Surfacing Paradigms underneath Research on Human and Social Aspects of Software Engineering

Jorge Melegati
Faculty of Computer Science
Free University of Bozen-Bolzano
Bolzano, Italy
jmelegatigoncalves@unibz.it

Xiaofeng Wang
Faculty of Computer Science
Free University of Bozen-Bolzano
Bolzano, Italy
xiaofeng.wang@unibz.it

***Abstract*—Software engineering is a wide field with topics ranging from coding to organizational aspects. In the last two decades, researchers have developed a growing interest in the human and social aspects of software development. To investigate these phenomena, researchers have often employed methods common in social sciences such as case study and ethnography. These methods usually have specific underpinnings about the relationship between knowledge and the world, the so-called research paradigms. Although these paradigms are essential to define what is legitimate knowledge and how research should be performed, so far, this topic has been barely discussed in the software engineering community. In this paper, our goal is to explore how different research paradigms reflected in the current studies of human and social factors in software engineering. To achieve this goal, we present an overview of the main research paradigms: positivism, postpositivism, constructivism-interpretivism, critical theory, and pragmatism. Then, we analyze the papers published in the Technical Track of the International Conference on Software Engineering (ICSE) that focus on human and social aspects to explore if and how these paradigms influence Software Engineering research. Our results show that these studies generally do not explicate the paradigms they are following but, from several aspects, it is possible to relate the studies to a pragmatic perspective with a strong postpositivistic influence. We also discuss topics on which research could be enriched by employing diverse paradigms.**

***Index Terms*—research paradigms, human factors, research methods**

I. Introduction

Computer science in general and software engineering in specific are essentially multidisciplinary. In the computer science field, we can observe such diverse research communities ranging from theoretical arguments as algorithms to organizational and human aspects like human-computer interaction. The rise of ethical and social implications of computing led Connolly [1] to argue Computer Science as a social science. In software engineering, so diverse elements as programming languages and development practices co-exist. For instance, in ACM's Computer Classification System[1], the high-order category *Software and its engineering* is divided into three categories: *software organization and properties*, *software notation and tools*, and *software creation and management*. While the second category concerns "hard" arguments like compilers and formal language definitions, the third deals with more human-related aspects such as development process management and development techniques.

Such a variety of arguments led to an increasing diversity of research methods and tools, especially the so-called qualitative methods used extensively in the social sciences. Software engineering researchers started to employ them to study human and social aspects of software development, as what we can observe in the pioneer paper of Seaman [2]. Since the publication of this paper in 1999, not only data collection and analysis methods Seaman mentioned but also entire research strategies have been employed in software engineering research, such as case studies [3] and grounded theory [4]. The publication of studies using these methods in major venues like *IEEE Transactions on Software Engineering* (e.g., [5]) and the *International Conference on Software Engineering* (e.g., [6]), along with the success of the Cooperative and Human Aspects of Software Engineering (CHASE) workshop (recently converted to a working conference), demonstrates how these studies gained visibility.

Nevertheless, researchers often employ these methods to study human and social aspects of software engineering without considering their underlying philosophical underpinnings as demonstrated by research methods usage analysis, e.g., [4] and [7]. In this regard, one neglected aspect in software engineering literature is about research paradigm. A research paradigm is a set of basic beliefs [8] about the social world providing "a philosophical and conceptual framework for the organized study of that world" [9]. Guba and Lincoln [8] identify three defining characteristics of different paradigms: ontology, epistemology, and methodology. These aspects can be mapped to the following questions [10]: what is the nature of reality, what is the relationship between researcher and that being researched, and what is the process of research. The authors analyzed the paradigms employed in social sciences, namely positivism, post-positivism, critical theory and related, and constructivism. In addition to those, pragmatism is also often employed in information systems research [11].

Given these philosophical assumptions, these paradigms define what valid knowledge and legitimate research [8] are. Besides that, the explicit adoption of a paradigm could guide researchers in adopting methods, avoiding the misuses identi-

[1]https://dl.acm.org/ccs

fied in the Software Engineering literature. Nevertheless, to the best of our knowledge, there is no study on if and how different research paradigms have been employed in the research of human and social aspects of software engineering.

To fill this gap, we analyzed the papers published in the International Conference on Software Engineering (ICSE) in the last three years that focus on human and social aspects of software engineering. Our results indicate a predominance of pragmatic elements, such as varied research methods, with an influence from a postpositivistic view of obtaining general knowledge and mitigating biases. To conclude, we list research problems on which investigation could be enriched by employing diverse paradigms.

The remaining of this paper is organized as follows: Section II briefly summarizes previous studies in software engineering literature about research methodology, while Section III presents a summary of research paradigms. Section IV explains the research method we employed and Section V, the results. In Section VI, we discuss our results and foresee research problems that different paradigms could enrich. Finally, Section VII concludes the paper.

II. Related work

Although, currently, empirical software engineering dominates the main venues, it was not always this way. We can, at least partially, attributed this change to some seminal work. For instance, we can mention the work from Basili and colleagues promoting the use of controlled experiments. In a seminal work [12], they write "any scientific theory must be: 1) falsifiable, 2) logically consistent, 3) at least as predictive as other competing theories, and 4) its predictions have been confirmed by observations during tests for falsification." In a similar vein, Kitchenham and colleagues [13] argued for an empirical-based software engineering inspired by the counterpart in medicine.

This push for empirical software engineering led to the use of a plethora of research methods. The lack of training from software engineering researchers in many applied research methods prompted the emergence of several methodological papers in Software Engineering venues. An early example of this movement was a series of papers on the Software Engineering Notes about surveys by Pfleeger and Kitchenham [14]. Since then, several methodological papers emerged, e.g., Runeson and Höst [3] for case studies, Passos et al. [15] for ethnographies, and Molleri et al. [16] for surveys. This issue led to a recent initiative of developing empirical standards of software engineering research [17]. Other papers mixed methodological guidelines with a critical view of how researchers have employed the methods, such as Stol and Fitzgerald [4] for grounded theory. In these reviews, the authors generally conclude that the methods are often not appropriately employed, e.g., in the cases of Grounded Theory [4] and Case Survey [7]. This misuse is present even for Case Study, as concluded in a recent essay from Wohlin [18], even though the method has been extensively applied, and a book on it was published explicitly in the context of software engineering [19].

