Available online at www.sciencedirect.com

ScienceDirect

Journal of Systems and Software 00 (2025) 1–46

JSS

The Qualitative Factor in Software Testing: A Systematic Mapping Study of Qualitative Methods

Baris Ardic[a,*], Carolin Brandt[a], Ali Khatami[a], Mark Swillus[a], Andy Zaidman[a]

[a]*Delft University of Technology*

Abstract

Software testing research has provided metrics on efficiency, error rates, and insights into the effectiveness of testing methodologies and tools. However, these tell only a part of the story. The qualitative dimension, which studies experiences, perceptions, and decision-making processes is crucial, but less prevalent in literature. This study aims to systematically map qualitative research in software testing to consolidate and categorize the methodologies used in qualitative testing research, highlight their importance, and identify patterns, gaps, and future directions. We conducted a systematic mapping study, identifying and analyzing 102 primary studies from 2003 to 2023. We categorized the studies according to research strategies, data collection, and data analysis methods. We identified case studies and grounded theory as the most prevalent research strategies. Researchers primarily used semi-structured interviews and thematic analysis to understand how practitioners work and gather stakeholder perspectives. The subject areas most covered by qualitative studies included software testing processes and risks, and test automation. Areas such as test oracles, and machine learning were underrepresented. We also assessed the quality of reporting and the methodological rigor, emphasizing the challenges and limitations identified during the process. Through this study, we provide a comprehensive overview of qualitative research practices in software testing, revealing trends, gaps, and methodological insights.

Keywords: systematic mapping, software testing, qualitative analysis, human factors, qualitative methodology

1. Introduction

Quantitative research has long been the backbone of insights in software testing, offering metrics on efficiency, error rates, and other tangible outcomes [1, 2]. However, this only tells part of the story. The qualitative dimension of software testing encompassing the experiences, perceptions, and decision-making processes of individuals and teams is equally crucial [3, 4]. These aspects have been less prevalent in the academic literature on software testing [5], but qualitative methods have been used to address human-involved problems in software engineering with increasing interest [6, 7, 8]. Qualitative studies explore these areas, providing depth and context that numeric data cannot capture [9, 6].

We chose systematic mapping as our approach because it allows a broad overview of the field, allowing us to highlight the wide array of methods and practices from different qualitative studies and mixed-method studies with an emphasis on qualitative aspects. Our goal is to strengthen qualitative research in software testing. In order to

*Corresponding author at: Delft University of Technology, Postbus 5 2600AA, Delft, Netherlands

Email addresses: b.ardic@tudelft.nl (Baris Ardic), c.e.brandt@tudelft.nl (Carolin Brandt), s.khatami@tudelft.nl (Ali Khatami), m.swillus@tudelft.nl (Mark Swillus), a.e.zaidman@tudelft.nl (Andy Zaidman)

do so, we first highlight the importance of qualitative research in the field while seeking to systematize knowledge, facilitate methodological refinement, identify patterns, gaps, and future directions. Furthermore, we intend to create a comprehensive overview for researchers, whether they are currently engaged in or aspiring to conduct qualitative research in this domain. This will involve an exploration of the methodologies and the common resources that emerge from qualitative studies in software testing. We present a holistic view of the qualitative research practices in software testing, enriching our understanding not only of the outcomes but also of the methodologies and reporting techniques that shape this field. This endeavor also presents an opportunity to learn and reflect on our methodologies and reporting practices, thus contributing to the advancement of qualitative research practice in software testing.

1.1. Research Questions

In this systematic mapping study, we are interested in including any study that is in the larger area of software testing where the main research method is qualitative. The studies we aim to examine do not have common research subjects; instead, their similarities lie in their methodologies. Therefore, the research questions we investigate are not concerned with the findings of these qualitative studies. We have the following research questions for this study:

- **RQ1:** How are qualitative methods utilized in software testing research?

  This overarching question aims to explore the use of qualitative methods in software testing research by addressing the following sub-questions:

  - **RQ1.1:** What are the main qualitative methods used in software testing studies?

    To identify current research trends and methodological preferences in software testing, we analyze the most widely used qualitative methods. This allows researchers who are acquiring an interest in this domain to better understand the methods for conducting qualitative research.
  - **RQ1.2:** What are the purposes of qualitative studies in software testing?

    To understand why researchers opted for qualitative approaches, we examine the purposes of qualitative studies. Knowing the use cases for these methods should help clarify how qualitative methods contribute to deeper insights for software testing research.

