The Who, What, How of Software Engineering Research: A Socio-Technical Framework

Margaret-Anne Storey · Neil A. Ernst ·
Courtney Williams · Eirini Kalliamvakou

Abstract Software engineering is a socio-technical endeavor, and while many of our contributions focus on technical aspects, human stakeholders such as software developers are directly affected by and can benefit from our research and tool innovations. In this paper, we question how much of our research addresses human and social issues, and explore how much we study human and social aspects in our research designs. To answer these questions, we developed a socio-technical research framework to capture the main beneficiary of a research study (the *who*), the main type of research contribution produced (the *what*), and the research strategies used in the study (*how* we methodologically approach delivering relevant results given the *who* and *what* of our studies). We used this Who-What-How framework to analyze 151 papers from two well-cited publishing venues—the main technical track at the International Conference on Software Engineering, and the Empirical Software Engineering Journal by Springer—to assess how much this published research explicitly considers human aspects. We find that although a majority of these papers claim the contained research should benefit human stakeholders, most focus on technical contributions without engaging humans in their studies.

Although our analysis is scoped to two venues, our results suggest a need for more diversification and triangulation of research strategies. In particular, there is a need for strategies that aim at a deeper understanding of human and social aspects of software development practice to balance the design and evaluation of technical innovations. We recommend that the framework should be used in the design of future studies in order to nudge software engineering research towards explicitly including human and social concerns in their designs, and to improve the relevance of our research for human stakeholders.

Keywords Empirical methods · Human studies · Software engineering, Meta-research · Survey

Margaret-Anne Storey · Neil A. Ernst · Courtney Williams · Eirini Kalliamvakou
University of Victoria, Canada
E-mail: mstorey@uvic.ca, nernst@uvic.ca, courtneyelwilliams@gmail.com, ikaliam@uvic.ca

1 Introduction

Nowadays we recognize software engineering as a socio-technical endeavor [42], and we increasingly see social aspects as a critical part of the software engineering practice and research landscape [14]. What is more, while we may expect that many of our contributions are purely technical, somewhere, at some time, a software developer may be affected by our work. It is crucial to account for the social aspects of software engineering in our research, and we know that to capture them, we need appropriate driving research questions and methods, as well as a focus on relevant stakeholders [31]. In this paper, we ask if and how we are making these provisions in our empirical studies.

The focus of our investigation is how software engineering research approaches the inclusion and study of social aspects in software development. This led us to articulate questions about *who* our research intends to benefit, *what* are our research contributions, and *how* methodologically we approach delivering relevant results given the *who* and *what* of our studies.

We analyzed papers from two well-cited publishing venues to assess how much empirical software engineering research may explicitly consider or study social aspects. We considered a cohort of papers (from 2017) published in the main technical track at the International Conference on Software Engineering (ICSE) and in the Empirical Software Engineering journal by Springer (EMSE). For these papers, we aimed to answer the following questions:

- **RQ1:** Who are the **beneficiaries** (technical systems, human stakeholders, researchers) of the research contributions? Note that for papers that do not consider human stakeholders in their research goals, we would not expect the paper to directly study or address human and social aspects.
- **RQ2:** What is the main type of **research contribution** (descriptive or solution) provided? Descriptive papers add new or refute existing knowledge about a software engineering problem or context. Solution papers present the design and/or evaluation of a new intervention (i.e., a process or tool).
- **RQ3:** Which **research strategies** are used? Some research strategies innately involve human subjects to collect or generate data, while others may rely on collecting previously archived, tool generated or simulated data.
- **RQ4:** How do the reported research strategies **map** to the beneficiary and types of contributions in these papers?

To answer our four research questions, we developed a socio-technical research framework to capture the main beneficiary of the research, the main type of research contribution, and the research strategies used. We refer to this framework as the Who-What-How framework. We find that the majority of the 151 papers published in both venues in 2017 (ICSE and the EMSE journal) present research that the authors claim should **benefit human stakeholders** at some point in time, but most of these do not use **research strategies** that directly involve human participants. In terms of the **types of contributions**, we find that the majority of papers published at ICSE 2017 presented solutions (mostly technical) to address a software engineering problem, while

the majority of papers published in the EMSE journal in 2017 are descriptive contributions that present insights about software engineering problems or how technical solutions are used. We conclude by calling for more diversification and triangulation of research strategies so that we may gain a deeper understanding of human and social aspects of software development practice to balance the current focus on the design and evaluation of technical innovations.

The remainder of our paper is structured as follows. We discuss related work that has both informed and motivated this research in Section 2. In Section 3, we introduce the Who-What-How framework we designed for categorizing the beneficiaries, research contribution and research strategies in the papers we studied. In Section 4, we present the methodology we followed to answer our research questions. In Section 5, we present the results from the four research questions we posed. We then interpret these results in Section 6, discussing possible explanations and the implications of our findings. We describe the limitations of our research in Section 7. Finally, we conclude the paper by identifying areas for future work and list important takeaways in Section 8. Traceability artifacts from our analysis and a replication package are at DOI: 10.5281/zenodo.3813878.

2 Background

The importance of social aspects in software engineering was recognized long ago [4, 7, 35, 41]. Typically there is at least one track on social aspects in the main research conferences, as well as special purpose workshops on the topic, such as the CHASE series[1]. The papers presented at CHASE tend to address broad socio-technical topics, but the workshop focuses on early results. The Empirical Software Engineering and Measurement (ESEM) conference and the Empirical Software Engineering (EMSE) journal also attract papers that consider social aspects as their focus is on empirical methods, many of which directly involve human participants. Some special journal issues have also addressed human aspects in software engineering using qualitative methods [9, 10]. Still, the debate about whether we study social aspects enough is ongoing and some researchers claim that coverage of these aspects is lacking [23].

Beyond the discussion of *how much* we study social aspects as part of software engineering research, is the discussion of *how* we approach them methodologically. Researchers have focused on discussing specific methods (e.g., focus groups [20], personal opinion surveys [19], or data collection techniques for field studies [36]) and providing guidelines on how to use them, or explaining the benefits and drawbacks of methods to assist in research design choices [11]. Seaman [31, 32] highlighted that a study focusing on social aspects asks different questions from one focusing on technical aspects, and needs to use appropriate

---

[1] Cooperative and Human Aspects of Software Engineering, co-located with ICSE since 2008 http://www.chaseresearch.org/

methods that capture firsthand behaviors and information that might not be noticed otherwise. She discussed ways to incorporate qualitative methods into empirical studies in software engineering.

Social aspects can be approached methodologically by inferring behaviour from analyzing trace data of developers' past activities (e.g., code commits, code review comments, posted questions and answers on developer forums, etc.). But the analysis of trace data alone is fraught with threats to validity as it shows an incomplete picture of human behaviour, intent and social interactions in software engineering [1, 17]. Furthermore, trace data alone cannot be used to predict how a new solution may perturb an existing process in industry settings [21], although relying on trace data can bring early insights about the feasibility of a solution design. To appropriately capture and account for social aspects in software engineering research, we need to use dedicated methods that directly involve human participants in our empirical studies.

Method choice in empirical software engineering studies has been an item of reflection, especially around methods that are borrowed from other domains. Zelkowitz [44] reported that researchers were using terms such as "case study" to refer to different levels of abstraction, making it hard to understand the communicated research. However, as we are starting to recognize the misuse of certain methods, the research community is coming up with guidelines on how to use them correctly. Stol, Ralph, and Fitzgerald produced guidelines for grounded theory in the context of software engineering [62] as they found that many papers reporting the use of grounded theory lacked rigor. Runeson and Høst adapted case study research guidelines to the software engineering domain [29], also in part to address the misuse of the term "case study" in empirical software engineering studies. Sharp *et al.* [33] advocate using **ethnography** in software engineering studies as a way to capture rich insights about what developers and other stakeholders do in practice, *why* they follow certain processes, or *how* they use certain tools.

In this paper, we analyze published research in software engineering to investigate how social aspects are accounted for and studied. However, there have been previous efforts to reflect on and reexamine software engineering research to better understand where the community places value. For example, Shaw [34] analyzed the content of both the accepted and rejected papers of ICSE 2002, as well as observed program committee conversations about which papers to accept. She found low submission and acceptance rates of papers that investigated "categorization" or "exploration" research questions, and low acceptance of papers where the research results presented were "qualitative or descriptive models". A 2016 replication of Shaw's methodology [40] drew similar conclusions and identified a new category of papers, mining software repositories, was becoming common. While the earlier efforts focused on categorizing research and empirical studies in software engineering, the work we report in this paper focuses specifically on how studies approach social aspects and discusses the trade-offs and implications for the software engineering community's collective knowledge on the choices made in the studies we ex-

amined. In the next section of this paper, we present a framework specifically designed for this purpose.

3 A Socio-Technical Research Framework: The Who, What, How of Software Engineering Research

To answer our research questions and guide our analysis of how empirical software engineering research papers may address human and social aspects in software engineering, we developed a socio-technical research framework to capture the main beneficiary of the research (the *who*), the main type of research contribution in each paper (the *what*), and the research strategies used (the *how*). The shape of our framework and the questions it poses emerged from several early iterations we followed when we tried to compare how papers address social and human aspects in software engineering research. Accordingly, our Who-What-How framework has three main parts, as shown in Fig. 1 and described below.

3.1 *Who* Is the Research Beneficiary?

The first part of the framework (see top of Fig. 1) considers *who* are the main beneficiaries of the research contributions claimed in a paper. All research papers intend for someone, or something, to be a primary recipient of the improvement or insight proposed by the research contribution, assuming that contribution is valid and practically relevant. We consider the following possibilities for the beneficiary in our framework:

- **Human stakeholders**[2], which may include software developers, architects, analysts, managers, end users and the social organizations they form;
- **Technical systems**, which may include tools, frameworks and platforms being used to support development[3]; and
- **Researchers**, which may include software engineering academics or industry research and internal tools teams.

A paper's research contribution may be aimed at multiple beneficiaries. For example, a paper may provide insights for researchers to consider in their future research studies, while at the same time make recommendations for practitioners to consider. Likewise, a paper may provide insights that improve a technical system, such as improving the accuracy of a bug detection tool, but at the same time provide cognitive support to the developer who will use the tool. Alternatively, some papers may be clearly aimed at a single beneficiary. For example, we intend for the contributions of this paper to benefit researchers.

---

[2] We shorten this to "Humans" in the rest of the paper.

