[3]. Indeed, in their review published in 2002, Glass et al. concluded that SE research is *"narrow regarding research approach and method"* [4]. However, more recently, a broader range of approaches have been introduced, including qualitative approaches such as grounded theory studies, ethnographies, and delphi studies [5].

There is much disagreement and confusion about terminology in software engineering research, and a common taxonomy is lacking. For instance, when speaking of research methods, it would be common to list case studies, controlled experiments, interview studies, grounded theory studies, and observational studies. However, these methods and techniques are, in fact, a "mixed bag" [6], in that they are not at the same level of abstraction. Some researchers refer to 'case study' as a 'strategy' or an 'approach'; others speak of it as a research method. Likewise, an interview study is a field study that uses interviews as a data collection technique. Many 'case studies,' however, also use interviews—while it is recommended to use multiple data sources in a case study, this is not necessarily required. What, then, is the difference between a case study and an interview study? Sometimes, the differences can be very small indeed.

We argue that limiting a study's characterization as a 'case study' or an 'experiment' does not add that much value in terms of distinction. Case studies, for instance could be descriptive, exploratory or evaluative, and even within each of those, the level of 'control' that a researcher (believes he or she) exerts varies also. Rather than discussing research *methods*, we believe it is better to think of a 'research strategy,' which represents a higher-level design of the study, similar to the concept of an architecture (of a building or a software system). Similar to how a software architecture has a significant impact on a system's quality attributes such as performance and reliability [7], a research strategy, too, has a significant impact on what can and cannot be achieved in a study in terms of acquiring new insights and a deeper understanding of phenomena. This awareness is not widely acknowledged in the SE research community. For instance, researchers have argued for more realism in software engineering experiments [8], specifically with respect to participants (or *actors*) (e.g., students v. professionals), tasks (or, what we could also describe as *behavior*) (e.g., toy problems v. real problems) and environments, or *contexts* (e.g., pen and paper v. industrial development environment). However, optimizing a study to consider all three (actors, behavior and context) is impossible. McGrath characterized this problem as a *"three-horned dilemma"* [6]. Ideally, a researcher optimizes a research design that can generalize to a large population (the 'actors,' A), the task (or 'behavior,' B) and the context (C). However, as McGrath stated [6]: *"there is no way—in principle—to maximize all three (conflicting) desiderata of the research strategy domain."*

The choice of research strategy is an important one, and one that has received significant attention in the SE research community in recent years. Several authors have provided guidance in this matter [9]–[11]. Existing overviews are excellent starting points to learn about specific methods such as case study methodology,[1] surveys, and action research. However, these overviews tend to lack a 'big picture' of how these different strategies relate to one another. In this

[1]We are aware of the variety of case study approaches and agree that there is no single 'case study' method; however, we believe there is increasingly common agreement on what 'case study' means in SE research.

paper we focus on strategies to conduct *primary* research, and therefore do not consider strategies to conduct *secondary* studies including systematic literature reviews.

This paper proceeds as follows. We discuss related work in Section II. Section III presents a framework for research strategies in software engineering research. Section IV discusses the different research settings in which the different strategies operate, which also offers more insights into the differences among the strategies. Section V concludes the paper by discussing implications for software engineering research.

II. Related Work

The choice of research strategy is an essential one for any researcher. A common problem is that researchers use what they know best—after sufficient years of training with a hammer, everything looks like a 'nail.' However, choosing a research strategy without careful consideration will cause many difficulties later on during the research.

While the SE community has always conducted empirical studies, there has been a significant increase and sophistication in the conduct of empirical studies since 2004, following the publication of the "Evidence-Based Software Engineering" paper by Kitchenham et al. [12] who looked at evidence-based medicine for inspiration. Since then, the SE community has repeatedly advocated empirical studies that provide useful evidence as to what works and what doesn't—mirroring the rationale of evidence-based medicine.

There have been numerous discussions of research methods and techniques inside and outside the software engineering community. For instance, Edmondson and McManus argued that the choice of research method must be appropriate given the current state of prior theory and research [13]. Marshall and Rossman argued that the choice depends on the study's purpose [14].

Within the SE research community, several authors have presented overviews of available methods [9], [10], [15], [16]. Others have argued that SE should adopt methods from other disciplines [17]. Based on an analysis of the research presented in the International Conference on Software Engineering (ICSE), the premier outlet for SE research, Shaw categorized the research questions addressed and the research methods adopted to address those questions [15]. Wohlin and Aurum proposed a decision-making structure for selecting a research design [18].