Despite the many methodological papers, we are not aware of published articles about research paradigms in software engineering research. At most, we found a discussion about it in a methodological paper. In their paper on grounded theory [4], Stol and Fitzgerald examined the method's philosophical foundation. To do so, they briefly presented positivism and interpretivism and acknowledged that "many studies do not sit neatly in either paradigms." They recognized that it was easy to classify the grounded theory method as interpretivist because of its use of qualitative data but reminded that it was developed "due to a desire to build theories more rigorously and dispassionately by grounding them in objective reality." Then, they presented three variants of the method: the first based on Glaser and Strauss's original work, the Straussian version based on the proposal of Strauss and Corbin, and, finally, a constructivist variant from Charmaz.

It is interesting to observe that the Information Systems (IS) community, which is close to Software Engineering, has been discussing research paradigms for thirty years now. Orlikowski and Baroudi's early and seminal work [20] reviewed a sample of Information Systems studies published between 1983 and 1988. They concluded that there was only "a single set of philosophical assumptions" guiding these studies, that is, positivism. Then, they presented and advocated for the use of two other paradigms: interpretivism and critical. After ten years of the publication of this paper, Chen and Hirschheim [21] performed an analysis of a large number of Information Systems papers published in the period since Orlikowski and Baroudi's paper and concluded that, although interpretivist studies arose, the large majority (81%) still fit the positivist perspective. Some years later, Richardson and Robinson [22] reviewed this study to discuss the lack of studies based on critical theory.

III. Research paradigms

In this section, we describe the commonly employed research paradigms in the social sciences. We will base this discussion on the writings of Guba and Lincoln [8], [23] and Ponterotto [9]. The authors divide the paradigms into four categories: positivism, postpositivism, critical theory and similar, and constructivism-interpretivism. We also include pragmatism in this study, following Goldkuhl [11]. It is beyond this paper's scope to thoroughly define all these paradigms, and we invite interested readers to consult the original sources. Nevertheless, we believe a common ground is essential to our further discussion. Therefore, we present these paradigms' ontological, epistemological, and methodological perspectives. Additionally, we present some aspects that distinguish the paradigms from one another as summarized in Table I.

A. *Positivism*

From an ontological perspective, adopters of positivism believe that there is a unique, true reality. This perspective is called realism (or naive realism to contrast with more critical variations). From an epistemological perspective, this paradigm assumes a dual viewpoint where the investigator

TABLE I
SUMMARY OF RESEARCH PARADIGMS BASED ON ONTOLOGY, EPISTEMOLOGY, AND METHODOLOGY (BASED ON [8]).

Paradigm	Ontology	Epistemology	Methodology
Positivism	Realism (there is a "true" apprehensible reality).	Objective, separating object under study from the researcher.	Mainly experimental, controlling confounding factors to verify theories.
Postpositivism	Critical realism (there is a "true" reality but it is not apprehensible by a fallible human mind).	Objectivity is not reachable but a goal. Findings are probably true and are falsifiable.	Critical multiplism. More natural settings and possible use of qualitative methods.
Constructivism-Intepretivism	Relativism: the reality is built based on the interpretation from those that experience it. Multiple and equally valid realities.	Knowledge is built on the interaction between researcher and the study object.	Hermeneutical and dialectic. Based on the interaction between researchers and subjects. Importance of qualitative methods.
Critical theory and similar	A "true" reality exists but based on power relationships among different actors in the society.	Knowledge is based on interpretation but mediated by values.	Dialectic in nature. The researcher is a transformative actor and should inform the subjects about the power structures so they could fight against it.
Pragmatism	A "true" reality exists but some elements in the world are created by the mind of individuals.	Action on the world is responsible for knowledge creation.	Not constrained by methods. The researcher should use the method most suitable for the problem at hand.

and the investigated are separated entities. Researchers employ techniques to reduce bias and threats to validity, so the findings obtained are true. From a methodological perspective, hypotheses are formulated in a propositional form and tested through empirical tests where confounding factors are controlled.

B. Postpositivism

This paradigm emerged as an answer to the criticism of positivism. Advocates of postpositivism still believe that there is a single reality, but it is "only imperfectly apprehendable because of basically flawed human intellectual mechanisms and the fundamentally intractable nature of phenomena" [8], the so-called critical realism. From an epistemological perspective, dualism is considered impossible, but objectivity is the "ideal" [8]. Therefore, researchers employ safeguards to mitigate threats to this objectivity, such as fitness to previous knowledge and referral process [8]. However, a key distinction to positivism is that findings should be falsifiable rather than verified, as a clear reference to the work of the philosopher Karl Popper [9]. From a methodological perspective, adopters of this paradigm follow a "refurbished version of triangulation" to falsify rather than verify hypotheses [8]. They often collect data from natural settings as an answer to the criticism of lack of realism. Consequently, they started employing qualitative techniques. A key defining characteristic for both post and positivism is their goal of explanation for further prediction and control [9].

A typical example of this paradigm for qualitative studies is the case study method as described by Eisenhardt [24]. As observed by Welch et al. [25], according to Eisenhardt, the method is grounded on the aim for "the development of testable hypotheses and theory which are generalizable across settings" [24]. The excerpt from the paper of Basili and colleagues [12], mentioned at the beginning of Section II, where the authors claimed that theories should be falsifiable, logically consistent, have predictive power, and have been tested for falsification, is also a clear instance of this paradigm.

C. Constructivism and interpretivism

Following Guba and Lincoln [8] and Ponterotto [9], we will discuss constructivism and interpretivism together. The adopters of these two paradigms share "the goal of understanding the complex world of lived experience from the point of view of those who live it" and the belief that "to understand this world of meaning, one must interpret it" [26]. Nevertheless, they present a subtle difference: while "interpretivism seeks to build knowledge from understanding individuals' unique viewpoints," "constructivism views knowledge as constructed as people work to make sense of their experience" [27]. For a thorough comparison between the two paradigms, we invite interested readers to consult Schwandt [26].