[3] We recognize that most technical systems are studied or improved with the final goal to benefit a human stakeholder. However, we found in many papers that these human stakeholders are not discussed and that the research is aimed at understanding or improving the technical system.
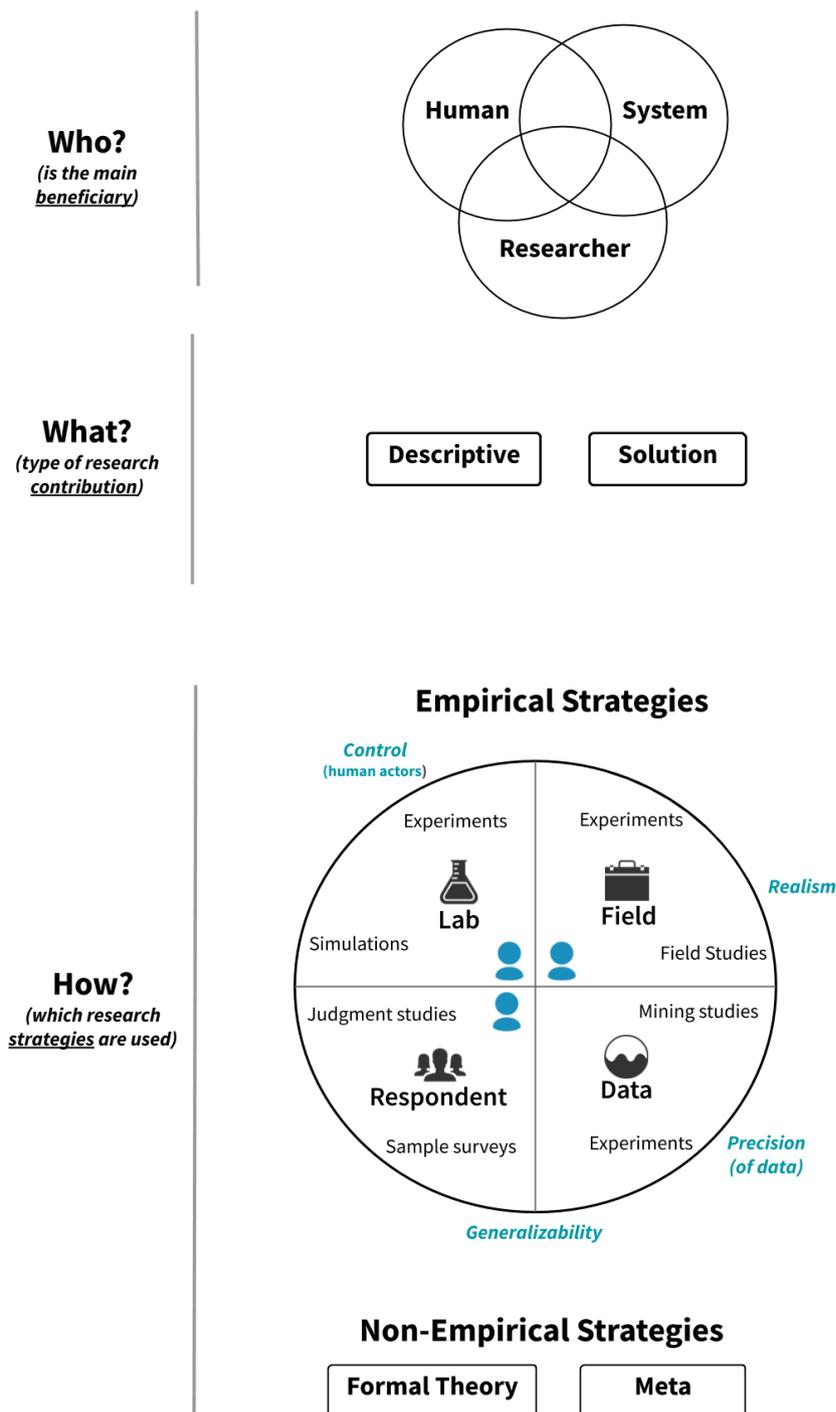
Who?
(is the main beneficiary)

Human   System   Researcher

What?
(type of research contribution)

Descriptive   Solution

How?
(which research strategies are used)

Empirical Strategies

Control
(human actors)

Experiments   Experiments

Realism

Lab   Field

Simulations   Field Studies

Judgment studies   Mining studies

Respondent   Data

Sample surveys   Experiments

Precision
(of data)

Generalizability

Non-Empirical Strategies

Formal Theory   Meta

Fig. 1: Our research strategy framework for categorizing the who (beneficiary), what (type of contribution) and how (research strategy) of empirical software engineering research. For the circumplex (circle) of empirical research strategies, note that in addition to labeling each quadrant with the type of strategy (Lab, Field, Respondent, and Data), we also show the main representative strategies in each quadrant (e.g., Experiment, Field Studies in the Field quadrant). We show **research quality criteria** (Control, Realism, Precision, and Generalizability) on the outside of the circumplex, positioned closer to the research strategies that have the highest *potential* to increase those criteria in particular.

3.2 *What* Is the Main Research Contribution Type?

The second part of the framework (see middle of Fig. 1) captures *what* research contribution a paper claims. To characterize the type of research contribution, we turn to a design science lens developed in previous work [12]. Design science is a paradigm for conducting and communicating applied research, such as software engineering. Similar to other design sciences, much software engineering research aims to understand the nature of problems and real-world design contexts, that is **descriptive knowledge**, and/or produce prescriptive knowledge to guide the design or improvement of **solutions** that address engineering problems. Although some papers can have both a descriptive and a solution (prescriptive) contribution, we categorize papers according to a single main contribution, as emphasized by the authors of the paper.

3.3 *How* Is the Research Conducted?

The last part of the framework (see bottom of Fig. 1) helps us articulate *how* the research was conducted by capturing the research strategies used. This part of the framework is derived in part from Runkel and McGrath's model of research strategies [25, 30]. We describe the original Runkel and McGrath research strategy model and how we adapted it in the appendix of our paper (Appendix A). A recent paper by Stol and Fitzgerald [39] also uses the Runkel and McGrath circumplex to provide consistent terminology for research strategies, which we also discuss in Appendix A.

The *how* part of our framework first distinguishes empirical from non-empirical research. Non-empirical research papers are not based directly on data (human or system generated) [15]. Some non-empirical papers present literature reviews or meta-analyses of previous studies or empirically collected data, or they may describe research that uses formal methods, provides proofs or generates theories from existing theories. As we show later, the vast majority of papers published at ICSE and in the EMSE journal report or collect empirical data (some in addition to meta and/or formal theory research). This is not surprising as the EMSE journal in particular is aimed at empirical software engineering research.

There are four empirical strategy quadrants that we show embedded in a circumplex in Fig. 1. In addition to labeling each quadrant with the type of strategy (Lab, Field, Respondent, and Data), we also show two representative strategies for each quadrant (e.g., Experiment and Field Studies in the Field quadrant). It is important to not confuse research strategy with research method. A research method is a technique used to collect data. For example, interviews may be used as a method for collecting information from field actors as part of a field study, or as a method for collecting data as part of a sample survey [11]. In contrast, a research strategy is a broader term [38] that may involve the use of one or more methods for collecting data. It indicates how

data is generated, whether directly or indirectly from participants or produced by the researcher, and suggests the setting used for the study.

Three of the empirical strategy quadrants (lab, field and respondent strategies) directly involve human participants (represented by a person icon in the framework). While strategies that fall in these three quadrants all involve human participants, they vary significantly in terms of the study setting. Field studies occur in the context of work. For example, a software company, or a classroom when studying the educational aspects of software engineering. Lab studies are conducted in a contrived context, often in a university or research center, and respondent studies are conducted in settings of convenience, such as a workplace, home, classroom or conference.

The fourth empirical research strategy quadrant, which we refer to as *data*, captures studies that are conducted *in silico*[4] and do not directly involve human participants, although they may use previously generated human or system data that is behavioral.

Every empirical research strategy has strengths and weaknesses to consider in terms of **research quality criteria** [30], and every study design decision leads to trade-offs in terms of the potential to increase or decrease quality criteria. We consider four quality criteria:

- **Generalizability** of the evidence over the population of human or system actors studied;
- **Realism** of the context where the evidence was collected and needs to apply;
- **Control** of extraneous human behaviour variables that may impact the evidence being collected; and
- **Precision** of the system data that is collected as evidence.

The four criteria are shown on the outside of our empirical strategy circumplex in Fig. 1, and they are positioned in proximity to the quadrants whose strategies have the potential to increase those criteria (but doing so is not a given, as we describe below). Since all strategies have inherent strengths and weaknesses, it may be important to **triangulate** across research strategies to mitigate the weaknesses of using a single strategy [30] in one's research.

Next, we describe research strategies that belong to the four different empirical quadrants in more detail. We discuss how the strategies show potential for higher or lower realization of the research quality criteria listed above.

3.3.1 Field Strategies

Field strategies involve researchers entering a natural software development setting to study the socio-technical system in action. This includes the technical systems used to support engineering activities, the system or project under development, and the human stakeholders, such as developers, engineers and managers.

---

[4] By *in silico*, we mean performed on a computer or via computer simulation.

A *field study* is where the researcher observes (e.g., through an ethnography) study subjects without any explicit interventions in place. With a field study, realism is high but control over human activities is low, and generalizability is low as only a limited number of companies are typically considered. For example, a researcher may observe how agile practices are adopted in a startup company, leading to descriptive insights about the practice in a realistic but specific setting.

A *field experiment* occurs in a natural development setting but one where the researcher controls certain aspects of the setting and may aim to compare the impacts of different solutions or environment conditions. A field experiment may be more obtrusive than a field study, and thus will lower realism, but it has the potential for higher control over human participants' activities. For example, a field experiment may involve the comparison of a novel automatic testing tool to an existing tool in the developers' realistic and natural development setting. A study that only considers data traces from the field (e.g., from a data mining study or use of machine learning in an experiment) is categorized as a data study (see below) as these studies do not involve the direct involvement or observation of human participants in their natural environment.

3.3.2 Lab Strategies

Lab strategies typically involve testing hypotheses in highly controlled situations or contrived environments with human participants and technical systems as actors. Control of human actor activities may be achieved but at the expense of realism (the setting is contrived), and it may be more difficult to achieve generalizable results.

A *lab experiment* is one lab strategy where the experimenter controls the environment and interventions or tools used. For example, a researcher may investigate the effects of a new debugging tool in comparison to the *status quo* debugger on programming task efficiency using graduate students as participants in a lab setting.

In comparison, an *experimental simulation* may try to increase realism by setting up an environment that mimics a real environment in industry. For example, a researcher may investigate different modalities for project management meetings in an environment that is similar to a collaborative meeting room in a company. Doing so increases realism but at the expense of control over variables that may come into play in a simulated environment where human actors may have more freedom to act naturally. Of note is that many lab strategy studies tend to use students in software engineering research. As part of our analysis, we also noted which of these studies involved practitioners and mention this result when we discuss the implications of our findings.

3.3.3 Respondent Strategies

Respondent strategies are often used in software engineering research to gather insights from practitioners and other stakeholders. The ability to collect data from many participants using respondent strategies has the potential for higher generalizability (if a broad and large sample is recruited), but at the expense of lower realism as factors that may influence each individual participant's responses cannot be observed or anticipated.

A *sample survey* is a respondent strategy which may, for example, involve an online questionnaire or a set of interviews to gather opinions from human participants. For example, a questionnaire or interviews may be used to learn how developers perceive code quality, or to learn how continuous integration tools are used by developers and which challenges they encounter using these tools.

A *judgment study*, another respondent strategy, asks human participants to give their expert opinion on the effectiveness of a new tool or process. Typically participants try out a new tool or process in a setting of convenience. For example, a researcher may wish to ask developers for their opinion about a new debugging tool by asking them to use it briefly in place of their regular tool and to provide an opinion. Judgment studies are distinguished from laboratory and field strategies by the use of an environment of convenience and by the lower control in how the tool is used.

3.3.4 Data Strategies

Data strategies refer to empirical studies that rely primarily on archival, generated or simulated data. The data used may have been collected from a naturally running socio-technical system, or it may be partially generated in an experiment. This type of research strategy is frequently used in software engineering research due to the technical components in software engineering, as well as the widespread availability of extensive trace data and operational data from modern software development tools.