Terminology has always been a challenge in research methodology. There is no single, commonly adopted taxonomy of research methods and techniques. A number of fundamental distinctions have been made, such as *desk* v. *field* research; *quantitative* v. *qualitative* research; and *primary* v. *secondary* research [19]. While these provide some basic hints as to an author's intention, they do not fully convey the details of the research strategy the researcher has in mind. Even agreement about seemingly trivial questions such as what constitutes a case study is not straightforward; as Easterbrook et al. [9] pointed out, *"There is much confusion in the SE literature over what constitutes a case study."* Indeed, as Easterbrook et al. and others [20] have observed, the term case study has been used as an empirical method to study a phenomenon (e.g., [21]) or as a 'worked example' (e.g., [22]). Expectations as to what represents a "good case study" also vary. For instance, we have received review comments saying that *"Case study reports are typically tantalizing [...] but limited, because they report a single case."* Others have lamented the *"[lack of] control usually required to do a good case study paper."*

The term 'survey' is equally problematic; in most cases this means a sample survey, but the term can also be used to refer to a literature review (e.g., [23]). Consequently, it is not always clear what is meant by 'research method,' 'research strategy,' and 'data collection' method or 'technique.' Is Grounded Theory a 'method' or a 'technique'?

Seaman presented one of the first in-depth overviews of "qualitative methods" for software engineering research, but we would argue that her set was not one of research methods, but rather of data collection methods and analysis techniques (e.g., interviews and 'constant comparison') [24]. Lethbridge et al. proposed a taxonomy of data collection techniques for field studies [25]. Their taxonomy was organized around the level of human intervention; that is, how much involvement of the researcher or participants is needed to collect data. Techniques varied from 'first degree' (e.g., interviews) to 'second degree' (e.g., instrumenting systems, fly-on-the-wall), to 'third degree' (e.g., document analysis).

As a result of this terminological confusion, the discussion thus far about different research methods and strategies tends to be what McGrath called a "mixed bag" [6]. The SE literature suffers from this, too. In a review of methods for evaluation, Zelkowitz [26] observed that *"many of the authors used terms like 'experiment,' 'case study,' 'simulation,' 'controlled,' etc. in very different ways."* It is quite common that researchers design and conduct a sound study for which no commonly agreed term is used. Table I presents a selection of "mixed bags" of methods that have been used as overviews of different methods in the SE literature by different authors. It becomes clear that the controlled experiment, case study and survey are the three best-known methods. In some instances, there is a further sophistication, distinguishing random controlled experiments from quasi-experiments, for example.

III. Research Strategies in Software Engineering

Runkel and McGrath derived a framework to position different research strategies [29]. The framework outlines two key dimensions that span four quadrants, and within each quadrant two alternative research strategies are identified, resulting in a total of eight research strategies. Fig. 1 shows this 'circumplex' (as Runkel and McGrath called it) of research strategies. The key dimensions are the level of 'obtrusiveness' and the level of universality (or particularity) of, what Runkel and McGrath called the "behavioral system." In software engineering research we are studying behavior, too, whether it be behavior of systems or of the people involved in developing those systems.

TABLE I
A "MIXED BAG": ALTERNATIVE RESEARCH METHODS IN SOFTWARE ENGINEERING

Glass et al. 2002 [4]	Zannier et al. 2006 [27]	Tonella et al. 2007 [28]	Easterbrook et al. 2008 [9]	Wohlin et al. 2012 [11]
Replicated	Controlled experiment	No experimentation	Controlled experiment	Survey
Synthetic	Quasi experiment	Experience report	Case study	Case study
Dynamic analysis	Case Study	Case study	Survey	Experiment
Simulation	Exploratory case study	Quasi experiment	Ethnography	
Project monitoring	Experience report	Randomized experiment	Action research	
Case study	Meta-analysis	Observational study		
Assertion	Example application			
Field study	Survey			
Literature search	Discussion			
Legacy data				
Lessons learned				
Static analysis				

The level of obtrusiveness is an appropriate dimension to position different research strategies on, as it indicates the level of involvement of participants in a study or the level of involvement that a researcher has in creating the study setting. In SE research, researchers often contrast the 'control' that can be had in experimental studies over the 'lack of control' in exploratory case studies, for instance. Understanding the level of involvement required from participants is important to know upfront, as this is a key consideration in planning the research.

Furthermore, the level of generalizability of a study is also a key consideration. For instance, field studies such as case studies are inherently limited to their context. For that reason, the external validity of case studies is often viewed as limited, applying only to the specific context in which the research took place. Sample surveys (to differentiate them from a 'survey' as a literature review), on the other hand, aim to achieve a representative sample so as to be able to achieve a high level of external validity. However, the level of depth and detail achieved in a sample survey is very limited when compared to the richness of data that can be collected in a case study.

Fig. 1 also contains three symbols, A, B, and C, which refer to the three aspects mentioned above, namely actors, behavior and context. The positioning of the symbols indicate where these concerns reach their 'maximum.' That is, generalizability with respect to population (actors) is at its maximum with general theory or sample surveys. Precision of measurement of behavior or control over a study is at its maximum in laboratory experiments. Realism of context is maximum in field studies. We will return to these points below.