- **RQ2:** Are there any common resources used by the qualitative studies of software testing?

  To better understand the research methods employed within qualitative studies of software testing, we identify the prevalent resources used by qualitative studies to explore research method literature, tools, and guidelines. This helps guide future researchers in designing their studies with a solid theoretical and methodological foundation, while also highlighting potential shortcomings in foundational support within the literature.

- **RQ3:** What is the distribution of qualitative research across different areas of software testing?

  This overarching question aims to map the application of qualitative methods across various areas of software testing, identifying both the most frequently explored domains and the areas where qualitative research remains limited. It encompasses the following sub-questions:

  - **RQ3.1:** Which areas of software testing use qualitative methods the most?

    To reveal the subjects that have received the most attention, we determine the areas of software testing with more qualitative work. This extends our mapping efforts by covering both methods and subject areas.
  - **RQ3.2:** Which areas of software testing use qualitative methods the least?

    To highlight potential knowledge gaps, we identify which areas of software testing have the least qualitative research conducted. This directs researchers to under-explored domains where qualitative insights can provide valuable contributions, especially in connecting human factors with the technological aspects of software testing.

We want to clarify that our intention is not to conduct a detailed critique of individual papers included in our study or to criticize their authors in any way. Instead, we focus on creating an overall view of studies in which qualitative methods are the main driver of research. The main objective of this study is to determine what exists in this picture and

to use this to consider the opportunities we see to strengthen software testing research with a qualitative perspective. We prepare an agenda for future research based on the direction software engineering is taking and the results of our analysis. In addition, in our discussion, we also explore various suggestions for improving the existing methods of research from method-specific observations to challenges in reporting.

To our knowledge, this paper is the first systematic mapping of overall qualitative research methods used in software testing. The main contributions of this paper are:

- A comprehensive categorization of qualitative methods used in software testing studies by linking research strategies to data collection and analysis methods.
- A systematic mapping study that organizes related research work in qualitative software testing, capturing and analyzing the methodologies and application areas of included studies.
- An identification of prevalent trends and patterns, highlighting areas where qualitative methods are most and least applied.
- An evidence-based research agenda for future work using qualitative methods in software testing.

This paper proceeds as follows. Section 2 presents the background information that contextualizes our research on the use of qualitative methods and the use of mapping studies in software testing. Section 3 details the methodology we employed to conduct our systematic mapping study for the search and selection of studies. Section 4 reports on the data we have synthesized, highlighting key findings and trends. Section 5 offers a discussion of these results, interpreting their implications for the field of software testing. Section 6 addresses potential threats to the validity of our study, ensuring a transparent and rigorous approach. Finally, Section 7 concludes with a summary of our findings and suggestions for future research directions.

2. Background and Related Work

This section introduces relevant background knowledge on qualitative methods in software engineering. We also discuss literature mapping studies in general and in software testing.

2.1. Qualitative Methods

Qualitative methods cover a range of approaches that all focus on interpreting qualitative data. This type of data is descriptive and is typically found as “text” and “pictures” rather than as numerics [10, 11]. The unstructured nature of this material can offer unique insights for research on questions involving “why” and “how” [12]. This characteristic of qualitative data makes qualitative methods ideal for interpreting nuanced human behaviors and experiences.

Qualitative research methods were developed primarily by educational researchers and other social sciences to investigate human behavior since the complexities of such phenomena are not adequately expressed with quantitative methods [13]. Qualitative methods allow the study of social and cultural contexts by enabling researchers to explore the complexities of a problem rather than abstracting them [7, 14]. Popularized by social sciences, qualitative methods are widely used today in a multitude of fields, from psychology [15] to marketing research [16]. They are also finding their way to software engineering [6]. This is not surprising, as software engineering is a complex group activity that addresses multiple human and technical aspects [17]. Qualitative methods allow researchers to learn participants’ viewpoints without enforcing their interpretations [18], which is useful for software engineering contexts.

This provides the researcher with subjective meanings, behaviors, and social contexts as perceived by participants [12]. Specialized knowledge of the reality of software engineering is crucial for understanding and studying software engineering problems; therefore, practitioners can provide researchers with insights they may not be able to deduce on their own. Qualitative methods can also be employed to examine artifacts that are not traditionally treated as primary informational sources. For software engineering contexts, these could be online community posts [4], job advertisements [19] or even education [20] and training [21] related artifacts.