Data strategy papers may use a wide range of specific methods, including *experiments*, to evaluate and compare software tools (e.g., such as a defect prediction tool) with historical data sources. Data strategy papers may also refer to *data mining studies* used to gather descriptive insights about a socio-technical system. They are sometimes used to infer human behaviours from human-generated, behavioral trace and operational data (e.g., source code, commit comments, bugs). However, as other researchers have reported, this data is inadequate for understanding previous human behaviour [1] and can be misleading in terms of predicting future developer behaviours [21]. Data strategy papers have lower control over human behavioral variables but have the potential for higher precision of system data. They may have potential for higher realism (if the data was collected from field sites) and higher generalizability (if the project data is from many projects).

In the next two sections of the paper, we describe how we used the Who-What-How research framework to answer our research questions. Later, we discuss how the framework may be used to reflect on and guide software engineering research.

4 Methodology

To answer our research questions (see Section 1), we read and analyzed ICSE Technical Track and EMSE journal papers from 2017.

We selected the ICSE venue for our analysis because it is considered the flagship conference in software engineering, and because one would expect it represents the broad areas of software engineering research. We considered EMSE as it is one of the top two journals in software engineering and focuses on empirical studies. We selected these venues because they both cover a broad set of topics—we selected EMSE in particular because of its focus on empirical software engineering research. We do not aim for broad coverage of all SE venues, but rather two exemplar flagship venues that can illustrate our framework.

ICSE 2017 had 68 papers in the technical track, and EMSE published 83 papers in 2017 (including special issues but excluding editorials). This resulted in a dataset of 151 papers.

As we read the papers, we answered the questions posed using the framework described above in Section 3 and recorded our answers in a shared online spreadsheet (available in our replication package at DOI: 10.5281/zenodo.3813878). This process was highly iterative and our framework (and coding categories) emerged from earlier rounds. In particular, the need to consider "precision" as a quality criterion in our framework emerged when we realized that many of the papers published in software engineering rely on datasets procured from technical components that offer precise measurement. When the framework had stabilized to its current form, each paper was read and coded by at least two (but often three or more) members of our research team. Any differences in our responses to the framework questions (e.g., should this paper be described as benefiting researchers, systems, or humans) were discussed to decide what the most appropriate answer should be. We did this by reading the paper again, and for some, recruiting an additional member of our research team to read and discuss the paper. Initial disagreements were easy to resolve as additional information was often contained in an unexpected section of a paper (e.g., a small judgment study may have been done but not discussed until later in the paper). The process was laborious as sometimes research strategies or beneficiaries were mentioned in unexpected places. We always relied on the paper author's own words to justify our choices. For example, a paper had to explicitly refer to a human beneficiary ("developers" or "programmers", for instance) for us to justify coding it as such. Our selected answers, with quotes or comments to justify coding that was more subjective

or saw some disagreement, can be found in our artifact package (see DOI: 10.5281/zenodo.3813878).

RQ1 is to identify *who* is the main beneficiary of the research. To answer **RQ1** we considered both the framing of the research questions and the introductions of the papers. In some cases, it was difficult to answer this question and we had to refer to the discussion and/or conclusion of the papers to identify who or what was the claimed or intended beneficiary of the research. In the case where we identified that a human stakeholder was a stated beneficiary, we copied a quote from the paper into our analysis spreadsheet. We deemed something as involving human beneficiaries (as 'human' or 'both') if the paper contained a statement referring to human stakeholders. To find this mention, we read each paper, augmenting our reading with keyword searches for 'developer', 'human', 'user', 'tester', 'engineer', 'coder', and 'programmer'.

Mention of humans in the paper did not always include an in-depth discussion of how a human might benefit from the tool or process. Some papers only tangentially mentioned that humans might benefit from their approach. For example, Lin *et al.*'s [59] ICSE 2017 data strategy paper, "Feedback-Based Debugging", mentions that their "approach allows developers to provide feedback on execution steps to localize the fault". By comparison, a paper we coded as referring to a system as the only stakeholder noted that "LibD can better handle multi-package third-party libraries in the presence of name-based obfuscation" (Li *et al.* [58]).

To answer **RQ2** and to identify *what* the main research contribution is, we read the abstract, introduction, results/findings, discussion and conclusions of each paper. Again, categorizing a paper as a solution or descriptive paper was not always straightforward, but we relied on how the authors framed their results to decide which was the main contribution. When a descriptive paper concluded with a proposed solution that was not evaluated, we coded it as a descriptive paper—in fact, the solution was often provided as a way to justify the impact of the descriptive results reported. Many solution papers had a (usually) small descriptive contribution, thus coding as both descriptive and solution would not allow us to discriminate the contributions in the papers we analyzed. For example, a paper that provides a theory of how continuous integration improves software quality or describes challenges with using continuous integration tools would be categorized as descriptive, whereas a paper that proposes and/or evaluates a new continuous integration tool would be categorized as a solution paper.

To answer **RQ3** and to identity *how* the research was conducted in terms of the research strategy used, we focused on the methodology sections of the papers. Again, we used our framework to help distinguish different kinds of strategies. For each paper, we noted if one or more strategies were used. Sometimes an additional strategy was mentioned as a small additional step in the reported research in the discussion or background sections of the paper, so we also read the entire paper to be sure we captured all the strategies reported. Although not one of our main research questions, we also coded which papers

directly involved industry practitioners in their studies. We discuss this finding in the discussion section of this paper.

To answer **RQ4** and to identify how the reported research strategies *map* to the beneficiary and type of research contribution, we mapped our responses in our spreadsheet and visualized the results. For this question, we expected to see that papers with a human beneficiary may be more likely to also use research strategies that directly involve and control for human behaviours. In terms of how research contributions map to beneficiary and research strategy, we were curious to see if there were some patterns in this regard as we did not have an initial expectation about this mapping.

For the purposes of replication and traceability, we provide our methodological tools, the anonymized raw data, and analysis documents and spreadsheet at DOI: 10.5281/zenodo.3813878.

5 Findings

We present the findings from applying the Who-What-How framework to ICSE conference and EMSE journal papers published in 2017. We interpret and discuss the possible implications of these findings later, in Section 6.

5.1 RQ1: Who are the **intended beneficiaries** of the published research?

Figure 2 shows an upset plot [24] of the intended beneficiaries. We were liberal in coding the possible beneficiaries. For a paper to be coded as Human, we checked if an author noted their paper described a tool that could solve a *human* problem (e.g., improve developer productivity) or benefit human or organizational stakeholders. For example, in the 2017 EMSE journal paper, "A robust multi-objective approach to balance severity and importance of refactoring opportunities" [60], the primary contribution is for a System (Note: all emphasis is ours):

> "The results provide evidence to support the claim that our proposal enables the *generation of robust refactoring solutions* without a high loss of quality using a variety of real-world scenarios."

However, we also identified a claim for human benefits in this paper:

> "The importance and severity of code fragments can be different after new commits introduced by *developers*. [...] the definition of severity and importance is very subjective and depends on the developers perception."

Likewise, if a paper that predominantly studied human behaviours mentioned their findings may improve a tool, we coded the paper as also benefiting a *system* component. We had a flexible interpretation of 'tool'. For example, in the ICSE 2017 paper, "How Good is a Security Policy against Real Breaches? A HIPAA Case Study" [55], the tool in question is "a formal representation of

security policies and breaches [and] a semantic similarity metric for the pairwise comparison of norms", but the beneficiaries include human stakeholders:

> "Our research goal is to help *analysts* measure the gaps between security policies and reported breaches by developing a systematic process based on semantic reasoning."

One could conclude that all papers benefit researchers in some way. However, we coded papers as benefiting Researchers when a main focus of a paper contribution was clearly aimed at researchers (e.g., in the case of a benchmark for future research use). We found that EMSE reports more systematic literature reviews, papers that lead to artifacts, and benchmarks aimed at researchers than ICSE does.

Since our coding permitted one or more of our three beneficiary types (Human, System, Researcher), we can clearly see multiple beneficiaries in the up-
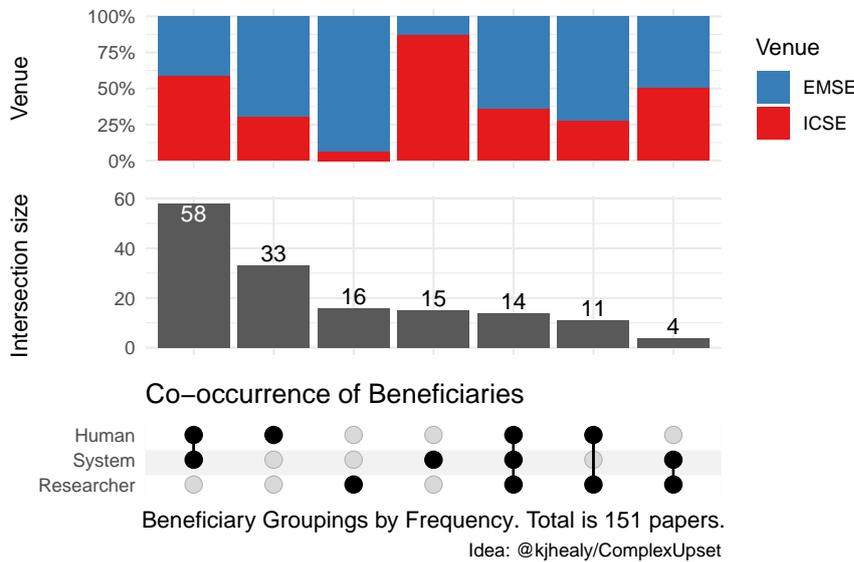

Fig. 2: Intended beneficiaries of the research contributions. This upset plot [24] captures how our beneficiaries overlap. The bottom filled/unfilled circles represent set membership, i.e., a filled circle indicates papers we coded with that beneficiary. Proceeding vertically up the diagram, the intersection size represents the overall number of papers with those memberships. For example, papers with Human&System beneficiary were 58/151 of the papers in the data we coded. The Venue plot shows what proportion of those papers were found at EMSE, or at ICSE. For the Human&System papers, over half were published at ICSE. On the other hand, we see EMSE in 2017 published more papers with only Researchers, while ICSE 2017 published more papers with System as the only beneficiary.

set plot—for All, Human&System, Human&Researcher, and System&Researcher. We found that the majority of papers from both venues claim human stakeholders as a possible beneficiary. Specifically, 77%/76% of the EMSE/ICSE papers (respectively) claim their research may benefit human stakeholders— many also claim technical systems and/or researchers as beneficiaries—but we find that more of the EMSE papers (27%) claim humans as the sole beneficiary compared to 15% of ICSE papers.

5.2 RQ2: What type of **research contributions** are provided?

Table 1 illustrates how many Descriptive vs Solution papers we captured using the framework. Note that many of the papers we coded as solution papers also had a descriptive contribution, either in terms of problem understanding or solution evaluation. For example, "Code Defenders: Crowdsourcing Effective Tests and Subtle Mutants with a Mutation Testing Game" from ICSE 2017 [61] describes problems with generating effective mutation tests. We coded this as Solution since the main contribution is a code defender multiplayer game for crowdsourcing test cases. However, in the process of conducting the empirical study, the paper describes many of the problems with the approach.