Together, the level of obtrusiveness and generalizability span an area in which the various research strategies can be positioned. The remainder of this section presents the eight research strategies in further detail. In so doing, we assume the reader has a basic knowledge of different research methods such as case study, experiment, and ethnography.

A. Field Studies

We use the term Field Study to refer to any research conducted in a real-world setting to study software professionals or software systems. A very common research method within this strategy is that of the exploratory case study, which focuses on a particular phenomenon, organization or system. Among the most highly cited case studies is one of the first exploratory case studies on open source software (OSS) development by Mockus et al. who studied the Apache web server project [30]. Field studies, as they are defined here, are unobtrusive. That is, they do not include any deliberate modification of the environment in which the research is conducted.

Another example of a field study in software engineering is the ethnographic study of XP (Extreme Programming) [31]. 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."*

There are many other studies in the SE literature that do not actively intervene, but compare the output of two different ways of working and for which archival data are available. Although sometimes these are reported as industry experiments, in our view these are field studies. For instance, one study investigated the utility of Test-Driven Development (TDD) in an industry context [32]. As the authors pointed out, the *"projects were*

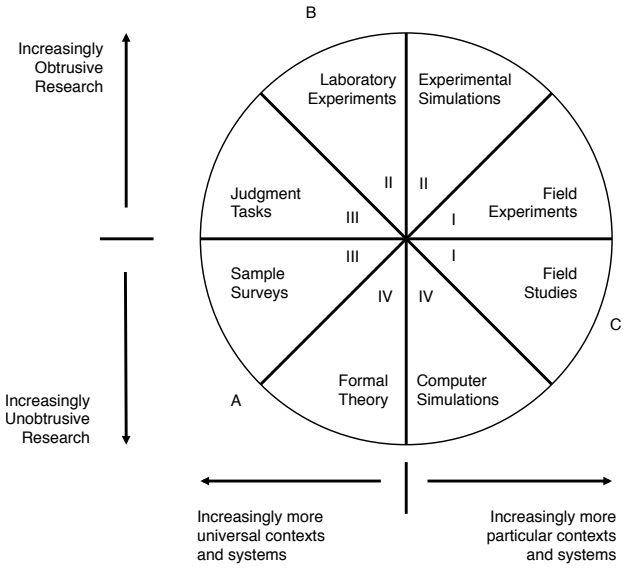

Fig. 1. Research Strategies (Adapted from Runkel and McGrath 1972)

analyzed post hoc, and the developers did not know during development that their work was going to be assessed." In other words, the data were available and analyzed and compared after it had taken place, without any intervention by the researchers.

Field studies achieve a 'maximum' in realism of context (indicated by the symbol C in Fig. 1). However, this comes at a cost of having a low level of precision of measurement or control as would be found in laboratory experiments. Also, the level of generalizability to a large population is much lower (due to the specific context) than what one would have found in sample surveys.

B. Field Experiments

Field experiments is the general term for those studies conducted in the field in which the researcher manipulates some properties or variables in the environment. Example research methods within this strategy are *in-vivo* experiments (to contrast them with *in-vitro* experiments, which fall under the laboratory experiment strategy), and action research. In this strategy, elements of the research setting are changed by the researcher, which could happen for any reason, depending on the study's goal. For instance, in a controlled experiment, a researcher's goal is to investigate a causal relationship between a number of variables. On the other hand, action research aims to intervene and improve a specific setting through a cycle of making changes, observing the new situation and making further changes.

An example of a field experiment in software engineering is a study by Anda et al. in which four companies were selected to implement the same software system [33]. The systems were implemented by real companies (a natural setting), and the researcher's manipulation here is that of specifying the system.

C. Experimental Simulations

In an Experimental Simulation the activities (*behavior*) that take place are natural, but the setting within which these occur has been created for the purposes of the study only. Thus, the term 'simulation' in this context refers to the artificially created setting in which the 'experiment' or the observed behavior takes place.

One recent example of an experimental simulation is a study on design competitions in the context of crowdsourcing software development [34]. The design competitions were artificially created specifically for the study—the context was therefore contrived and not what one would find in a real crowdsourcing setting (as found in an exploratory industry case study [35]—that study was a field study as defined above).

D. Laboratory Experiments

Laboratory Experiments differ from Field Experiments (see Sec. III-B above) in that the former aims to investigate a contrived setting whereas the latter investigates a natural context and setting. This absence of extraneous conditions [29, p.105] can be enforced in a laboratory setting, but clearly not in a real-world environment. Therefore, a Laboratory Experiment allows a researcher to exercise a much greater level of precision of measurement and control than would have been possible in a real world setting.

An example of a Laboratory Experiment is a study on pair programming [36]. 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; none of the 295 participants would have done these tasks without explicitly being requested (and paid). The change tasks themselves were also carefully designed as part of the study.