2.2. Mapping Studies

Systematic mapping studies are conducted to provide an overview of a research area through the classification of studies [22]. They start with a systematic search of the literature to detect the topics that were covered. Systematic mapping studies (SMS) are similar to systematic literature studies (SLR) with regard to the identification and selection of studies [23]. The differences emerge from the goals of these approaches. Literature review studies aim to answer research questions and provide evidence on a specific topic by critically synthesizing the findings of selected studies. Mapping studies explore a research field over a variety of topics and methodologies. The purpose of mapping studies is to highlight the distribution of effort across a subject to help visualize the landscape of broader research in a research field [24].

Petersen et al. [22] emphasize that mapping studies in software engineering systematically identify and classify research. This establishes a basis for more detailed investigations and highlights areas that require further attention. Similarly, Kitchenham and Charters [23] stress that mapping studies offer a structured method of synthesizing evidence, which is vital for informing future practice.

Table 1. Overview of Systematic Mapping Studies in Software Testing

Study	Focus / Subdomain	# Papers	Method and Key Findings
Engström and Runeson [25]	Software product line testing	64	Surveyed and categorized existing research. Found most papers proposed solutions, with system testing as the primary focus. Highlighted a need for more validation and evaluation studies to strengthen the field.
Catal and Mishra [26]	Test case prioritization in regression testing	120	Performed a systematic mapping of prioritization techniques, revealing coverage-based methods as the most prevalent. Suggested leveraging public datasets, exploring model-based approaches, and utilizing industrial datasets to address real-world challenges.
Garousi et al. [27]	Web application testing	79	Analyzed trends and practices in web testing. Found increased interest in client-side testing and identified critical gaps, such as the oracle problem and the absence of standardized validation repositories.
Durelli et al. [28]	Machine learning in software testing	48	Reviewed studies applying ML to software testing. Found ML was predominantly used for test generation and evaluation. Highlighted the need for further empirical evidence to validate ML effectiveness in testing tasks.
Salahirad et al. [29]	Evolution of software testing	~50K	Conducted a co-word analysis of author-provided keywords, uncovering dominant themes such as test automation and creation. Highlighted emerging focus areas including machine learning, web and mobile applications, among others. Developed a comprehensive map of software testing subdomains.
Our Study	Qualitative methods in software testing	102	Performed a systematic mapping of qualitatively focused research in software testing. Identified key gaps in the application of qualitative methods and offered insights for their improved use, along with a research agenda to guide future use qualitative methodologies in software testing.

2.3. Mapping Studies in Software Testing

There are a variety of systematic mapping studies that have been conducted in software testing. Among the multiple methods and application domains that exist in software testing, current mapping studies in software testing often focus on specific subdomains in testing [25, 26, 27, 28]. A systematic mapping study on software product line testing by Engström and Runeson [25] surveyed existing research to identify useful approaches and future research needs. They classified 64 papers, finding that most were solution proposals (a novel or significant extension to an existing technique), while system testing was found to be the largest focus area. They concluded that more validation and evaluation research is necessary to enhance the foundation of software product line testing.

Catal and Mishra et al. conducted a study that investigates test case prioritization techniques in regression testing [26]. They examined 120 relevant papers and found coverage-based prioritization techniques to be the dominant trend. The study recommends focusing on public datasets, developing more model-based methods, comparing prioritization methods, evaluation techniques, and using industrial project datasets to represent real-world problems.

Garousi et al. examined the field of web application testing [27], emphasizing the importance of the area due to increasing reliance on the web. The researchers classified and analyzed 79 papers selected from the literature to provide an overview of trends and emerging practices. They categorized publications by types of papers, sources for deriving test cases, and evaluation methods. Their findings revealed trends such as increased interest in client-side testing and areas that need further research, such as the oracle problem or lack of standard validation repositories.

In addition to mapping studies conducted to gain a broad perspective of an established subtopic, we also see studies done on general software testing [28, 29]. For example, Durelle et al. have carried out a mapping study on the

application of machine learning in automating software testing [28]. They analyzed 48 primary studies classified by study type, testing activity, and the ML algorithm used. Machine learning algorithms were found to be predominantly utilized for test case generation, refinement, and evaluation. They highlight the most commonly used ML algorithms in software testing and the need for more empirical studies to investigate the effectiveness of machine learning in this context.