If the solution contribution was minor (e.g., a recommendation for a new tool following a mostly descriptive paper), we coded the paper as a Descriptive paper. For example, we coded "An initial analysis of software engineers' attitudes towards organizational change" [57] from the 2017 EMSE journal as a Descriptive study using a respondent strategy based on a survey to describe attitudes. One outcome of the paper, however, is:

> "[a] proposed model [that] prescribes practical directions for software engineering organizations to adopt in improving employees responses to change".

Across both venues, 43% of papers were Descriptive, and 57% presented Solutions (see Table 1). More ICSE papers were identified as Solution papers, and most solutions were technical in nature. ICSE and EMSE published 81%/37% Solution papers and 19%/63% Descriptive papers, respectively. This large difference in contribution type mirrors results by Shaw *et al.*'s analysis of ICSE 2002 papers [34] and the replication of her study in 2016 [40] that found lower submission and acceptance of papers with results that were seen as "qualitative and descriptive models".

Table 1: Counts and Proportions of Research Contributions Per Venue

Purpose	All		ICSE		EMSE	
	Count	Proportion	Count	Proportion	Count	Proportion
Descriptive	65	0.43	13	0.19	52	0.63
Solution	86	0.57	55	0.81	31	0.37

5.3 RQ3: Which **research strategies** are used?

Figure 3 shows the framework quadrants for research strategies. We report totals for EMSE and ICSE separated by a vertical pipe character. For example, we identified 23 EMSE and 14 ICSE papers using respondent strategies (shown on the figure as 23|14). Since some papers report two strategies, the totals add up to more than the 151 papers we analyzed. Section 6.4 expands on how some papers use triangulation of research strategies.
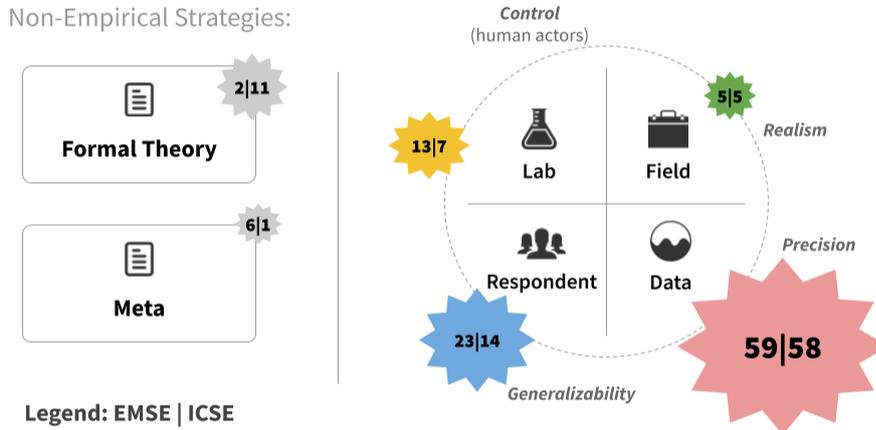

Fig. 3: Counts of the research strategies used in the EMSE/ICSE 2017 papers, respectively. Note: Some papers reported more than one strategy.

Among the 151 papers we examined, we found a higher use of Data strategies (59 EMSE | 58 ICSE) compared to any of the other research strategies (see Fig. 3).

We expand on how we classified papers with specific examples. We provide the complete classification in our replication package available at DOI: 10.5281/zenodo.3813878. A short sample is provided in Appendix B.

For Data strategies, an example is from Christakis *et al.*'s ICSE 2017 paper, "A General Framework for Dynamic Stub Injection" [48]. This paper describes their novel stub injection tool and subsequent evaluation by running it on a series of industry applications which they used to instrument system calls and monitor faults. Since this paper generates data on system faults based on their tool, we classified this as a Data strategy.

A second example of a paper we coded as Data strategy is Joblin *et al.*'s ICSE 2017 paper, "Classifying Developers into Core and Peripheral: An Empirical Study on Count and Network Metrics" [54], which analyzes commit data from GitHub projects to study aspects of human behavior and uses prediction algorithms to classify developers as core or peripheral in open source GitHub projects. This descriptive study relies on GitHub trace data.

There were significantly fewer instances of the other empirical research strategies: Field (5 EMSE | 5 ICSE), Lab (13 EMSE | 7 ICSE), and Respondent (23 EMSE | 14 ICSE). We discuss the possible implications of this imbalance of research strategy use in Section 6, but first we give examples of each of the non-data quadrants shown in Fig. 3. We also report the number of papers we identified for each strategy (totals and EMSE|ICSE).

Field Strategies

We identified 8 papers (4|4) that conducted field studies (total of both venues). Heikkil *et al.* [51] conducted a large field study at Ericsson examining requirements flows for a paper in the 2017 EMSE journal. Since the interviews were done with Ericsson developers in their natural work environment, realism was potentially high for this field study:

> "We present an in-depth study of an Ericsson telecommunications node development organization .... Data was collected by 43 interviews, which were analyzed qualitatively."

Field experiments were relatively uncommon (2 in total, 1|1). In their ICSE 2017 paper, He *et al.* [53] report a data study followed by a field experiment when studying test alarms. They integrated their tool into the system of their industry partner and observed the results of practitioners using the tool. This significantly increased the realism of the study.

Lab Strategies

For laboratory experiments, we identified 16 papers in total (11|5). An example is Charpentier *et al.*'s 2017 EMSE journal paper, "Raters' reliability in clone benchmarks construction" [47]. The study used a combination of experts and students in a lab setting to rate software clone candidates, which were then used to evaluate clone detection tools. While there was a portion of the strategy in which respondents—their subjects—were asked to evaluate the clones, this is not a respondent strategy but rather a particular method used in the lab experiment. The experimenters controlled the conditions under which clones were evaluated in order to evaluate the independent variable of rater experience.

Experimental simulations occurred 4 times (2|2). In their ICSE 2017 paper, "Do Developers Read Compiler Error Messages?" [45], Barik *et al.* conducted an eye-tracking study of students at their institution. Participants were asked to use error messages to debug software programs, and eye-tracking hardware was used to understand how participants solved the problem. Since this was not a controlled experiment, we coded this as an experimental simulation (a Lab quadrant strategy) since the main objective was to simulate, to some extent, the realism of actual debugging.

Respondent Strategies

Online questionnaires (surveys) and interviews were common ways to implement a sample survey strategy in our sample. Sample surveys occurred 21 times (16 EMSE | 5 ICSE).

The NAPIRE survey series, published in the 2017 EMSE journal and authored by Méndez *et al.* [50], is a good example of using a series of online questionnaires to understand the problems practitioners have with requirements engineering. The surveys are broadly distributed to maximize generalizability from respondents.

Hoda and Noble's ICSE 2017 paper, "Becoming Agile: A Grounded Theory of Agile Transitions in Practice" [52], used interviews to gather responses from various developers about their experiences transitioning to Agile development methods in their work. Because these researchers gathered responses from a wide variety of developers working in different settings, this increased the generalizability of their findings to other development contexts beyond those represented in the studies. Had they instead focused on a single company in detail, we would have coded this as a field study rather than a respondent strategy (sample survey).

The other type of respondent strategy is the judgment study, which occurred 16 times (7|9). This strategy can be seen in Bezemer *et al.*'s 2017 EMSE journal study, "An empirical study of unspecified dependencies in make-based build systems" [46]. In this paper, the authors began with a data strategy and then asked for professional feedback on the results:
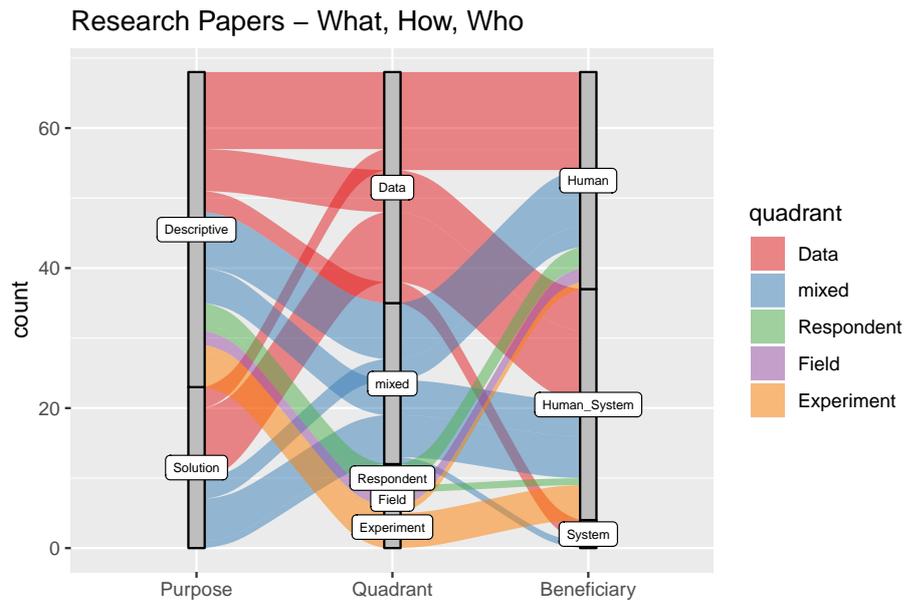
> "We contacted a GLIB developer to validate the patches that we submitted."

Similar studies might post results as pull requests (for bug fixes, for example) and monitor acceptance rates. Many of the judgment studies we saw involved few participants so these studies did not realize the high potential for generalizability that could have been achieved with a respondent strategy.
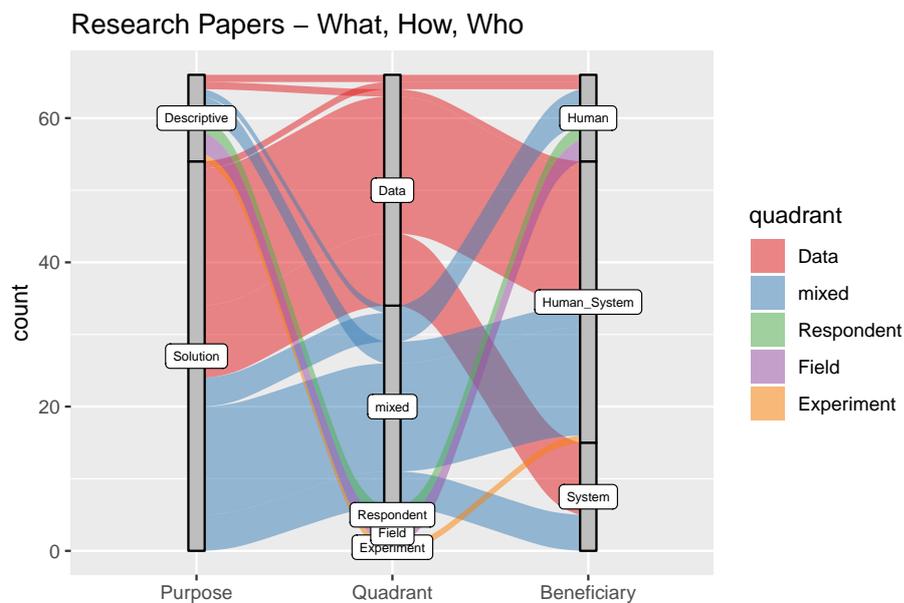
Non-Empirical Strategies

For the non-empirical strategies, 13 papers (2|11) contained an aspect of formal methods or developed a formal theory, such as Faitelson and Tyszberowicz's ICSE 2017 paper, "UML Diagram Refinement (Focusing on Class- and Use Case Diagrams)" [49]. A total of 7 non-empirical strategy papers (6|1) were meta papers, i.e., aimed at other researchers (such as systematic literature reviews or discussion of research methods). For example, "Robust Statistical Methods for Empirical Software Engineering" by Kitchenham *et al.* in the 2017 EMSE journal [56] aims to:

> "explain the new results in the area of robust analysis methods and to provide a large-scale work example of the new methods."