Laboratory experiments offer a researcher the option to exert a maximum level of precision of measurement of behavior (indicated by the symbol B in Fig. 1). However, this comes at the 'cost' of having a very low level of realism of context. Furthermore, the level of generalizability to a large population of actors is also limited when compared to sample surveys, for instance.

E. Judgment Tasks

Judgment tasks is one of two strategies (Sample Surveys being the other) for gathering observations that are independent of a specific setting (a point we return to later). Judgment tasks are similar to sample surveys, but differ in two key ways. First, in Judgment Tasks, participants are asked to judge or rate behaviors or discuss a topic of interest. Second, sampling is *systematic* rather than *representative* [37] whereas Sample Surveys intend to be representative for a certain population.

One example of a study following this strategy was a Delphi study [38] with a panel of experts from both industry and academe [39]. The study investigated the characteristics for effective tailoring of agile methods. Indeed, the selection of participants in this study was systematic rather than representative. Rather than attempting to get a representative sample, the researchers selected a total of 36 key experts.

F. Sample Surveys

Sample surveys aim to achieve generalizability over a certain population (*actors*), for instance, project managers, software developers, or end-users of software. The sample survey is one of the most common strategies in software engineering research (see also Table I).

A recent example is a survey on knowledge needs for program comprehension [40], which was implemented using an online survey tool. The survey consisted of 19 fixed-choice questions, and took (according to the authors) *"about 15 minutes"* to complete.

A sample survey could, of course, also be conducted on software systems rather than human participants. Researchers can mine software repositories (often OSS repositories such as GitHub as these provide freely and easily accessible data) and collect data of large samples of software projects. This allows researchers to study questions such as, *"What is the effect of programming languages on software quality?"* [41].

In sample surveys the level of generalizability to a large population of actors (systems or software professionals) is high,

given a large enough sample. However, the level of precision of measurement of behavior is low in sample surveys (due to the unobtrusive nature of sample survey research), as is the realism of context, a point that we return to in Section IV.

G. Formal Theory

Formal Theory is one of two strategies in which no empirical[2] observations are made (the other being Computer Simulations, discussed below). Developing formal theory is a strategy that aims at a high level of universality, and seeks to maximize generalizability over populations (actors). However, theory development in such an isolated manner tends to be low on realism of context and precision of measurement of behavior.

An example of Formal Theory is reported in an article by El Kholy et al. [42], who developed the temporal logic CTL^{cc} that extends Computation Tree Logic (CTL). This extension allows the representation of, and reasoning about, communicating 'commitments' when building a model. This extended formal notation does not focus on a particular system (thus it can be used in a more universal setting). (The article also reports applications in a number of case studies (in this case the term means 'worked example'); however, as pointed out earlier, papers can report on multiple studies. The example of Formal Theory is here the derivation of the CTL^{cc}).

Similar to Sample Surveys, Formal Theory can achieve a high level of generalizability. The goal of a general theory aims to be applicable to any situation (within boundaries), although empirical evidence must of course be gathered to test the theory, for which other strategies are needed.

H. Computer Simulations

A computer simulation attempts to create a symbolic replica of a certain type of concrete systems [29, p.87]. In a computer simulation, everything is represented symbolically and created artificially. Where studies in natural settings (field studies, field experiments) can be costly to conduct as it involves a higher level of participation from participants, simulations *"are like virtual laboratories where hypotheses about observed problems can be tested,"* before they are implemented in real-world systems [43].

One recent example of a computer simulation is a study that simulated adaptive security in buildings [22]. Before actually implementing such a system in the real world (which involves making many changes to the physical building), the authors performed a simulation to proof the concept. By analyzing a number of 'threat scenarios,' the researchers could evaluate the applicability and effectiveness of the approach as well as evaluate the classes of security requirements that could be handled by such a system.

IV. Comparing Research Strategies

The research strategies outlined above are positioned along two dimensions: the level of obtrusiveness and the level of generality. The level of obtrusiveness refers to the level of 'intrusion' into the research setting. The level of generality refers to whether a study applies to a particular behavior or system, or whether it aims to understand 'universal' behavior (e.g., of software professionals or software systems).

The settings in which the research strategies are adopted vary, and Runkel and McGrath identified four different types of research settings; these are indicated by the roman numbers I to IV in Fig. 1. By distinguishing between these four research settings we can compare the different research strategies. Our presentation discusses the research settings in the order of Roman-numbered 'quadrants' in Fig. 1.

A. Natural Settings

Field Studies are conducted in *natural* settings as they occur in the real world, and a researcher's 'intrusion' into that setting only serves the purpose of making observations and gathering data. The researcher in this setting has no goal to make any changes or "control" any variables of interest. For instance, in their ethnographic study of XP, Sharp and Robinson made it quite clear that their aim was to understand the community that uses an agile method [31]. Likewise, Field Experiments are also conducted in natural settings (for instance, industrial settings) and have limited intrusion, but more so than in a Field Study, since the researcher controls or manipulates some variables or properties so as to be able to measure any changes. This was also the case in the study by Anda et al. [33] in which four different companies implemented the same system.