Salahirad et al. investigated the evolution of software testing with a mapping study [29]. They have used co-word analysis of author-provided keywords to map out software testing research. They have found that the most popular keywords relate to test automation and creation. Additionally, emerging keywords were related to web and mobile applications, machine learning, and energy consumption.

Our study is similar to the studies by Salahirad et al. [29] and Durelli et al. [28], as it is concerned with the general domain of software testing. However, instead of an emerging technology or evolution, we investigate the use of qualitative methods regarding the overall body of work done in software testing. Table 1 provides an overview of the focus areas and key findings of our study alongside the software testing mapping studies discussed in this section.

3. Methodology

In this section, we introduce the methodology we follow during our systematic mapping study. Utilizing the guidelines by Petersen et al. [22] we first define our search strategy, together with inclusion and exclusion criteria. We outline our collaborative process, involving four researchers, for analyzing the identified primary studies and describe the data analysis approach used to address our research questions.

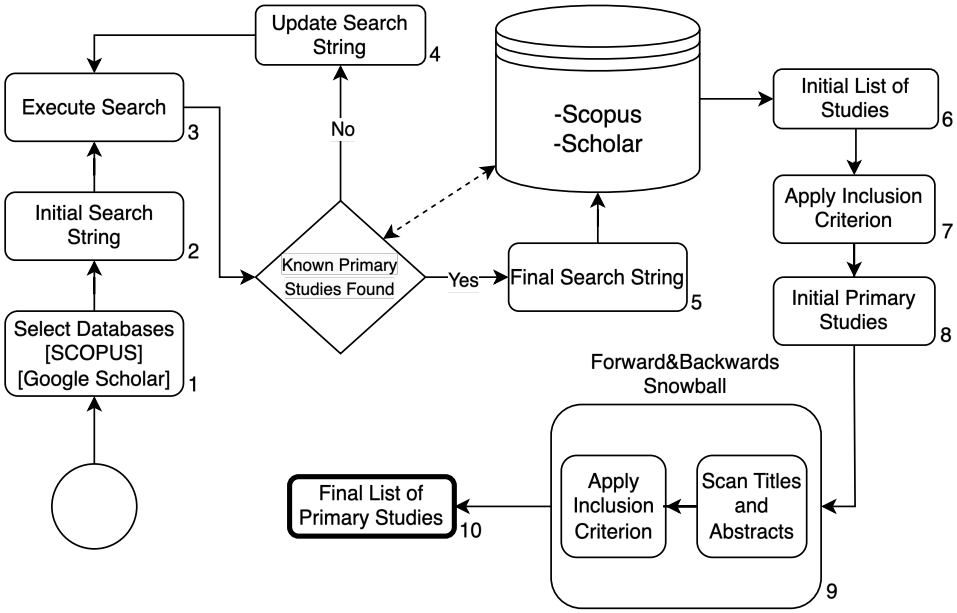

Figure 1. Overview of the screening process

3.1. Study Search and Selection

We selected SCOPUS [30] from Elsevier as our search database due to its comprehensive indexing of major digital libraries pertinent to software engineering research, including IEEE Xplore and the ACM Digital Library [31, 32]. Furthermore, it has been demonstrated to offer extensive article coverage when used together with Google Scholar and snowballing techniques [33]. Therefore, we also use Google Scholar to supplement SCOPUS. Database selection is demonstrated as Step 1 in Figure 1.

Our study search and selection process consists of four main parts: (1) creating the search string, (2) searching on SCOPUS, (3) performing a search on Google Scholar, and (4) snowballing. The detailed process is illustrated in Figure 1.

Our search strategy was not confined to specific sub-areas of software testing. Instead, to encapsulate a broad spectrum of qualitative methods, our initial search utilized the terms 'qualitative analysis' and 'software testing' shown as Step 2 in Figure 1. We execute a search with the initial search string and proceed to update our search string by incorporating relevant keywords from studies that were detected in the initial search which pass our inclusion criteria. We added keywords such as "qualitative data analysis", "qualitative study", "experience report", "interview study", "semi-structured interviews", "mixed method" stopping when a previously known set of primary studies [34, 35, 36, 37] were successfully detected. This process refers to steps 3 to 5 in Figure 1.

The final version of our search string is as follows, where the command "TITLE-ABS-KEY" refers to a search in any matches in titles, abstracts, or keywords of an article:

TITLE-ABS-KEY("Software Testing") AND TITLE-ABS-KEY("qualitative analysis" OR "qualitative data analysis" OR "qualitative study" OR "quantitative and qualitative analysis" OR "Interview Study" OR "Semi structured interviews" OR "Mixed method" We queried SCOPUS in August 2023 using these search terms, which yielded about 300 articles. After applying the inclusion and exclusion criteria outlined below, we identified 47 primary studies from this collection.