(a) EMSE alluvial diagram

(b) ICSE alluvial diagram

Fig. 4: These two alluvial diagrams capture flows from human and system beneficiary (Who), to empirical research strategy quadrant (How), to purpose (What). The size of the lobes in each strata (column) reflects proportion of classified papers (for example, there were few papers classified as benefiting researchers, in the left-most column). The width of the alluvia (flow lines) likewise captures proportion and is colored by the chosen research strategy captured in the legend. For the **EMSE papers** (top, Fig. 4a), we see an emphasis on human beneficiaries but more use of data strategies producing more descriptive contributions. For the **ICSE** papers (bottom, Fig. 4b) we see a more even emphasis on human and system beneficiaries but more use of data strategies over other strategies, leading to more solution oriented contributions.

5.4 RQ4: How do the reported research strategies **map** to the beneficiary and types of contributions in the 2017 EMSE and ICSE papers?

To illustrate the mappings from research strategy to research contribution type and beneficiary, we created the alluvial flow diagrams shown in Fig. 4. These diagrams outline the mappings between research beneficiary (RQ1), research quadrant (RQ3), and research purpose (RQ2) for the two venues. We show only the human and system beneficiary categories on the left side, omitting the few papers that focus on researchers only as beneficiary (to improve the clarity of our diagram).

The alluvial diagrams show that both venues report a high use of data strategies (the how) despite some key differences in beneficiaries (who) and contribution types (what). That said, there is more use of non-data strategies (that is, strategies that directly involve human participants) in the EMSE papers (47% of EMSE empirical papers, compared to 36% of the ICSE empirical papers). This may be because more EMSE papers aimed at human stakeholders as the sole claimed beneficiary.

For ICSE papers, the majority of non-data strategy studies map to descriptive research contributions, and the majority of data strategy studies map to solution contributions. For the EMSE papers, by contrast, we see that many of the data studies map to descriptive contributions (EMSE has fewer solution papers), but we note that many of these descriptive contributions aimed at humans as the beneficiary. We discuss the possible implications of not involving human actors in studies that are aimed at human beneficiaries later in the paper (see Section 6.2).

6 Interpreting Our Findings

This section presents our interpretation of the findings, with key insights highlighted in bold. First, we discuss why we see an emphasis on data strategies in software engineering research (for both venues), and how data strategies may be limited for understanding and experimenting with the human and social aspects in software engineering. Then, we discuss how research strategies are not triangulated as much as we may expect in papers presented in these venues, and how more triangulation of research strategies could bring more attention to human and social aspects in software engineering research and practice.

6.1 A Penchant for Data Strategies

We found that the majority of the ICSE conference and EMSE journal papers we analyzed relied on data strategies (71% and 85%, respectively), and over half of the total papers relied solely on data strategies in their choice of method. There are several reasons that may help explain why data strategies are commonly used in software engineering research.

Data strategies are **well-suited to understanding and evaluating technical aspects** for papers that focus on technical systems as the beneficiary. Many authors in the set of papers we analyzed aimed to improve a technical system or component as one of their main research beneficiaries. We found that most of the ICSE papers we analyzed are solution papers, and data strategies are used in many solution papers to show the feasibility and scalability of technical solutions. And for descriptive papers—which the majority of the EMSE papers we analyzed can be considered—much can be learned from data about technical systems, and valuable knowledge of human and social aspects can also be gleaned from data alone.

In recent years, there is **increased availability of data** from software repositories and diverse data sources concerning software projects. These data sources encompass a rich resource concerning both technical and human aspects for conducting empirical software engineering research. We have access to open or proprietary source code when research collaborations are in place; development data, such as issues, bugs, test results, and commits; crowdsourced community knowledge repositories, such as Stack Overflow and Hacker News; and operational and telemetry data from the field, such as feature usage, A/B testing results, logging data, and user feedback [16]. Note that the analysis of ICSE papers from 2016 by Thiesen *et al.* [40] also showed an increase in mining software repository papers (one type of data strategy paper) over Shaw's earlier study on ICSE 2002 papers [34].

Data analysis is a core skill that many computer scientists and software engineering researchers possess as they are more likely to come from scientific, engineering and mathematical backgrounds, with expertise in statistics, data mining, natural language processing and AI. In addition, the emergence of new and powerful machine learning and AI techniques that scale to software engineering projects and is taught as part of software engineering curricula also may help explain why data strategies are used so frequently in our field.

Data strategy studies have the potential to achieve high **generalizability**, particularly when multiple projects are studied. The analysis of data from repositories hosted on sites such as GitHub may be more generalizable to a broader set of projects. Similarly, using data from real-world projects has the potential to increase **realism**, particularly in terms of the technical systems studied.

Data strategies also lend themselves naturally to **replication**, a desirable aspect to improve scientific rigor. We are starting to see more data strategy papers in software engineering that are accompanied (either as a requirement or optionally) by replication packages that include the software artifact data, algorithms, scripts, and other tools used in the studies. These packages are recognized by the community as having high potential for replication. Some evidence further suggests that replication packages and open science leads to more citations [5], which might also motivate this strategy choice. Other strategies (such as lab, field, and respondent) can also provide replication packages, but exact replications are more difficult when human participants are involved as their behaviour can be impacted by more nuanced variables,

and their behaviour is not as predictable as technical components in studied socio-technical systems.

As our answer to RQ4 shows, many data papers aim at humans as a beneficiary, and while some triangulate using another strategy that involves human participants, many do not directly study humans at all. We discuss the implications of this below, but first discuss why an over reliance on data strategies may be a problem.

6.2 The Downside of Using Data Strategies for Studying Human Aspects

Our analysis found that many papers in both venues we inspected claim their research would benefit humans in some way, but did not directly involve them in their studies: 43% and 54% for EMSE and ICSE (respectively) claim they benefit humans but did not directly study them, instead relying on data traces.

In some cases, this may be justified. For example, a study that evaluates a new technique to improve the build time of a project may not benefit from human feedback as the faster compilation probably implies a better developer experience.

But many proposed solutions may need to be evaluated in a human stakeholder's context. For example, a solution paper that proposes a recommender system of possible defects may need to be evaluated with human actors to see how the tool perturbs the context in which it is used [21], and to control for other variables that may be important when human actors are involved (e.g., the technique may lead to information overload or other types of complacency). Moreover, solution evaluations that rely on historical data alone assume that future developers will use the new intervention in exactly the way the previous intervention was used, but this is not likely the case [21].

For descriptive research that aims to capture human and social aspects, data alone may also not tell the whole story and any conclusions drawn should be corroborated with other methods (such as interviews, surveys, or observations). For example, Aranda and Venolia showed in their paper [1] that many important aspects of software bugs cannot be discerned from data alone. Similarly, two papers that look at Git and GitHub as rich data sources ("Promises and Perils" of Mining Git [3] and GitHub [17]) highlight the many potential pitfalls when using these data sources alone.

6.3 Why Are Human-Oriented Research Strategies Less Common in Software Engineering Research?

We saw relatively few respondent, laboratory and field strategies in the ICSE and EMSE papers we analyzed (see Fig. 3). Furthermore, many of these were presented as a secondary strategy to a data strategy and were of limited scope:

Table 2: This table shows the number/proportion of papers that involve **practitioners** in reported empirical studies for the two venues and for both combined (all).

Practitioners?	All		ICSE		EMSE	
	Count	Proportion	Count	Proportion	Count	Proportion
True	41	0.27	17	0.25	24	0.29
False	110	0.73	51	0.75	59	0.71

some were informally conducted or reported[5]). Although these strategies show the potential for advantages in terms of generalizability, realism, and control of human behaviour variables, these strategies are seldom used, either as a main or secondary strategy, even for research that claims to benefit human stakeholders. We do not know if the reason for this lack of use may be due to fewer submissions of such papers, or if they are less likely to be accepted.

One possible reason for the apparent low use of these strategies across the paper venues we analyzed may be a **lack of expertise** by software engineering researchers that publish in those venues. Strategies that study human behaviours require expertise that is not typically taught in a software engineering or computer science educational program (when compared to sociology and other social sciences).

A possible reason for the very few lab and field studies may be due to **limited access** to developer participants and sites. Field studies can be **obtrusive** and **practitioner time** is expensive. We found that only 27% of the papers (29% and 25% of the EMSE and ICSE papers, respectively) reported involving practitioners in their empirical studies and these counts include several studies where just one or two practitioners were used in a small judgment study (see Table 2).

Another possible cause for the low appearance of laboratory and field studies may be because of their potential **lack of generalizability**. Reviewers can easily attack poor generalizability as a reason to quickly reject a paper, which may deter researchers who have tried to use such strategies [43].

Field and lab strategies also often rely on the use of **qualitative methods** and should be evaluated using quite different **criteria** than those used to evaluate quantitative research. Reviewers that expect to see threats to validity, such as external, internal, and construct, may find qualitative criteria, such as credibility and transferability, as unacceptable and unfamiliar due to a **different epistemological stance** [13].

Respondent strategies were used more often than lab and field strategies in the papers we analyzed, but not as often as we had anticipated. Respondent strategies are often seen as easier to implement for many researchers as they are done in settings of convenience. But conducting surveys and analyzing survey data bring other challenges: designing surveys normally takes

---

[5] For example, one EMSE paper we read reported a user study but did not indicate how many participants were involved, nor who the participants were.

several iterations, and recruiting survey respondents that are a good sample of the studied population is challenging. Furthermore, conducting open-ended surveys is **time consuming**[6].

Finally, running studies with human subjects requires the additional step (in many academic institutions) of acquiring approval from an **ethics** review board [37], and the use of human subjects inevitably introduces other complications (sometimes people do not show up or have unique abilities that impact the study). This step is generally not needed for data strategies, although ethical concerns about the use of some data resources have been raised in the research community.

6.4 Using Triangulation to Balance Benefits for Human and Technical Aspects in Empirical Software Engineering Research

Denzin [8] describes several different types of triangulation in research: investigator triangulation, data triangulation and methodological triangulation. Methodological triangulation refers to using different strategies and/or methods, while data triangulation refers to when the same research method may be used but different sources of data are used. Investigator triangulation refers to the use of different investigators in running studies to reduce investigator bias. Triangulation of research strategy (what Denzin refers to as methodological triangulation) is how researchers can improve the balance of desirable research quality criteria [30]. Our framework's circumplex (Fig. 1) highlights how each strategy has strengths and weaknesses when it comes to improving generalizability, enhanced control over variables that may be influenced by human actors, improved study realism, and more precision over data measurements. The choice of research strategy, and more specifically which methods form part of that strategy, indirectly influences *who* benefits from the research: practitioner stakeholders, technical components/systems or researchers.