B. Contrived Settings

The next two strategies, Experimental Simulations and Laboratory Experiments, are both used in *contrived* settings that are artificially created by a researcher for the sole purpose of the study (quadrant II in Fig. 1.) Therefore, they are positioned at the top of the 'obtrusiveness' dimension as the researcher has a great deal of control over the properties and variables in the study. In an Experimental Simulation, a researcher has even more control over the behavior than in a Field Experiment, because in the former the researcher also attempts to create a setting that mirrors a class of naturally occurring systems [29, p. 96]. This increased level of control due to the increased intrusion by the researcher comes at the cost of decreased context; as more aspects of a study are controlled by a researcher, the context-specific details are stripped out of the study setting. For instance, LaToza et al. [34] created the crowdsourcing design competitions, and as such this was a contrived setting, but it was in the researchers' interest to make the setting look similar to one that could be find in a natural setting.

Laboratory Experiments further increase the level of control a researcher has on the behavior setting to its maximum. This happens at the cost of realism, but the goal of this strategy is not to resemble a natural setting or phenomenon, but to isolate the processes and variables of interest. While a Laboratory Experiment aims to gather 'universal' insights rather than insights from a particular system through the

[2]By 'empirical' we mean observations in the real world—not observations made by means of a computer simulation.

TABLE II
RESEARCH STRATEGIES AND EXEMPLAR STUDIES IN SOFTWARE ENGINEERING RESEARCH

Strategy	Description	Reference
Field Studies	An ethnographic study of Extreme Programming (XP).	Sharp & Robinson in Empir. Software Eng. [31]
Field Experiments	A study in which four companies develop the same system.	Anda et al. in IEEE Trans. Softw. Eng. [33]
Experimental Simulations	A study of crowdsourcing design competitions.	LaToza et al. in ICSE'15 [34]
Laboratory Experiments	An evaluation of the Pair Programming technique in a laboratory setting, involving 295 participants hired for the study's purpose.	Arisholm et al. in IEEE Trans. Softw. Eng. [36]
Judgment tasks	A Delphi study to investigate method and developer characteristics that make agile methods amenable to tailoring.	Conboy & Fitzgerald in ACM Trans. Softw. Eng. Methodol. [39]
Sample Surveys	Study on knowledge needs for program comprehension; online survey with 19 fixed-answer questions and 1,477 respondents.	Maleej et al. in ACM Trans. Softw. Eng. Methodol. [40]
Formal Theory	Developent of CTL^{cc}, an extension of Computation Tree Logic (CTL) that allows the representation and reasoning of conditional commitments in a formal language.	El Kholy et al. in ACM Trans. Softw. Eng. Methodol. [42]
Computer Simulations	Study simulating adaptive security for buildings. Rather than implementing it in the physical world, the researchers performed a simulation to evaluate the effectiveness and assess which types of threats could be handled.	Tsigkanos et al. in Requirements Engineering '14 conference [22]

precise measurement and control of dependent and independent variables. Thus, while this strategy therefore must have a high internal validity, the external validity in such studies is arguably less as the behavior (or effects or otherwise measured in such studies) occur in *contrived* settings, not real ones.

C. Setting-Independent

The two research strategies in quadrant III are Judgment Tasks and Sample Surveys. These strategies aim to gather observations of behavior (people, systems) that is independent from the setting. Thus, these strategies are *setting-independent*. Both strategies involve the elicitation of opinions or beliefs regarding the topic of study, and both aim to gather observations that have a more universal applicability rather than to a particular system (hence, their independence of the setting).

Judgment Tasks are used to elicit responses from a set experts, or *judges* about a certain topic, stimulated by the researcher. The 'stimuli' provided by the researcher are carefully prepared by a researcher, which represents the higher level of 'control' when compared to Sample Surveys. Given the usually limited set of respondents, Judgment studies have less generalizability over the population (actors). The example of a Judgment Task study discussed above was a Delphi study [39]. The 'stimuli' in this case were characteristics that had been derived from the literature on agile method tailoring, and the experts on the Delphi panel were interviewed in a number of rounds to provide feedback.

Sample Surveys differ from Judgment Tasks in that they aim to gather data from a larger group of respondents and as such usually have a better generalizability (given a large enough sample). The set of questions is typically limited.

D. No Empirical Setting

The last two strategies are not empirical strategies but rather are theoretical. Both strategies are useful to explore topics of interest prior to conducting empirical research and as shown in Table II both are used and published in some of the top-tier publication outlets in our field. As these strategies do not represent *empirical* approaches, there is no empirical setting in which actors (software professionals or software systems) or their behavior or properties are observed.