From an ontological perspective, adopters of this paradigm argue that reality is built based on the interpretation and mental constructions of the actors involved. Therefore, there is no single reality, but there are many depending on these actors, and they are "equally valid" [9]. In extension, researchers are also subject to interpreting reality, so, from an epistemological perspective, there are no objective methods. This view can be traced back to the philosopher Immanuel Kant and his idea that "you cannot partition out an objective reality from the person (research participant) who is experiencing, processing, and labelling the reality" [9]. Findings are created based on the interaction between the researcher and the object being studied. A key aspect of this paradigm is the focus on understanding rather than explaining as in the (post-)positivism paradigm [26]. From a methodological perspective, it is adopted a hermeneutical and dialectical perspective where "individual constructions can be elicited and refined only through interaction *between* and *among* investigator and respondents" (emphasis from the original) [8]. As a consequence, qualitative research methods are predominant.

To exemplify this paradigm, we could take Charmaz' words when describing his Constructivist Grounded Theory [28, pp. 13], a method often employed in Software Engineering studies. She advocates for "the assumption that social reality is multiple, processual, and constructed" so "we must take the researcher's position, privileges, perspective, and interactions into account as an inherent part of the research reality."

D. Critical theory and related paradigms

The term critical theory is an umbrella for several related worldviews such as neo-Marxism, feminism, materialism [8], and queer theory [23]. The common ground among these paradigms is their notion that reality is built based on the influences of economic, political, or other power-related factors. From an ontological perspective, they accept a reality built by these power relationships approximating a post-positivist perspective. Nevertheless, from an epistemological perspective, they acknowledge the influence of the researcher's values on the findings. A key characteristic of this paradigm is regarding what Guba and Lincoln called *voice*. While (post-)positivists perform the role of "disinterested scientists," creating knowledge useful for other actors such as policymakers or change agents, and interpretivists seek knowledge for the sake of understanding, critical theorists are "transformative intellectuals" whose work should drive the change. This principle influences the methodology applied that is dialectic in nature. Besides building knowledge, it also informs the research targets so they could transform the structures.

E. Pragmatism

The key distinction between pragmatism and the other paradigms is its adopters focus on action. In their view, knowledge should be useful for action [11]. Goldkuhl [29] argue for the existence of three types of pragmatism: functional, referential, and methodological. These types are related to different relationships among knowledge and action. In the functional view, knowledge is for action, that is, it is concentrated on its usefulness for practice. In the referential view, knowledge should be about action, that is, researchers should focus their undertakings on "actors, actions, action-objects, activities, and practices." Finally, in the methodological view, knowledge is generated through action, that is, its development is based on a "continual interaction between knowing and acting."

From an ontological perspective, pragmatism accepts the idea of a unique reality as (post)positivism "but at the same time emphasizes reason and thought as originators of elements in the external world" [11]. From an epistemological perspective, knowledge is created by the action on the world, and it should be useful for intervention and change [11]. From a methodological perspective, the paradigm is not restricted to specific methods but advocates using the most appropriate method for the problem at hand.

A good example of pragmatism comes from Miles, Huberman, and Saldaña, authors of a seminal book of qualitative methods often cited in Software Engineering papers. In the fourth edition of the book [30], they write that they label themselves as "pragmatic realists." They say they "believe that social phenomena exist not only in mind but also in the world," but they agree with interpretivists "who point out that knowledge is socially constructed." Their goal is to make assertions and build theories "to account for a real world" because, in their opinion, "human relationships and societies have unique peculiarities and inconsistencies that make a realistic approach to understanding them more *complex* - but not *impossible*" (highlight from us).

IV. Research method

Given the lack of studies on different paradigms underneath Software Engineering research, we performed an exploratory literature review to start filling this gap. To guide our study, we proposed the following research question:

RQ: How are diverse paradigms reflected in the current research on human and social aspects of software engineering?

To reach our goal, we decided to analyze a sample of such studies. Initially, we inspected papers published in ICSE 2020 that focused on human and social aspects of software engineering. After this initial exploration, we observed that the number of papers to be analyzed could overwhelm our analysis capacity. Therefore we decided to limit the number of venues and years considered. Following what Baltes and Ralph [31] classify as a purposive sampling strategy, we purposefully selected the relevant papers presented in the Technical Track of ICSE in the last three years (2018-2020). We made this decision given the significance of ICSE in the software engineering research field, the diversity of papers accepted in the track, and the maturity these studies present when compared, for instance, with those from CHASE, a venue with more relevant papers to our study but often at an earlier stage of development. More mature studies will probably present all the aspects relevant to our analysis.

The selection of the papers to be analyzed consisted of obtaining the list of all documents published in the proceedings through the IEEE Xplore digital library. We removed those documents that were not research papers, such as tables of contents and organizers' messages. Then, we inspected the titles and abstracts according to the following inclusion (exclusion) criteria:

- research papers (excluding, for instance, table of contents and messages from organizers);
- focused on human or social aspects of software engineering, that is, studies should focus on phenomena happening on the practitioners' individual level or within human relationships occurring in the software engineering process;
- primary studies (excluding literature reviews and mapping studies); and
- original conference publications (removing journal-first studies).

In this step, we followed a conservative approach, and, in case of doubt, we considered the paper to be further analyzed. In the next step, we downloaded the papers' full-text and re-evaluated them using our inclusion and exclusion criteria. Our final set consisted of 27 papers (the complete reference list is at the end of the paper), where five were from 2018, nine from 2019, and 13 from 2020. Figure 1 summarizes this process.

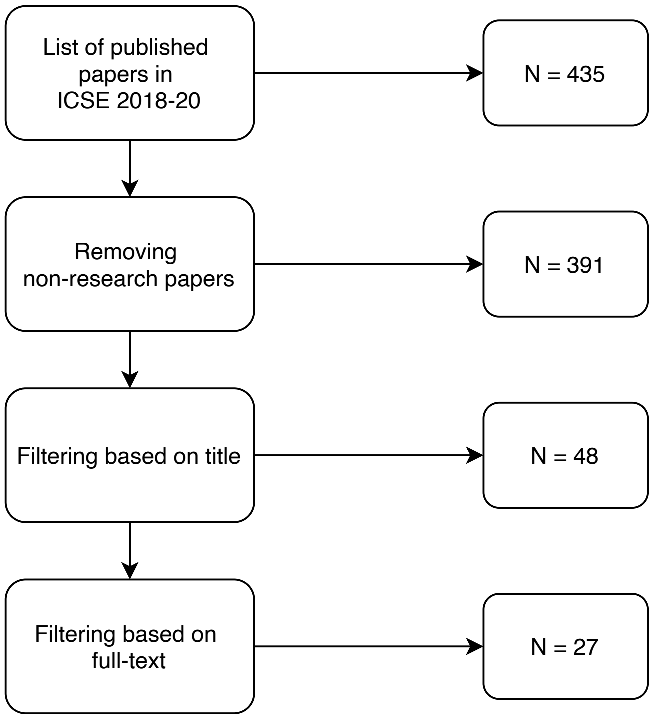
List of published papers in ICSE 2018-20
N = 435
Removing non-research papers
N = 391
Filtering based on title
N = 48
Filtering based on full-text
N = 27

Fig. 1. The paper selection process.