The potential impact of the choices made by the authors of the papers we analyzed on four research quality criteria is summarized in Fig. 3. Most notably, few papers report on research strategies that attempt to control variables that come into play when human actors may be involved (e.g., very few report on lab and field experiments). Papers that do not control for human variables are limited in how they may claim their research is relevant to or benefits human actors. Realism is also potentially low in the cases where a realistic evaluation should involve human participants when the research claims to benefit them. In contrast, the use of data strategies may help improve generalizability when datasets from multiple cases are considered, and such papers potentially achieve higher precision over data measurements from the technical systems studied.

In our study of the EMSE journal and ICSE conference papers from 2017, only 37 papers (24%/25% of the EMSE/ICSE papers respectively) reported

---

[6] http://www.gousios.gr/blog/Scaling-qualitative-research.html

studies from more than one research strategy quadrant. Figure 5 shows which strategies were triangulated with one another. For the data strategy papers, the vast majority did not triangulate their findings with other strategy quadrants, but when triangulation did occur, data strategies were mostly triangulated with a respondent strategy (surveys or interviews).

Although triangulation of research strategy was low within the papers we analyzed and across the venues we studied, we found that data triangulation was quite common in our study sample. Data triangulation occurred when a paper author replicated results with additional cases (datasets) using data-driven strategies. While the use of data triangulation may improve generalizability, it does not improve realism or control over human variables. These latter two criteria are particularly important to improve for research that is explicitly aimed at human beneficiaries.
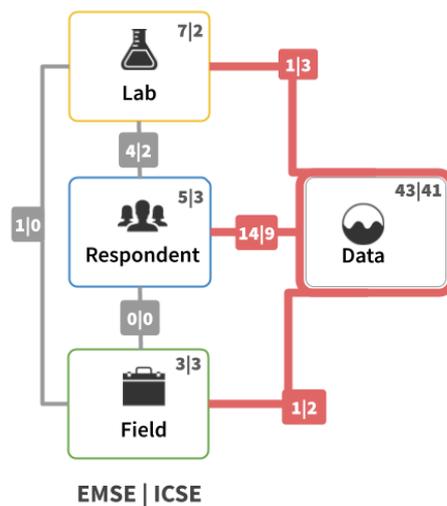

Fig. 5: **Triangulation** of research strategies reported in our set of EMSE/ICSE papers, respectively. Numbers placed on the edges indicate the number of papers that triangulate using that pair of strategies. The red (thicker) edges indicate triangulation of data strategies with non-data strategies. The numbers in the white strategy boxes indicate number of strategies that were not triangulated with other strategies.

In terms of diversifying strategies, there are other benefits and drawbacks to be considered (in addition to realism, control over human variable, generalizability and data precision). Field strategies often lead to immediate actionable descriptive insights and understanding of why things work the way they do, as well as to innovative solution ideas from watching how experienced and

creative developers deal with or work around a problem in the real world. Likewise, non-adoption of a particular solution in the field can lead to innovative solutions. On the other hand, data strategies may be easier to replicate, while both respondent and data strategies may more easily scale to larger and broader populations, potentially increasing generalizability.

We could not determine whether authors triangulated their work outside of the papers we considered, and we recognize that the responsibility for triangulation does not need to be on the level of individual papers. Given that EMSE is a journal and many papers extend conference papers, we expected to see **more triangulation** in EMSE papers, however, we did not see this in our analysis (see Fig. 5). For ICSE papers, we see that data strategies are frequently used for solution papers but more diverse strategies are used for descriptive papers. For EMSE, we see that even descriptive papers rely more on data strategies alone. Of course, the EMSE authors of descriptive papers may have relied on data strategies that use human-generated trace data to study human and social aspects, but in doing so, may have missed other context variables that limit the results.

Finally, the choice of research strategy only offers the *potential* to achieve a given criterion. Many studies do not maximize this potential: they may use students instead of professional developers (reducing realism, see Table 2), have low statistical power (reducing generalizability), convenience sample from unrepresentative populations (reducing control), or make questionable analysis choices (reducing precision). Each method must still be judged on its merits and according to best practices.

In sum, multiple research strategies should be used to support triangulation, not just to triangulate specific findings, but to further add to insights concerning software engineering problem contexts, possible solutions for those problems, and the evaluation of those solutions [12]. Doing so will allow our research to be more relevant and transferable to broader problem contexts, leading to richer theories about why proposed solutions work or do not work as expected in certain human and social aspects.

7 Limitations

We identify the limitations associated with this research and the measures we took to mitigate these issues through our research design. We use the Roller and Lavrakas [28] "Total Quality Framework" (TQF) for qualitative research, consisting of the subdomains "Credibility" (of the data collection), "Analyzability" (of data analysis), "Transparency" (of the reporting), and "Usefulness" (of the results). The TQF, being specifically derived from experiences in qualitative research, is more relevant to this paper than the typical "internal/external/construct" frame that is often applied in statistical studies. It closely parallels the discussions of high-quality qualitative research in Miles, Huberman and Saldana [26], Kirk and Miller [18], and Onwuegbuzie

and Leech [27], among others (a more complete discussion can be found in Lenberg *et al.* [22]).

7.1 Credibility

Credibility is an assessment of the **completeness and accuracy** of the data gathering part of the study.

The scope of our study was limited to full research papers from the EMSE journal and ICSE conference in 2017. Different years, different venues, and different tracks may have produced a different distribution of research beneficiary, research contribution, and research strategy use. However, we show our analysis as an example of the Who-What-How framework applied to two venues and do not claim this to be exhaustive. We know from an earlier iteration of a similar study that considered the preceding two years at ICSE (2015 and 2016), as well as ICSE 2017, that the trends were extremely similar [43]. Journals and conferences are different venues, and in software engineering they are seen to serve different purposes so some differences were expected. However, EMSE and ICSE serve similar communities. For example, 26 of the 201 members of the 2020 ICSE Technical Program Committees and EMSE Editorial Board are in common so some similarities were anticipated and observed. However, the Who-What-How framework did help illuminate some similarities and differences between the ICSE and EMSE venues.

As mentioned earlier, there are also other venues that clearly focus on human aspects, including the CHASE workshop that has been co-located with ICSE since 2008, as well as other venues such as VL/HCC[7] and CSCW[8]. Thus we recognize our findings are particular to the ICSE technical track and EMSE journal, but we feel it is important to share as these venues are recognized as being inclusive in terms of topics and methods, but are also seen by many as two of the premier publishing venues in software engineering.

7.2 Analyzability

Analyzability is an assessment of the accuracy of the paper's **analysis and interpretations**.

The authors come from a predominantly human-centered software research background and so this may have influenced our interpretation of papers outside our area of expertise (e.g., for areas such as automated testing). Our analysis tasks relied on human judgment in terms of classification of the beneficiary, contribution, and research strategy in the papers we analyzed.

To consider who the **beneficiary** was, we relied on the paper text to discern if the research contribution of the paper aimed to benefit human stakeholders

---

[7] Visual Languages and Human-Centric Computing, `http://conferences.computer.org/VLHCC/`

[8] ACM Conference on Computer Supported Cooperative Work `https://cscw.acm.org`

(e.g., developers, managers, end users), researchers (e.g., tool designers), or technical systems (e.g., a build system or recommender). If the study intended for other beneficiaries, or had other beneficiaries as some eventual outcome of a wider research program, our analysis would not be able to identify this. Many papers alluded to multiple beneficiaries. In the case where this was somewhat subjective on our part, we included a quote from the paper in our spreadsheet—in particular, we included quotes when we identified the research was aimed at a human stakeholder beneficiary.

We assigned each paper a single **contribution** type for the research study (i.e., either descriptive or solution). However, we recognize this is a coarse description of a single paper's epistemological goals. For example, some studies might do exploration leading to descriptive insights and then design or propose a tool. We relied mostly on the authors' own descriptions in their papers to help us decide if descriptive or solution was the best categorization for research contribution. A richer categorization could leverage design science terminology more fully, such as in Engström *et al.* [12], but such a detailed categorization was beyond the scope of our research.

For coding **research strategy**, we relied on the strategy quadrants we identified in the *how* part of the framework we developed in Section 3. This aspect of our framework was developed with a focus on studying human aspects. In an earlier study reported by Williams (one of this paper's authors), she considered the research strategy across ICSE technical track papers from 2015-2017 and asked authors to classify their own papers according to the research strategy used [43]. She found that her categorization very closely aligned with the authors' views and any discrepancies were easily resolved (for example, an author may have been confused by the terminology we used or forgot that they used an additional strategy in their work). Although this was an earlier version of the research strategy (How) part of our framework, this earlier finding increases our confidence that our assignment of research strategy would match the authors' views at least for the ICSE 2017 papers. We note, however, that different frameworks for categorizing research strategy would lead to different categorizations (e.g., see a related framework by Stol *et al.* [39]).

To mitigate researcher bias for all of the coding we did, two of us independently coded the papers. We revisited any codes we disagreed on, and reread the papers together to arrive at an agreement. For some papers, we recruited additional readers to join our discussions. We recognize that some of our codes may still be open to interpretation, and thus we make our spreadsheet available for others to peruse and to validate our analysis. Our spreadsheet contains many quotes from the papers we analyzed to help others understand our coding decisions.

7.3 Transparency

Transparency refers to the way in which we have presented our results. We rely mainly on the replication package to ensure transparency. Our coding spreadsheet has additional information on any of the papers where we disagreed or where we felt our coding may have been subjective. We capture this in comments associated with each paper in the spreadsheet. We can also rely, to some extent, on the familiarity readers likely have with the domain. Since we write about software research, we did not see a need to provide detailed descriptions of the domain.

7.4 Usefulness

We developed and presented a socio-technical research framework—the Who-What-How framework—for reflecting on socio-technical contributions in software engineering research. We believe our framework is ready to apply to other venues (e.g., other software engineering journals, conferences, or tracks). To facilitate this further application, we provide a number of documents on our supplementary website designed to help other researchers follow our methodology. We welcome replication studies—especially triangulation studies with new research strategies—on additional years of ICSE or EMSE and other venues to explore the differences that may exist between venues and time periods in software engineering research. Finally, although we do not demonstrate this in our paper, we have found anecdotally from our students and colleagues that the framework is useful in helping design and reflect on research.

8 Conclusions

Understanding the complexities of human behavior requires the use of diverse research strategies—specifically the use of strategies focused on human and social aspects. Through our analysis of 151 ICSE technical track papers and EMSE journal papers from 2017, we found a skew towards data strategies in these publishing venues, even for papers that claimed potential benefits for human stakeholders, in addition to their claimed improvements to technical components. Relying on data strategies alone may mean we miss important aspects of the complex, socio-technical context of software engineering problems and hinder our evaluation of how tools may be used in real practice scenarios.

We might expect initial research on a socio-technical software engineering problem to consider only technical aspects or rely only on limited behavioral trace data. But at a community level, we would expect to see more studies that expand on these initial works to rigorously examine the human aspects. In the cohort of papers we analyzed, we found that a minority of data strategy papers triangulated using additional strategies, many of which were limited in scope, while the majority did not triangulate their research strategies at all.

Earlier efforts on analyzing research at particular publishing venues focused on categorizing research and empirical studies in software engineering (see Section 2), while the work we report in this paper focuses specifically on how studies approach social aspects, and discusses the trade-offs and implications for the software engineering community's collective knowledge on the choices made in the studies we examined. We encourage other researchers to use the framework and apply it to reflect on other publishing venues, and possibly compare with the EMSE journal and ICSE technical track venue analysis we report.