Formal theories aim to model a generic class of behavior (to address questions such as: What motivates of software engineers?) or systems (to address questions such as: How does OSS development work?) [29]. Theories always have boundaries outside of which they do not apply [44], but like Sample Surveys the aim is to generalize to a certain population of 'actors' (e.g., software developers, OSS).

A Computer Simulation is a symbolic representation of some concrete type of system [29]. A researcher can model exactly the parameters, properties and relationships between them that are of interest, which also means that they are predetermined (programmed, if you will). Because the model in a computer simulation must operationalize variables, properties and constructs, a computer simulation represents a more specific 'system' than a formal theory does, which can be formulated without operationalizing its constructs. For instance, to model the adaptive security, Tsigkanos et al. had to model and implement the various 'threats' in computer code so that the simulation could be run. No such specification was needed in the development of CTL^{cc}, which also makes the latter more universally applicable.

V. DISCUSSION AND CONCLUSION

A. Implications for Software Engineering Research

In this paper we have drawn from insights from the social sciences by using a two-dimensional framework to position a set of eight different research strategies. The framework does not claim to be complete, in that there may be other research strategies. As we have argued, there is a strong emphasis

TABLE III
STRENGTHS AND WEAKNESSES OF THE DIFFERENT RESEARCH STRATEGIES

No.	Setting	Strategy	Strengths/Weaknesses
I	Natural Setting	Field Study	+ Maximizes realism of context – Low on precision of measurement – Low on generalizability of results
		Field Experiment	+ Allows a researcher to test hypotheses in real-world settings with high level of realism – Low level of generalizability to populations
II	Contrived Setting	Experimental Simulation	+ Offers more control over measurement of behavior and processes ('greenhouse' setting [6]) – Low level of generalizability to populations
		Laboratory Experiment	+ Maximizes control and precision of measurement – No realistic context
III	Setting-independent	Judgment Tasks	+ Offers a limited level of control to steer the topic of study; compromise between level of control/precision of measurement and generalizability to large population – Observations are independent of specific context
		Sample Surveys	+ Maximizes generalizability to populations – No realistic context
IV	No Setting	Formal Theory	+ Maximizes generalizability to settings (within the boundaries of the theory) – Limited value without empirical validation or application
		Computer Simulations	+ Allows researcher to model a potentially expensive setting rather than implementing it in real life – Does not help in gathering new empirical data

in SE research on the three best known strategies, namely sample surveys (as opposed to literature surveys, although systematic reviews are widespread), laboratory experiments and field studies. As the framework shows, other strategies are available, and indeed, these can be observed in the SE literature (see Table II). However, there is no common terminology in use for the various strategies. The framework presented in this paper offers an alternative set of terms which we believe will help in identifying the research strategies used in papers. This is highly useful in the analysis phase of systematic literature reviews. In most SLRs, the researchers report that they merely reported the methods as reported by the authors of the original study (we have done this ourselves [45]). One problem with this is, of course, that given the disagreement and inconsistency in terminology observed above, the results from systematic reviews will suffer from the same inconsistencies.

An important consequence of the observations with respect to the strengths and weaknesses of different research strategies is that the SE research community needs better guidelines to more fairly evaluate the quality of research studies. When evaluating a Field Study, a referee should not lament the supposedly "lack of control" in the study, because that is indeed its admitted weakness. Rather, such studies with a high degree of realism should be assessed on those strengths. Likewise, it would be unfair to criticize a Laboratory Experiment because of its lack of a realistic context; rather it should be assessed according to the extent to which it has achieved an appropriate degree of control of variables (construct validity, internal validity). This is not to say that calls for more realism such as by Sjøberg et al. [8] should be ignored, but rather that both researchers and referees should acknowledge and accept the inherent limitations of the adopted research strategy. We believe that studies should be evaluated based on the extent to which its *potential* strengths have been realized and its inherent limitations have been minimized. To achieve this, the SE research community need a common understanding of the different research strategies and how they relate; in this paper we have attempted to contribute to this goal.

The dichotomy of 'rigor' versus 'relevance' is a false one. Clearly, some research strategies, such as laboratory experiments, allow a researcher to enforce a higher level of 'control' in terms of the research design and measuring behavior. Others, such as field studies, take place in a realistic setting, and by necessity leave less to be controlled by a researcher. This is indicated by the three symbols A, B, and C in Fig. 1.