The analysis consisted of identifying, in the papers, elements from the paradigms through a process of coding [2]. Instead of using a pre-defined set of codes, we employed an integrated approach [32] where we defined an initial list of aspects to observe on the studies but without a set of specific codes. The codes emerged during the analysis, and, through constant comparison, they were merged, discarded, or new ones were created. Our initial list of aspects consisted of:

- Explicit claims: in this aspect, we observed if papers explicitly stated their adherence to a specific paradigm.
- Research questions: given the different ontology and epistemology perspectives, we might expect diverse goals presented in research questions. Given the objective of universal knowledge in (post)positivistic paradigms, researchers employing this paradigm focus on general research questions. Besides that, they could base their studies on hypotheses derived from previous theories. Meanwhile, for constructivist-interpretivist paradigms, there should be a preference for contextual research questions.
- Research methods: as discussed in the previous section, the ontological and epistemological stances from different paradigms lead to diverse methodological perspectives. (Post)positivists employ methods that isolate the researcher from the object of study, while the constructivist-interpretivist encourages interaction.
- Threats to validity: similar to research methods, depending on the paradigm, researchers have different concerns regarding their studies' validity.

Besides these points, we also extracted from the studies the aspects that were not related to the initial list, pointing to an evident influence of other paradigms. For instance, we can detect the influence of pragmatism if a study's goal is implementing change in the studied environment.

V. Results

Before presenting the coding results, it is essential to describe the topics addressed by the reviewed studies. Most of the analyzed papers (12) investigated collaboration aspects of software development. These elements ranged from intra-team matters such as code reviews to developer communities like those in open source software. Seven papers focused on aspects of developers' work, including coding proficiency [P1], security [P2] and programming rationale [P3]. We classified two papers under developers' emotions [P4] and biases [P5] because they regard human factors per se and not related to work practices. Another two papers focus on developers' training: programming language learning [P6] and tool adoption [P7]. Finally, one paper [P8] investigated the accessibility of apps. Table II summarizes the research topics investigated by the papers. Besides that, it is interesting to notice that a recurrent domain among the investigated problems was open source software: nine studies focused on this domain.

TABLE II
Research topics ordered by the number of occurrences.

Topic	Papers
Collaboration aspects	[P9] [P10] [P11] [P12] [P13] [P14] [P15] [P16] [P17] [P18] [P19] [P20] (12)
Developers' way of work	[P21] [P2] [P22] [P23] [P3] [P1] [P24] (7)
Gender issues	[P25] [P26] [P27] (3)
Developers' emotions and biases	[P5] [P4] (2)
Learning and tool adoption	[P6] [P7] (2)
Accessibility	[P8] (1)

In the following sub-sections, we present the results of our analysis organized by the list of aspects.

A. Explicit claims

In our final list of analyzed papers, none of them explicitly claimed to adhere to a specific paradigm. We only noticed one paper, which was excluded in the full-text analysis phase because it is focused on a development practice, not on human factors, that has an explicit claim related to the research paradigm it followed. Sedano et al. [6] performed a constructive grounded theory to understand the product backlog in a company. Besides explicitly claiming the use of the constructive version of the method, the authors also discussed aspects of validity common in this paradigm, like transferability.

B. Research questions

Regarding this aspect, the most common way to represent the research goal was using general research questions, that is, aimed to uncover a universal truth. These questions were present in 17 studies. Some examples are "What motivates software professionals to participate in technology meetups?" [P10] or "What types of cognitive biases do developers frequently experience?" [P5]. In four studies, instead of developing research questions, researchers employed previous theories and existing knowledge to develop hypotheses to be tested. For instance, Murphy-Hill and colleagues [P7] investigated if a practice of preparing physical newsletters to developers about new tools, the so-called Testing on the Toilet, increased their adoption of these tools. To guide their study, the authors used the following hypothesis: "Testing on the Toilet increases usage of advertised developer tools."

In only three studies ([P9], [P18], and [P24]), researchers employed questions focused on a specific context. In two studies ([P21] and [P19]), there were no explicit research questions, and the authors declared to be conducting an exploratory study. Finally, one study ([P25]) evaluated a method and, therefore, its research questions focused on the method.

C. Research methods

Regarding the research methods used, many of the analyzed papers (10) claimed to use a mixed-methods approach. For instance, Egelman and colleagues [P9] performed a survey with 1317 developers and analyzed code review logs to predict negative feelings about the process. To investigate the influence of news aggregators on developer communities, Aniche et al. [P18] employed interviews, survey, and content analysis.

Some studies used generic terms and did not make explicit claims on the research methods employed. Three studies claimed to perform an empirical study and another three a field study. Two studies used the term "qualitative study." One used "quantitative study." Another one claimed to conduct "a large scale study."

In contrast, some studies did declare to use a specific method. Two papers used the term "survey" as the research method. Another two used grounded theory, and with just one occurrence each, there was action research, case study, and controlled experiment. Table III summarizes these results.

TABLE III
RESEARCH METHODS AS CLAIMED IN THE PAPER ORDERED BY THE NUMBER OF OCCURRENCES.