Although our original intention was to use this framework to reflect on human and social aspects of existing software engineering papers, we hope that the Who-What-How framework is also useful for framing or designing new or in-progress research studies, and to reflect on the implications of one's personal choice of strategies in terms of generalizability, realism, precision over data and control of human aspects.

Finally, we hope the Who-What-How framework will lead to some community-wide discussions about the overall shape of the research we do, and how certain values and expectations at the level of a particular publishing venue may impact the relevance of our research for researchers and practitioners.

References

1. Aranda, J., Venolia, G.: The secret life of bugs: Going past the errors and omissions in software repositories. In: icse, pp. 298–308 (2009). DOI 10.1109/ICSE.2009.5070530
2. Baecker, R.M., Grudin, J., Buxton, W.A.S., Greenberg, S. (eds.): Human-Computer Interaction: Toward the Year 2000. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1995)
3. Bird, C., Rigby, P.C., Barr, E.T., Hamilton, D.J., German, D.M., Devanbu, P.: The promises and perils of mining git. In: Proceedings of the International Working Conference on Mining Software Repositories (2009). DOI 10.1109/msr.2009.5069475
4. Brooks Jr., F.P.: The Mythical Man-month (Anniversary Ed.). Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1995)
5. Colavizza, G., Hrynaszkiewicz, I., Staden, I., Whitaker, K., McGillivray, B.: The citation advantage of linking publications to research data. Tech. Rep. arXiv:1907.02565, arXiv (2019). URL https://arxiv.org/abs/1907.02565
6. Cruz, A., Correia, A., Paredes, H., Fonseca, B., Morgado, L., Martins, P.: Towards an overarching classification model of cscw and groupware: a socio-technical perspective. In: International Conference on Collaboration and Technology, pp. 41–56. Springer (2012)
7. DeMarco, T., Lister, T.: Peopleware: Productive Projects and Teams. Dorset House Publishing Co., Inc., New York, NY, USA (1987)
8. Denzin, N.K.: The research act: A theoretical introduction to sociological methods. New Jersey: Transaction Publishers (1973)
9. Dittrich, Y., John, M., Singer, J., Tessem, B. (eds.): Special Issue on Qualitative Software Engineering Research, vol. 49 (2007). URL https://www.sciencedirect.com/journal/information-and-software-technology/vol/49/issue/6

10. Dybå, T., Prikladnicki, R., Rönkkö, K., Seaman, C., Sillito, J. (eds.): Special Issue on Qualitative Research in Software Engineering, vol. 16. Springer (2011). URL `https://link.springer.com/journal/10664/16/4`

11. Easterbrook, S., Singer, J., Storey, M.A., Damian, D.: Selecting Empirical Methods for Software Engineering Research, pp. 285–311. Springer London, London (2008). DOI 10.1007/978-1-84800-044-5_11

12. Engström, E., Storey, M.D., Runeson, P., Höst, M., Baldassarre, M.T.: A review of software engineering research from a design science perspective. CoRR **abs/1904.12742** (2019). URL `http://arxiv.org/abs/1904.12742`

13. Felderer, M., Travassos, G.H.: The evolution of empirical methods in software engineering (2019)

14. Feldt, R., Torkar, R., Angelis, L., Samuelsson, M.: Towards individualized software engineering: empirical studies should collect psychometrics. In: Proceedings of the 2008 international workshop on Cooperative and human aspects of software engineering, pp. 49–52. ACM (2008)

15. Guéhéneuc, Y.G., Khomh, F.: Empirical software engineering. In: Handbook of Software Engineering, pp. 285–320. Springer (2019)

16. Hassan, A.E.: The road ahead for mining software repositories. In: 2008 Frontiers of Software Maintenance, pp. 48–57 (2008). DOI 10.1109/FOSM.2008.4659248

17. Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D.M., Damian, D.: The promises and perils of mining GitHub. In: International Working Conference on Mining Software Repositories (2014). DOI 10.1145/2597073.2597074

18. Kirk, J., Miller, M.L.: Reliability and Validity in Qualitative Research. Sage Publications (1986). DOI 10.4135/9781412985659

19. Kitchenham, B.A., Pfleeger, S.L.: Personal Opinion Surveys, pp. 63–92. Springer London, London (2008). DOI 10.1007/978-1-84800-044-5_3

20. Kontio, J., Bragge, J., Lehtola, L.: The Focus Group Method as an Empirical Tool in Software Engineering, pp. 93–116. Springer London, London (2008). DOI 10.1007/978-1-84800-044-5_4

21. Lanza, M., Mocci, A., Ponzanelli, L.: The tragedy of defect prediction, prince of empirical software engineering research. IEEE Software **33**(6), 102–105 (2016)

22. Lenberg, P., Feldt, R., Tengberg, L.G.W., Tidefors, I., Graziotin, D.: Behavioral software engineering – guidelines for qualitative studies. arXiv preprint arXiv:1712.08341 (2017)

23. Lenberg, P., Feldt, R., Wallgren, L.G.: Towards a behavioral software engineering. In: Proceedings of the 7th international workshop on cooperative and human aspects of software engineering, pp. 48–55 (2014)

24. Lex, A., Gehlenborg, N., Strobelt, H., Vuillemot, R., Pfister, H.: UpSet: Visualization of intersecting sets. IEEE Transactions on Visualization and Computer Graphics **20**(12), 1983–1992 (2014). DOI 10.1109/tvcg.2014.2346248

25. McGrath, J.E.: Methodology matters: Doing research in the behavioral and social sciences. In: R.M. Baecker, J. Grudin, W. Buxton, S. Greenberg (eds.) Readings in Human-Computer Interaction: Toward the Year 2000, pp. 152–169. Morgan Kaufmann Publishers Inc. (1995)

26. Miles, M.B., Huberman, A.M., Saldana, J.: Qualitative data analysis: A methods sourcebook. SAGE Publications, Incorporated (2013)

27. Onwuegbuzie, A.J., Leech, N.L.: Validity and qualitative research: An oxymoron? Quality & Quantity **41**, 233–249 (2007)

28. Roller, M.R., Lavrakas, P.J.: Applied Qualitative Research Design: A Total Quality Framework Approach. Guilford Press (2015). URL `https://www.amazon.com/Applied-Qualitative-Research-Design-Framework/dp/1462515754`

29. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering **14**(2), 131 (2008). DOI 10.1007/s10664-008-9102-8

30. Runkel, P.J., McGrath, J.E.: Research on Human Behavior. Holt, Rinehart, and Winston, Inc. (1972)

31. Seaman, C.B.: Qualitative methods in empirical studies of software engineering. IEEE Transactions on software engineering **25**(4), 557–572 (1999)

32. Seaman, C.B.: Qualitative Methods, pp. 35–62. Springer London, London (2008). DOI 10.1007/978-1-84800-044-5_2

33. Sharp, H., Dittrich, Y., de Souza, C.R.B.: The role of ethnographic studies in empirical software engineering. IEEE Transactions on Software Engineering **42**(8), 786–804 (2016). DOI 10.1109/TSE.2016.2519887

34. Shaw, M.: Writing good software engineering research papers: Minitutorial. In: Proceedings of the 25th International Conference on Software Engineering, ICSE '03, pp. 726–736. IEEE Computer Society, Washington, DC, USA (2003)

35. Shneiderman, B.: Software Psychology: Human Factors in Computer and Information Systems (Winthrop Computer Systems Series). Winthrop Publishers (1980)

36. Singer, J., Sim, S.E., Lethbridge, T.C.: Software Engineering Data Collection for Field Studies, pp. 9–34. Springer London, London (2008). DOI 10.1007/978-1-84800-044-5_1

37. Singer, J., Vinson, N.G.: Ethical issues in empirical studies of software engineering. IEEE Transactions on Software Engineering **28**(12), 1171–1180 (2002)

38. Stol, K.J., Fitzgerald, B.: A holistic overview of software engineering research strategies. In: Proceedings of the Third International Workshop on Conducting Empirical Studies in Industry, CESI '15, pp. 47–54. IEEE Press, Piscataway, NJ, USA (2015)

39. Stol, K.J., Fitzgerald, B.: The abc of software engineering research. ACM Trans. Softw. Eng. Methodol. **27**(3), 11:1–11:51 (2018). DOI 10.1145/3241743. URL `http://doi.acm.org/10.1145/3241743`

40. Theisen, C., Dunaiski, M., Williams, L., Visser, W.: Writing good software engineering research papers: Revisited. In: 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C), pp. 402–402 (2017). DOI 10.1109/ICSE-C.2017.51

41. Weinberg, G.M.: The Psychology of Computer Programming. John Wiley & Sons, Inc., New York, NY, USA (1985)

42. Whitworth, B.: The social requirements of technical systems. In: B. Whitworth, A. de Moor (eds.) Handbook of Research on Socio-Technical Design and Social Networking Systems, pp. 2–22. IGI Global (2009). DOI 10.4018/978-1-60566-264-0

43. Williams, C.: Methodology matters: mapping software engineering research through a sociotechnical lens. Master's thesis, University of Victoria (2019). URL `https://dspace.library.uvic.ca//handle/1828/9997`

44. Zelkowitz, M.V.: Techniques for Empirical Validation, pp. 4–9. Springer Berlin Heidelberg, Berlin, Heidelberg (2007). DOI 10.1007/978-3-540-71301-2_2

References to papers within the study sample

45. Barik, T., Smith, J., Lubick, K., Holmes, E., Feng, J., Murphy-Hill, E., Parnin, C.: Do developers read compiler error messages? In: Proceedings of the ACM/IEEE International Conference on Software Engineering. IEEE (2017). DOI 10.1109/icse.2017.59. URL `https://doi.org/10.1109/icse.2017.59`

46. Bezemer, C.P., McIntosh, S., Adams, B., German, D.M., Hassan, A.E.: An empirical study of unspecified dependencies in make-based build systems. Empirical Software Engineering **22**(6), 3117–3148 (2017). DOI 10.1007/s10664-017-9510-8. URL `https://doi.org/10.1007/s10664-017-9510-8`

47. Charpentier, A., Falleri, J.R., Morandat, F., Yahia, E.B.H., Réveillère, L.: Raters' reliability in clone benchmarks construction. Empirical Software Engineering **22**(1), 235–258 (2017). DOI 10.1007/s10664-015-9419-z. URL `https://doi.org/10.1007/s10664-015-9419-z`

48. Christakis, M., Emmisberger, P., Godefroid, P., Müller, P.: A general framework for dynamic stub injection. In: Proceedings of the ACM/IEEE International Conference on Software Engineering, pp. 586–596 (2017). DOI 10.1109/ICSE.2017.60. URL `https://doi.org/10.1109/ICSE.2017.60`

49. Faitelson, D., Tyszberowicz, S.: Uml diagram refinement (focusing on class- and use case diagrams). In: Proceedings of the ACM/IEEE International Conference on Software Engineering, pp. 735–745 (2017). DOI 10.1109/ICSE.2017.73. URL `https://doi.org/10.1109/ICSE.2017.73`

50. Fernández, D.M., Wagner, S., Kalinowski, M., Felderer, M., Mafra, P., Vetrò, A., Conte, T., Christiansson, M.T., Greer, D., Lassenius, C., Männistö, T., Nayabi, M., Oivo, M., Penzenstadler, B., Pfahl, D., Prikladnicki, R., Ruhe, G., Schekelmann, A., Sen, S., Spinola, R., Tuzcu, A., de la Vara, J.L., Wieringa, R.: Naming the pain in requirements engineering. Empirical Software Engineering **22**(5), 2298–2338 (2017). DOI 10.1007/s10664-016-9451-7. URL `https://doi.org/10.1007/s10664-016-9451-7`