B. Conclusion

Existing guidelines offer various and limited sets of different research approaches, which are of varying levels of granularity (case study v. exploratory/evaluative case study; experiment v. controlled/quasi experiment, etc.), and thus represent a "mixed bag" of methods. We have adopted a framework from the social sciences, which provides a holistic overview of different research strategies. While existing overviews provide overviews of different methods, they do not systematically explain how these methods relate to one another. The framework in Fig. 1 positions a set of eight different research strategies along two dimensions: the level of 'obtrusiveness' (i.e., control by the researcher), and the level of generalizability. We illustrate each of these strategies with examples from the software engineering literature which have been published in the field's top outlets—an indication that these strategies are already prominent, merely lacking a common terminology. These strategies can be considered to be the architectures of research designs, which have associated weaknesses and strengths due to their relation with the level of the researcher's 'control' (the obtrusiveness dimension) and the level of generalizability

to populations (the universal v. particular system dimension). Rather than speaking of an "interview study" or a "case study," which does not convey any information regarding the generalizability or level of control that a researcher might have, we believe that this set of research strategies can offer guidance to researchers in the software engineering community in making an appropriate trade-off.

ACKNOWLEDGMENT

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 to Lero—the Irish Software Research Centre (www.lero.ie), the Irish Research Council under the New Foundations Programme, the Royal Irish Academy under the Charlemont Award Programme, and Enterprise Ireland grant IR/2013/0021 to ITEA2-SCALARE (www.scalare.org).

REFERENCES

[1] D. E. Perry, A. E. Porter, and L. G. Votta, "Empirical studies of software engineering: A roadmap," in *Future of Software Engineering*, 2000.

[2] B. A. Kitchenham, S. L. Pfleeger, L. M. P. Pickard, P. W. Jones, D. C. Hoaglin, K. El Emam, and J. Rosenberg, "Preliminary guidelines for empirical research in software engineering," *IEEE Trans. Softw. Eng.*, vol. 28, no. 2, pp. 721–734, 2008.

[3] V. R. Basili and M. V. Zelkowitz, "Empirical studies to build a science of computer science," *Commun. ACM*, vol. 50, no. 11, pp. 33–37, 2007.

[4] R. L. Glass, I. Vessey, and V. Ramesh, "Research in software engineering: an analysis of the literature," *Inform. Software Tech.*, vol. 44, 2002.

[5] T. Dybå, R. Prikladnicki, K. Rönkkö, C. Seaman, and J. Sillito, "Qualitative research in software engineering," *Empir. Software Eng.*, vol. 16, pp. 425–429, 2011.

[6] J. E. McGrath, "Dilemmatics: The study of research choices and dilemmas," *Am. Behav. Sci.*, vol. 25, no. 2, pp. 179–210, 1981.

[7] L. Bass, P. Clements, and R. Kazman, *Software Architecture in Practice*, 2nd ed. Addison-Wesley Professional, 2003.

[8] D. I. Sjøberg, B. Anda, E. Arisholm, T. Dybå, M. Jørgensen, A. Karahasanovic, E. F. Koren, and M. Vokac, "Conducting realistic experiments in software engineering," in *Proc. International Symposium on Empirical Software Engineering (ISESE02)*, 2002.

[9] S. Easterbrook, J. Singer, M.-A. Storey, and D. Damian, "Selecting empirical methods for software engineering research," in *Guide to Advanced Software Engineering*, F. Shull, J. Singer, and D. I. Sjøberg, Eds., 2008.

[10] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empir. Software Eng.*, vol. 14, pp. 131–164, 2009.

[11] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*. Springer, 2012.

[12] B. Kitchenham, T. Dybå, and M. Jørgensen, "Evidence-based software engineering," in *Proc. International Conf. Software Engineering*, 2004.

[13] A. C. Edmondson and S. E. McManus, "Methodological fit in management field research," *Acad. Manage. Rev.*, vol. 32, no. 4, 2007.

[14] C. Marshall and G. Rossman, *Designing Qualitative Research*, 3rd ed. SAGE Publications, 1999.

[15] M. Shaw, "Writing good software engineering research papers," in *Proc. 25th International Conf. Software Engineering*, 2003, pp. 726–736.

[16] F. Shull, J. Singer, and D. I. Sjøberg, Eds., *Guide to Advanced Empirical Software Engineering*. Springer, 2008.

[17] J. Singer, M.-A. Storey, and S. E. Sim, "Beg, borrow, or steal: Using multidisciplinary approaches in empirical software engineering research," in *Proc. International Conf. Software Engineering*, 2000.

[18] C. Wohlin and A. Aurum, "Towards a decision-making structure for selecting a research design in empirical software engineering," *Empir. Software Eng.*, vol. in press, 2014.

[19] B. Fitzgerald and D. Howcroft, "Towards dissolution of the IS research debate: from polarization to polarity," *Journal of Information Technology*, vol. 13, no. 4, pp. 313–326, 1998.

[20] H. Jordan, S. Beecham, and G. Botterweck, "Modelling software engineering research with rsml," in *Proc. 18th International Conference on Evaluation and Assessment in Software Engineering*, 2014.

[21] A. Mockus, R. Fielding, and J. Herbsleb, "Two case studies of open source software development: Apache and mozilla," *ACM Trans. Softw. Eng. Methodol.*, vol. 11, no. 3, 2002.