Research methods	Papers
Mixed-methods study	[P9] [P10] [P2] [P12] [P6] [P23] [P16] [P7] [P17] [P18] (10)
Empirical study	[P4] [P8] [P1] (3)
Field study	[P5] [P3] [P20] (3)
Qualitative study	[P21] [P15] (2)
Grounded theory	[P13] [P22] (2)
Survey	[P14] [P26] (2)
Action research	[P25] (1)
Case study	[P19] (1)
Controlled experiment	[P11] (1)
Large scale study	[P24] (1)
Quantitative study	[P27] (1)

1) Research techniques: Since the overall methods did not give us much useful information to evaluate the worldviews that influence the studies, we analyzed the data collection and analysis techniques used in the studies. Since many studies, especially those using mixed-methods, utilize more than one, the sum of occurrences is above the number of studies. Regarding data collection, the most common technique, with 14 occurrences, was applying a questionnaire, which was often called a survey in the papers. We will not use this term to avoid confusion with the comprehensive research method. Then, in 13 studies, researchers performed interviews with subjects. In 13 studies, researchers collected artifacts to be further analyzed, including log data, e.g., [P9], or mining repositories like in [P12]. In five studies, researchers observed subjects in their working environments. For instance, Chattopadhyay et al. [P3] mentioned using the fly-on-the-wall technique. In two studies, [P2] and [P20], researchers asked subjects to perform some form of tasks. Fig. 2 summarizes these results.

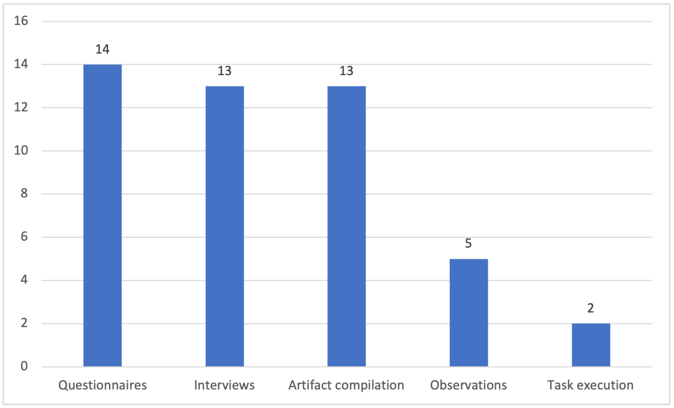

Fig. 2. Number of the occurrences of data collection techniques identified in the reviewed studies.

Regarding analysis techniques, the most common, present in 12 studies, was the use of some forms of statistical analysis, often performing statistical tests. In seven papers, researchers performed some kind of coding, including inductive coding, e.g., [P9], or thematic analysis, e.g., [P23]. Two studies, [P21] and [P1]) employed open card sorting. Fig. 3 summarizes these results.

D. Validity discussion

In the discussion of validity, almost all papers employed a perspective influenced by realism. The most common manner

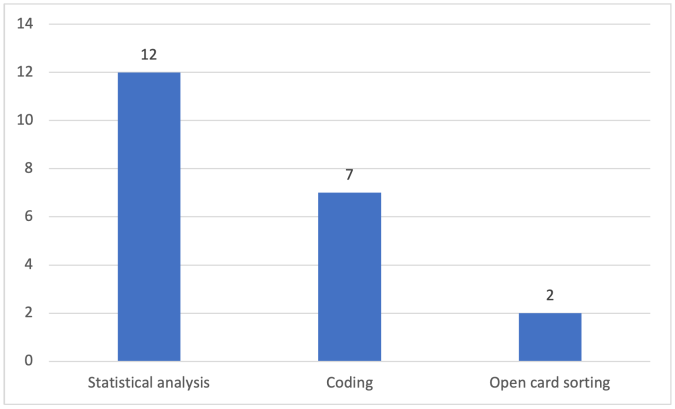

Fig. 3. Number of the occurrences of data analysis techniques identified in the reviewed studies.

(present in eight studies) was pinpointing threats to the triad consisted of construct, internal, and external validity. In other cases, authors only mentioned a subset like construct and external validity ([P24]) or external and internal validity ([P25]). In another study [P4], besides the triad, researchers also mentioned conclusion validity. Construct validity is related to the extent that the measurements employed by researchers represent the constructs they have in mind. Internal validity is related to causal inferences and represents if the research claims that a factor is determined by another when, in reality, a third factor, not considered in the study, is responsible. Finally, external validity is related to the applicability of the results to other contexts. In this sense, external validity is identical to the second most common aspect discussed, generalizability, considered in six studies. For instance, in a study about quasi-contributors on open source software, Steinmacher et al. [P17] argued that "when quantitatively analyzing data from the selected project," they "used statistical methods to mitigate the threats of generalizing data based on our personal hypothesis." In four studies, the authors' concern was to mitigate biases. In three studies, surveys were employed to generalize results.

Only one study, [P15], employed interpretivistic constructs to analyze threats to validity. The authors discussed credibility, transferability, dependability, and confirmability. Credibility is the counterpart of internal validity and is related to the results' reliability and conformance to the data collected. Transferability concerns the possibility of transfer the results to similar contexts and, as so, is a counterpart of external validity. Dependability is to which extent the results depend on the specific research and, as such, is similar to reliability. Finally, confirmability concerns if the results represent the view of participants and not the researchers'. Besides that, two other studies, [P5] and [P3], discussed why their results were transferable to other contexts, which is a common interpretivistic concern. Table IV summarizes these results.

TABLE IV
VALIDITY DISCUSSION PRESENT IN THE ANALYZED PAPERS ORDERED BY THE NUMBER OF OCCURRENCES.

	Validity discussion	Papers
Realism influences	Construct-External-Internal validity	[P21] [P2] [P11] [P14] [P1] [P26] [P27] [P19]
	Generalizability	[P9] [P10] [P21] [P5] [P17] [P18]
	Mitigate biases	[P10] [P5] [P8] [P3]
	Construct and External validity	[P24]
	External and Internal validity	[P25]
	Construct-External-Internal-Conclusion validity	[P4]
	Surveys to generalize results	[P22] [P17] [P18]
Interpretivism influences	Transferable results	[P5] [P3]
	Credibility-Transferability-Dependability-Confirmability	[P15]

E. Specific elements

In this section, we present aspects identified in the analyzed papers that, although not specifically related to elements described above, indicate a specific paradigm's influence.

Regarding (post)positivism, the only but recurrent aspect observed was the use of existent theories. For instance, in a study about the fairness perception of code reviews, German et al. [P19] employed fairness theory. Even in a grounded theory study, for which methodological papers suggest not to be biased by previous results [4], Danilova et al. [P22] based their study about security warnings to developers on previous work. This study also employed a survey to test the theory. In the paper's words: "we conducted a GT study based on Charmaz with theoretical sampling to get a broad overview of the problem space. We followed up this qualitative work with a quantitative survey to test the developed themes and our explanatory theory." It is interesting to notice that Charmaz's version of the method is of constructivist influence [4].