51. Heikkilä, V.T., Paasivaara, M., Lasssenius, C., Damian, D., Engblom, C.: Managing the requirements flow from strategy to release in large-scale agile development: a case study at ericsson. Empirical Software Engineering **22**(6), 2892–2936 (2017). DOI 10.1007/s10664-016-9491-z. URL `https://doi.org/10.1007/s10664-016-9491-z`

52. Hoda, R., Noble, J.: Becoming agile: A grounded theory of agile transitions in practice. In: Proceedings of the ACM/IEEE International Conference on Software Engineering. IEEE (2017). DOI 10.1109/icse.2017.21. URL `https://doi.org/10.1109/icse.2017.21`

53. Jiang, H., Li, X., Yang, Z., Xuan, J.: What causes my test alarm? automatic cause analysis for test alarms in system and integration testing. In: 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). IEEE (2017). DOI 10.1109/icse.2017.71. URL `https://doi.org/10.1109/icse.2017.71`

54. Joblin, M., Apel, S., Hunsen, C., Mauerer, W.: Classifying developers into core and peripheral: An empirical study on count and network metrics. In: Proceedings of the ACM/IEEE International Conference on Software Engineering, pp. 164–174 (2017). DOI 10.1109/ICSE.2017.23. URL `https://doi.org/10.1109/ICSE.2017.23`

55. Kafali, O., Jones, J., Petruso, M., Williams, L., Singh, M.P.: How good is a security policy against real breaches? a HIPAA case study. In: Proceedings of the ACM/IEEE International Conference on Software Engineering. IEEE (2017). DOI 10.1109/icse.2017.55. URL `https://doi.org/10.1109/icse.2017.55`

56. Kitchenham, B., Madeyski, L., Budgen, D., Keung, J., Brereton, P., Charters, S., Gibbs, S., Pohthong, A.: Robust statistical methods for empirical software engineering. Empirical Software Engineering **22**(2), 579–630 (2016). DOI 10.1007/s10664-016-9437-5

57. Lenberg, P., Tengberg, L.G.W., Feldt, R.: An initial analysis of software engineers' attitudes towards organizational change. Empirical Software Engineering **22**(4), 2179–2205 (2016). DOI 10.1007/s10664-016-9482-0

58. Li, M., Wang, W., Wang, P., Wang, S., Wu, D., Liu, J., Xue, R., Huo, W.: LibD: Scalable and precise third-party library detection in android markets. In: Proceedings of the ACM/IEEE International Conference on Software Engineering (2017). DOI 10.1109/icse.2017.38

59. Lin, Y., Sun, J., Xue, Y., Liu, Y., Dong, J.: Feedback-based debugging. In: Proceedings of the ACM/IEEE International Conference on Software Engineering (2017). DOI 10.1109/icse.2017.43. URL `https://doi.org/10.1109/icse.2017.43`

60. Mkaouer, M.W., Kessentini, M., Cinnéide, M.Ó., Hayashi, S., Deb, K.: A robust multi-objective approach to balance severity and importance of refactoring opportunities. Empirical Software Engineering **22**(2), 894–927 (2016). DOI 10.1007/s10664-016-9426-8. URL `https://doi.org/10.1007/s10664-016-9426-8`

61. Rojas, J.M., White, T.D., Clegg, B.S., Fraser, G.: Code defenders: Crowdsourcing effective tests and subtle mutants with a mutation testing game. In: Proceedings of the ACM/IEEE International Conference on Software Engineering (2017). DOI 10.1109/icse.2017.68. URL `https://doi.org/10.1109/icse.2017.68`

62. Stol, K.J., Ralph, P., Fitzgerald, B.: Grounded theory in software engineering research: A critical review and guidelines. In: Proceedings of the ACM/IEEE International Conference on Software Engineering, pp. 120–131 (2016). DOI 10.1145/2884781.2884833

A The Circumplex of Runkel and McGrath

Figure 6 shows a sketch of the research strategy circumplex designed by Runkel and McGrath [30] for categorizing behavioral research strategies. We adapted their model for the *How* part of our research framework. Runkel and McGrath's model of research strategies was developed
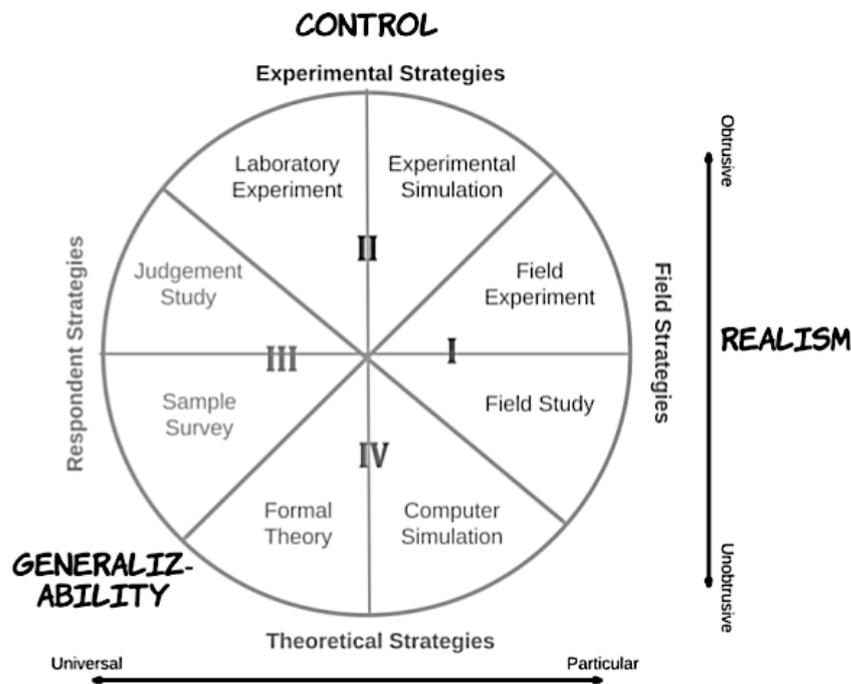
Fig. 6: Runkel and McGrath's research strategy circumplex.

in the 1970s for categorizing human behavioral research, hence it provides a good model for examining socio-technical factors in software engineering.

The McGrath model has been used by other software engineering researchers to reflect on research strategy choice and its implications on research design [11], and most recently by Stol and Fitzgerald [39] as a way to to provide consistent terminology for research strategies [39] [9] It is used extensively in the field of Human Computer Interaction [2] and CSCW [6] to guide research design on human aspects.

Three of our quadrants (Respondent, Lab, Field) mirror three of the quadrants in Runkel and McGrath's book (although we refer to Experimental Strategies as Lab Strategies as we find this less confusing). The fourth quadrant they suggest captures non-empirical research methods: they refer to this quadrant as Theoretical Strategies. We consider two types of non-empirical strategies in our framework: Meta (e.g., systematic literature review), and Formal Theory. We show these non empirical strategies separately to the four quadrants of empirical strategies in our framework. Our fourth quadrant includes Computer Simulations (which we consider empirical), but it also includes other types of data strategies that rely solely on previously collected data in addition to simulated data. We call this fourth quadrant in our framework "Data Strategies".

One of the core contributions of the Runkel and McGrath research strategy model is to highlight the trade-offs inherent in choosing a research strategy and how each strategy has

------

[9] Stol and Fitzgerald interpret and extend this model quite differently to us as they are not concerned with using their framework to discriminate which strategies directly involve human actors. Runkel and McGrath developed their model to capture behavioral aspects and we maintain the behavioral aspect in our extension of their model.

strengths and weaknesses in terms of achieving higher levels of generalizability, realism and control. Runkel and McGrath refer to these criteria as "quality criteria", since achieving higher levels of these criteria is desirable. *Generalizability* captures how generalizable the findings may be to the population outside of the specific actors under study. *Realism* captures how closely the context under which evidence is gathered may match real life. *Control* refers to the control over the measurement of variables that may be relevant when human behaviors are studied. Field strategies typically exhibit low generalizability, but have higher potential for higher realism. Lab studies have high control over human variables, but lower realism. Respondent strategies show higher potential for generalizability, but lower realism and control over human variables.

We added a fourth research quality criterion to our model, data *precision*. Data strategies have higher potential for collecting precise measurements of system data over other strategies. Data studies may be reported as 'controlled' by some authors when they really mean precision over data collected, therefore, we reserve the term control in this paper for control over variables in the data generation process (e.g., applying a treatment to one of two groups and observing effects on a dependent variable). McGrath himself debated the distinction between precision and control in his later work. We note that McGrath's observations were based on work in sociology and less likely to involve large data studies, unlike in software engineering. The Who-What-How framework (bottom of Fig. 1) denotes these criteria in italics outside the quadrants. The closer a quadrant to the criterion, the more the quadrant has the potential to maximize that criterion.

We recommend that the interested reader refer to Runkel and McGrath's landmark book [30] for additional insights on methodology choice that we could not include in our paper.

B Sample Paper Classification

Table 3 shows a 15-paper sample classified using our Who-What-How framework. Full data is available at DOI: 10.5281/zenodo.3813878.

Venue	Paper Title	Authors	Strategies	Purpose	Beneficiaries
ICSE	The Evolution of Continuous Experimentation in Software Product Development	Fabijan et al.	FS	Descriptive	Human
ICSE	Learning Syntactic Program Transformations from Examples	Rolim et al.	D	Solution	Human-System
ICSE	Glacier: Transitive Class Immutability for Java	Coblenz et al.	D/LE	Solution	Human-System
ICSE	UML Diagram Refinement	Faitelson	FT	Solution	Human
ICSE	Understanding the Impressions, Motivations and Barriers of One Time Code Contributors to FLOSS Projects: A Survey	Lee et al.	SS	Descriptive	Human
ICSE	Fuzzy Fine-grained Code-history Analysis	Servant et al.	D/JS	Solution	Human-System
ICSE	On Cross-stack Configuration Errors	Sayagh et al.	D/D	Solution	All
ICSE	Machine Learning-Based Detection of Open Source License Exceptions	Vendome et al.	D/JS	Descriptive	Human-System
ICSE	Feedback-Based Debugging	Lin et al.	D/LE	Solution	Human-System
EMSE	On the long-term use of visual GUI testing in industrial practice - a case study.	Algroth et al.	FS	Descriptive	Human
EMSE	Evaluating code complexity triggers, use of complexity measures and the influence of code complexity on maintenance time.	Antinyan et al.	SS	Descriptive	Human
EMSE	Reengineering legacy applications into software product lines - a systematic mapping.	Assunao et al.	Meta	Descriptive	Researcher
EMSE	User satisfaction and system success - an empirical exploration of user involvement in software development.	Bano et al.	FS/Meta	Descriptive	Human
EMSE	Extracting and analyzing time-series HCI data from screen-captured task videos.	Bao et al.	D/LE	Solution	System-Researcher
EMSE	The last line effect explained.	Beller et al.	D/SS	Descriptive	Human-System

Table 3: Examples of our paper classification and coding. FS: Field Study, D: Data Study, LE: Lab Experiment, JS: Judgment Study, FT: Formal Theory, SS: Sample Study.