[22] C. Tsigkanos, L. Pasquale, C. Menghi, C. Ghezzi, and B. Nuseibeh, "Engineering topology aware adaptive security: Preventing requirements violations at runtime," in *Proc. 22nd IEEE International Requirements Engineering Conference*, 2014, pp. 203–212.

[23] D. I. Sjøberg, J. E. Hannay, O. Hansen, V. B. Kampenes, A. Karahansanovic, N.-K. Liborg, and A. C. Rekdal, "A survey of controlled experiments in software engineering," *IEEE Trans. Softw. Eng.*, vol. 31, no. 9, 2005.

[24] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Trans. Softw. Eng.*, vol. 24, no. 4, pp. 557–572, 1999.

[25] T. C. Lethbridge, S. E. Sim, and J. Singer, "Studying software engineers: Data collection techniques for software field studies," *Empir. Software Eng.*, vol. 10, pp. 311–341, 2005.

[26] M. V. Zelkowitz, "Techniques for empirical validation," *Empirical Software Engineering Issues*, vol. LNCS 4336, pp. 4–9, 2007.

[27] C. Zannier, G. Melnik, and F. Maurer, "On the success of empirical studies in the international conference on software engineering," in *Proc. International Conf. Software Engineering*, 2006, pp. 341–350.

[28] P. Tonella, M. Torchiano, B. Du Bois, and T. Systä, "Empirical studies in reverse engineering: state of the art and future trends," *Empir. Software Eng.*, vol. 12, pp. 551–571, 2007.

[29] P. J. Runkel and J. E. McGrath, *Research on Human Behavior: A Systematic Guide to Method*. Holt, Rinehart and Winston, Inc., 1972.

[30] A. Mockus, R. Fielding, and J. Herbsleb, "A case study of open source software development: the Apache server," in *Proc. International Conf. Software Engineering*, 2000.

[31] H. Sharp and H. Robinson, "An ethnographic study of xp practice," *Empir. Software. Eng.*, vol. 9, no. 4, pp. 353–375, 2004.

[32] N. Nagappan, E. M. Maximilien, T. Bhat, and L. Williams, "Realizing quality improvement through test driven development: results and experiences of four industrial teams," *Empir. Software Eng.*, vol. 13, pp. 289–302, 2008.

[33] B.C.D. Anda, D. Sjøberg, and A. Mockus, "Variability and reproducibility in software engineering: A study of four companies that developed the same system," *IEEE Trans. Softw. Eng.*, vol. 35, no. 3, 2009.

[34] T. D. LaToza, M. Chen, L. Jiang, M. Zhao, and A. van der Hoek, "Borrowing from the crowd: A study of recombination in software design competitions," in *Proc. International Conf. Software Engineering*, 2015.

[35] K. Stol and B. Fitzgerald, "Two's company, three's a crowd: A case study of crowdsourcing software development," in *Proc. 36th International Conference on Software Engineering*, 2014, pp. 187–198.

[36] E. Arisholm, H. Gallis, T. Dybå, and D. I. Sjøberg, "Evaluating pair programming with respect to system complexity and programmer expertise," *IEEE Trans. Softw. Eng.*, vol. 33, no. 2, pp. 65–86, 2007.

[37] T. A. Scandura and E. A. Williams, "Research methodology in management: Current practices, trends, and implications for future research," *Acad. Manage. J.*, vol. 43, no. 6, pp. 1248–1264, 2000.

[38] H. A. Linstone and M. Turoff, Eds., *The Delphi Method Techniques and Applications*. Addison-Wesley, 2002.

[39] K. Conboy and B. Fitzgerald, "Method and developer characteristics for effective agile method tailoring: A study of xp expert opinion," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 1, 2010.

[40] W. Maleej, R. Tiarks, T. Roehm, and R. Koschke, "On the comprehension of program comprehension," *ACM Trans. Softw. Eng. Methodol.*, vol. 23, no. 4, 2014.

[41] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proc. 22nd ACM SIGSOFT International Sym. Foundations of Software Engineering*, 2014.

[42] W. El Kholy, J. Bentahar, M. El Menshawy, H. Qu, and R. Dssouli, "Conditional commitments: Reasoning and model checking," *ACM Trans. Softw. Eng. Methodol.*, vol. 24, no. 2, 2014.

[43] M. Müller and D. Pfahl, "Simulation methods," in *Guide to Advanced Software Engineering*, F. Shull, J. Singer, and D. I. Sjøberg, Eds., 2008.

[44] K. Stol and B. Fitzgerald, "Theory-oriented software engineering," *Science of Computer Programming*, vol. 101, pp. 79–98, 2015.

[45] K. Stol, M. A. Babar, B. Russo, and B. Fitzgerald, "The use of empirical methods in open source software research: Facts, trends and future directions," in *Proc. 2nd International Workshop on Emerging Trends on FLOSS Research*, 2009.