Although not so strong as for positivist elements, we also identified some interpretivist influences. For instance, three studies ([P15], [P26], and [P19]) focused on developers' perceptions on diverse aspects. For instance, to understand why code review worked for open source communities, Alami et al. [P15] wanted "to dig deeply into the mindset of participants in this inherently human process that relies heavily on communication skills; that involves feedback, critique and rejection on daily basis; and that exposes power and decision hierarchies in communities." Nevertheless, the common aspect, present in 16 studies, generally associated with this paradigm [9], is the use of subjects' quotes, most commonly from interviews, such as in [P9] and [P15], and also open-ended questions in questionnaires, as in [P2], or from other analyzed artifacts such as StackOverflow answers, as in [P21].

Regarding pragmatism, a common aspect is to show the usefulness of the results to practitioners. For instance, Alami et al. [P15] performed an action research study about code reviews in open source software projects, and "formulate 20 proposals for how what [the authors] know about hacker ethics and human and social aspects of code review, could be exploited to improve the effectiveness of the practice in software projects." In their action research study to foster gender-inclusiveness in software development teams, Hilderbrand et al. [P25] presented the teams "insights and experiences in the form of 9 practices, 2 potential pitfalls, and 2 open issues [...]."

Regarding the critical theory paradigm, the influence is much smaller. We could detect it in the motivation of one study about gender issues. Hilderbrand et al. [P25] started their paper with the following: "Software has repeatedly failed diverse populations, falling short of aiding their productivity or even being usable by some populations."

VI. Discussion

Our results show that analyzed papers do not explicitly state the research paradigms that influence them. This dearth could indicate the lack of discussion of this aspect among software engineering researchers, which corroborates the need for more studies discussing research paradigms in Software Engineering research. Clear adherence to a paradigm is essential for readers and reviewers to assess if the employed methods are adequate and the study's conclusions, trustworthy. Nevertheless, the elements identified in the studies reveal an implicit pragmatic worldview strongly influenced by a postpositivistic perspective. Several elements corroborate this conclusion. From the angle of research questions, it was clear a predominance of aiming universal truths. Similarly, regarding threats to validity, researchers aimed to mitigate issues that would prevent their results from being general. To do so, they also acted to prevent biases. From a methodological perspective, it is interesting to see the predominance of mixed-methods studies. These combinations could be a consequence of the desire to study more natural settings as one of the criticisms of positivism mitigated by post-positivism. Nevertheless, it could also be related to a pragmatic view of using the most adapted method to investigate the problem at hand. Still, another interpretation is the need for quantitative data, usually in the form of a survey or artifact analysis, to generalize results obtained from qualitative data. These aspects could also explain the frequency at which researchers employ interviews in their studies and use quotes from them (or other sources) to illustrate their reports.

Our findings are in line with the results Engstrom and colleagues [33] obtained when using the design science research (an essentially pragmatic method) lenses to evaluate ICSE papers. The authors observed that most of the papers fitted these lenses. This pragmatic tendency is visible through the three types proposed by Goldkuhl [29]. From a functional perspective, we could observe that most authors argued the value of their studies to real problems. In several studies, the focuses were on the actions performed by actors, generally software developers, indicating a referential view. Finally, the plethora of different research methods employed, including those generally associated with constructivist-interpretivist traditions such as grounded theory and interviews, point to a methodological pragmatic view.

We must admit that we were not surprised by the lack of papers employing constructivist-interpretivist or critical theory paradigms. Nevertheless, during our analysis, we saw some opportunities for researchers wanting to employ different paradigms. Our goal is not to advocate that these paradigms are superior for these topics when compared to post-positivism or pragmatism. Instead, we would like to call the attention of researchers, reviewers, program committee chairs, and editors on alternative possibilities to perform research that could enrich the Software Engineering body of knowledge.

Regarding the constructivist-interpretivist paradigm, the scarcity of studies following this paradigm indicates a need to encourage more such studies. In our analysis, it was clear the importance of predicting theories or useful results. Studies to deeply understand a particular phenomenon would be valuable to our field.

A clear possibility for critical theory is the emerging field of research on gender issues in software development. Currently, studies mainly described the problem using a more conventional, detached view. Researchers could take a critical view and start trying to change the status quo, for instance, acting towards increasing women's participation in software development and reducing the bias against them. Another interesting topic is ethics. Even though not found in our results, it is an increasing interest of our community that could take advantage of critical theory.

VII. Conclusions

Software Engineering is a multidisciplinary research field where diverse problems coexist, ranging from programming languages to developers' emotions. This diversity led to the employment of a wide range of research methods, including, especially for human and social aspects, some techniques borrowed from social sciences. Unfortunately, these methods are employed without proper consideration of the underlying philosophical assumptions. Previous work has shown that this phenomenon led to many methods being used, not adhering to their guidelines properly. To the best of our knowledge, no study in the Software Engineering literature has investigated the employment of different research paradigms. To start filling this gap, we analyzed the papers published in the Technical Track of ICSE in the last three years regarding several aspects influenced by research paradigms. Our results indicate that, although researchers do not explicitly claim the underlying research paradigms they follow, the predominant view is pragmatic by focusing on "real" problems and employing diverse methods, and with a strong influence of post-positivism represented by a focus on general results and mitigating threats to a study's validity.

Our results were limited to the last three years of ICSE, and further investigations could analyze other venues, especially journals, that could present a broader range of topics. Besides that, we restricted our study to published papers. Future work could survey or interview researchers to better understand why they performed the studies in a given way and whether they consider future studies with a diverse paradigm, and the reasons behind those choices.

We expect that this paper acts as a call for even more diverse research on human and social factors of Software Engineering. We discussed some possible topics that could profit from other paradigms, especially critical theory. From a meta-research perspective, a possible avenue for further analysis is the discussion of threats to validity. A section about this aspect is almost always present in research papers, but no relation is made to the underlying researchers' worldviews. From an educational perspective, we expect that more studies like ours could bring attention to research paradigms in the existing courses of empirical software engineering for graduate students.

ANALYZED PAPERS

[P1] X. Xia, Z. Wan, P. S. Kochhar, and D. Lo, "How Practitioners Perceive Coding Proficiency," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, vol. 2019-May. IEEE, 2019, pp. 924–935.

[P2] D. V. D. Linden, P. Anthonysamy, B. Nuseibeh, T. T. Tun, M. Petre, M. Levine, J. Towse, and A. Rashid, "Schrodinger's security: Opening the box on app developers' security rationale," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 149–160.

[P3] S. Chattopadhyay, N. Nelson, Y. Ramirez Gonzalez, A. Amelia Leon, R. Pandita, and A. Sarma, "Latent Patterns in Activities: A Field Study of How Developers Manage Context," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, vol. 2019-May. IEEE, 2019, pp. 373–383.

[P4] D. Girardi, N. Novielli, D. Fucci, and F. Lanubile, "Recognizing developers' emotions while programming," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. New York, NY, USA: ACM, 2020, pp. 666–677.

[P5] S. Chattopadhyay, N. Nelson, A. Au, N. Morales, C. Sanchez, R. Pandita, and A. Sarma1, "A tale from the trenches: Cognitive biases and software development," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 75–86.

[P6] N. Shrestha, C. Botta, T. Barik, and C. Parnin, "Here we go again," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. New York, NY, USA: ACM, 2020, pp. 691–701.

[P7] E. Murphy-Hill, E. K. Smith, C. Sadowski, C. Jaspan, C. Winter, M. Jorde, A. Knight, A. Trenk, and S. Gross, "Do Developers Discover New Tools On The Toilet?" in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, vol. 2019-May. IEEE, 2019, pp. 465–475.

[P8] A. Alshayban, I. Ahmed, and S. Malek, "Accessibility issues in Android apps," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. New York, NY, USA: ACM, 2020, pp. 1323–1334.

[P9] C. D. Egelman, E. Murphy-Hill, E. Kammer, M. M. Hodges, C. Green, C. Jaspan, and J. Lin, "Predicting developers' negative feelings about code review," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 174–185.

[P10] C. Ingram and A. Drachen, "How sofware practitioners use informal local meetups to share sofware engineering knowledge," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 161–173.

[P11] D. Spadini, G. Çalikli, and A. Bacchelli, "Primers or reminders?" in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. New York, NY, USA: ACM, 2020, pp. 1171–1182.

[P12] S. Zhou, B. Vasilescu, and C. Kästner, "How has forking changed in the last 20 years?" in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Companion Proceedings*. New York, NY, USA: ACM, 2020, pp. 268–269.

[P13] F. Zieris and L. Prechelt, "Explaining pair programming session dynamics from knowledge gaps," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. New York, NY, USA: ACM, 2020, pp. 421–432.

[P14] A. Barcomb, K.-J. Stol, D. Riehle, and B. Fitzgerald, "Why Do Episodic Volunteers Stay in FLOSS Communities?" in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, vol. 2019-May. IEEE, 2019, pp. 948–959.

[P15] A. Alami, M. Leavitt Cohn, and A. Wasowski, "Why Does Code Review Work for Open Source Software Communities?" in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, vol. 2019-May. IEEE, 2019, pp. 1073–1083.

[P16] H. S. Qiu, A. Nolte, A. Brown, A. Serebrenik, and B. Vasilescu, "Going Farther Together: The Impact of Social Capital on Sustained Participation in Open Source," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, vol. 2019-May. IEEE, 2019, pp. 688–699.

[P17] I. Steinmacher, G. Pinto, I. S. Wiese, and M. A. Gerosa, "Almost There: A Study on Quasi-Contributors in Open Source Software Projects," in *Proceedings of the 40th International Conference on Software Engineering*. New York, NY, USA: ACM, 2018, pp. 256–266.

[P18] M. Aniche, C. Treude, I. Steinmacher, I. Wiese, G. Pinto, M.-A. Storey, and M. A. Gerosa, "How modern news aggregators help development communities shape and share knowledge," in *Proceedings of the 40th International Conference on Software Engineering*. New York, NY, USA: ACM, 2018, pp. 499–510.

[P19] D. M. German, G. Robles, G. Poo-Caamaño, X. Yang, H. Iida, and K. Inoue, ""Was my contribution fairly reviewed?"," in *Proceedings of the 40th International Conference on Software Engineering*. New York, NY, USA: ACM, 2018, pp. 523–534.

[P20] C. Mendez, H. S. Padala, Z. Steine-Hanson, C. Hilderbrand, A. Horvath, C. Hill, L. Simpson, N. Patil, A. Sarma, and M. Burnett, "Open source barriers to entry, revisited," in *Proceedings of the 40th International Conference on Software Engineering*. New York, NY, USA: ACM, 2018, pp. 1004–1015.

[P21] M. Lamothe and W. Shang, "When apis are intentionally bypassed: An exploratory study of apiworkarounds," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 2020, pp. 921–924.

[P22] A. Danilova, A. Naiakshina, and M. Smith, "One size does not fit all," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. New York, NY, USA: ACM, 2020, pp. 136–148.

[P23] F. Sarker, B. Vasilescu, K. Blincoe, and V. Filkov, "Socio-Technical Work-Rate Increase Associates With Changes in Work Patterns in Online Projects," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 2019, pp. 936–947.

[P24] M. Claes, M. V. Mäntylä, M. Kuutila, and B. Adams, "Do programmers work at night or during the weekend?" in *Proceedings of the 40th International Conference on Software Engineering*. New York, NY, USA: ACM, 2018, pp. 705–715.

[P25] C. Hilderbrand, C. Perdriau, L. Letaw, J. Emard, Z. Steine-Hanson, M. Burnett, and A. Sarma, "Engineering gender-inclusivity into software," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*. New York, NY, USA: ACM, 2020, pp. 433–444.

[P26] A. Lee and J. C. Carver, "FLOSS Participants' Perceptions About Gender and Inclusiveness: A Survey," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, vol. 2019-May. IEEE, 2019, pp. 677–687.

[P27] N. Imtiaz, J. Middleton, J. Chakraborty, N. Robson, G. Bai, and E. Murphy-Hill, "Investigating the Effects of Gender Bias on GitHub," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, vol. 2019-May. IEEE, 2019, pp. 700–711.

REFERENCES

[1] R. Connolly, "Why computing belongs within the social sciences," *Communications of the ACM*, vol. 63, no. 8, pp. 54–59, jul 2020.

[2] C. B. Seaman, "Qualitative methods in empirical studies of software engineering," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 557–572, 1999.

[3] P. Runeson and M. Höst, "Guidelines for conducting and reporting case study research in software engineering," *Empirical Software Engineering*, vol. 14, no. 2, pp. 131–164, apr 2009.

[4] K.-J. Stol, P. Ralph, and B. Fitzgerald, "Grounded theory in software engineering research," *Proceedings of the 38th International Conference on Software Engineering - ICSE '16*, no. Aug, pp. 120–131, 2016.

[5] C. Giardino, N. Paternoster, M. Unterkalmsteiner, T. Gorschek, and P. Abrahamsson, "Software Development in Startup Companies: The Greenfield Startup Model," *IEEE Transactions on Software Engineering*, vol. 42, no. 6, pp. 585–604, jun 2016.

[6] T. Sedano, P. Ralph, and C. Peraire, "The Product Backlog," in *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, no. Section IV. IEEE, may 2019, pp. 200–211.

[7] J. Melegati and X. Wang, "Case Survey Studies in Software Engineering Research," in *Proceedings of the 14th ACM / IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. New York, NY, USA: ACM, oct 2020, pp. 1–12.

[8] E. G. Guba and Y. S. Lincoln, "Competing paradigms in qualitative research," in *Handbook of Qualitative Research*, 1994, pp. 163–194.

[9] J. G. Ponterotto, "Qualitative research in counseling psychology: A primer on research paradigms and philosophy of science," *Journal of Counseling Psychology*, vol. 52, no. 2, pp. 126–136, 2005.

[10] J. Creswell, *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*. SAGE Publications, 2012.

[11] G. Goldkuhl, "Pragmatism vs interpretivism in qualitative information systems research," *European Journal of Information Systems*, vol. 21, no. 2, pp. 135–146, 2012.

[12] V. R. Basili, F. Shull, and F. Lanubile, "Building knowledge through families of experiments," *IEEE Transactions on Software Engineering*, vol. 25, no. 4, pp. 456–473, 1999.

[13] B. Kitchenham, T. Dyba, and M. Jorgensen, "Evidence-based software engineering," in *Proceedings. 26th International Conference on Software Engineering*. IEEE Comput. Soc, 2004, pp. 273–281.

[14] S. L. Pfleeger and B. A. Kitchenham, "Principles of Survey Research Part 1: Turning Lemons into Lemonade," *ACM SIGSOFT Software Engineering Notes*, vol. 26, no. 6, pp. 16–18, 2001.

[15] C. Passos, D. S. Cruzes, T. Dybå, and M. Mendonça, "Challenges of applying ethnography to study software practices," *International Symposium on Empirical Software Engineering and Measurement*, no. Dcc, pp. 9–18, 2012.

[16] J. S. Molleri, K. Petersen, and E. Mendes, "Survey Guidelines in Software Engineering," *Proceedings of the 10th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '16*, pp. 1–6, 2016.

[17] P. Ralph *et al.*, "Empirical standards for software engineering research," 2021.

[18] C. Wohlin, "Case study research in software engineering—it is a case, and it is a study, but is it a case study?" *Information and Software Technology*, vol. 133, p. 106514, 2021.

[19] P. Runeson, M. Höst, A. Rainer, and B. Regnell, *Case Study Research in Software Engineering: Guidelines and Examples*, H. Baumeister, H. Lichter, and M. Riebisch, Eds. Hoboken, NJ, USA: John Wiley & Sons, Inc., mar 2012, vol. 283.

[20] W. J. Orlikowski and J. J. Baroudi, "Studying information technology in organizations: Research approaches and assumptions," *Information Systems Research*, vol. 2, no. 1, pp. 1–28, 1991.

[21] W. Chen and R. Hirschheim, "A paradigmatic and methodological examination of information systems research from 1991 to 2001," *Information Systems Journal*, vol. 14, no. 3, pp. 197–235, jul 2004.

[22] H. Richardson and B. Robinson, "The mysterious case of the missing paradigm: A review of critical information systems research 1991-2001," *Information Systems Journal*, vol. 17, no. 3, pp. 251–270, 2007.

[23] Y. S. Lincoln and E. G. Guba, "Paradigmatic controversies, contradictions and emerging confluences," *Handbook of Qualitative Research, 2nd ed*, pp. 163–189, 2000.

[24] K. M. Eisenhardt, "Building theories from Case Study Research," *The Academy of Management Review*, vol. 14, no. 4, pp. 532–550, 1989.

[25] C. Welch, R. Piekkari, E. Plakoyiannaki, and E. Paavilainen-Mäntymäki, "Theorising from case studies: Towards a pluralist future for international business research," *Journal of International Business Studies*, vol. 42, no. 5, pp. 740–762, 2011.

[26] T. Schwandt, "Constructivist, Interpretivist approaches to human inquiry," in *The Landscape of Qualitative Research : Theories and Issues*, N. K. Denzin and Y. S. Lincoln, Eds., 1998, pp. 221–259.

[27] L. E. Tomaszewski, J. Zarestky, and E. Gonzalez, "Planning qualitative research: Design and decision making for new researchers," *International Journal of Qualitative Methods*, vol. 19, pp. 1–7, 2020.

[28] K. Charmaz, *Constructing Grounded Theory*, ser. Introducing Qualitative Methods series. SAGE Publications, 2014.

[29] G. Goldkuhl, "What Kind of Pragmatism in Information Systems Research?" *AIS SIG PRAG INAUGURAL meeting*, no. 1986, pp. 1–6, 2008.

[30] M. Miles, A. Huberman, and J. Saldana, *Qualitative Data Analysis: A Methods Sourcebook*. SAGE Publications, 2019.

[31] S. Baltes and P. Ralph, "Sampling in software engineering research: A critical review and guidelines," 2020.

[32] D. S. Cruzes and T. Dyba, "Recommended Steps for Thematic Synthesis in Software Engineering," *2011 International Symposium on Empirical Software Engineering and Measurement*, no. 7491, pp. 275–284, 2011.

[33] E. Engström, M.-A. Storey, P. Runeson, M. Höst, and M. T. Baldassarre, "How software engineering research aligns with design science: a review," *Empirical Software Engineering*, vol. 25, no. 4, pp. 2630–2660, jul 2020.