The search process on Google Scholar differed slightly, as it generates an extensive list of results ranked by relevance. We used the same search query described earlier but conducted the search within article titles and content, as Google Scholar functions differently from SCOPUS. Each page displayed 10 results, ordered by relevance.

For each search result, we verified whether the results are:

1. Articles (excluding books, theses, etc., included by the platform)
2. Not already identified through the SCOPUS query
3. Relevant to our study (i.e., related to software testing and containing qualitative work)

We continued reviewing results until encountering a complete Google Scholar result page with no potential primary studies, which occurred on page 15. After applying our inclusion and exclusion criteria, and removing duplicates already identified in SCOPUS, we retained 15 primary studies from Google Scholar.

To manage the filtering process, we used data sheets to apply the inclusion and exclusion criteria systematically. These data sheets were then merged to create an initial list of 62 primary studies, comprising 47 from SCOPUS and 15 from Google Scholar. This process is depicted in Steps 6–7 of Figure 1.

To further expand and refine the list of initial primary studies, we used both forward and backward snowballing methods. Backward snowballing involved examining the reference lists of these papers, while forward snowballing involved identifying studies that cited these 62 primary studies. We then applied our inclusion criteria to the studies that we discovered [38]. This snowballing process is illustrated in Step 9 of Figure 1. By the end of this process, we identified an additional 40 primary studies, bringing the total to 102.

3.2. Inclusion and Exclusion Criteria

The academic body of knowledge in software testing is quite large. There are feasibility issues with inspecting all software testing studies to see which qualitative aspects they include. A naive query on Scopus searching within titles, abstracts, and keywords of articles for "software testing" returns more than $85,000$ papers. Therefore, we outline a set of inclusion criteria in which we aim to ensure that included studies are the most relevant literature.

Inclusion Criteria:

- Qualitative Research Focus: Include studies that primarily utilize qualitative research methods such as interviews, focus groups, ethnographic studies, case studies, content analysis, or participant observation to explore software testing.
- Focus on Software Testing: Studies should explicitly focus on software testing, encompassing a broad range of subjects within this area, including but not limited to manual testing, automated testing, design of test cases, evaluation of test effectiveness, and management of testing processes.

- Academic Literature: Include academic sources such as journal articles, conference papers, and theses that contribute to understanding the qualitative aspects of software testing.
- Published in English: Studies must be available in English to ensure accessibility.
- No Time Restrictions: Include studies from the inception of software testing as a field to capture the full spectrum of qualitative-focused research conducted over time.

Exclusion Criteria:

- Gray Literature: Exclude gray literature and unpublished studies to maintain academic rigor.
- Multiple Versions: Should a study exist in multiple versions (e.g., an initial conference paper followed by an expanded journal article), exclude earlier versions in favor of the most comprehensive one.
- Indirect Research Studies: Exclude secondary or tertiary studies and research compilations that synthesize findings from other studies as their main contribution. This ensures focus remains on studies that directly engage with qualitative data and firsthand observations in software testing.

3.3. Data Analysis

This subsection outlines the processes we followed to analyze the primary studies we identified to address our research questions.

*3.3.1. **RQ 1.1** What are the main qualitative methods used in software testing studies?*

We investigated the qualitative methods of our primary studies in three distinct segments. These segments are "research strategy", "data collection", and "data analysis". Research strategies, in this context, help identify clusters of studies that share a common high-level goal and approach, grouping similarly minded papers. According to Stol and Fitzgerald [39], a research strategy is a broad term that involves the use of one or more data collection methods, while also being concerned with the setting used to conduct the study. Storey et al. [40] applied a framework from behavioral research strategies [41] to software engineering; we use a similar approach that separates strategy and data collection but with a specific focus on qualitative methods. To guide our work, we used the research strategies from "Qualitative inquiry and research design" by Creswell and Poth as a starting point [42].

- **Strategy** conveys how the study is designed, set up and conducted [40]. All studies have a strategy, even if the authors do not explicitly mention it. Examples include ethnographic studies, grounded theory studies, case studies, etc.
- **Data Collection** is how the authors have obtained their subject material. For example, these could be anything from interviews and surveys to workshops.
- **Data Analysis** is how data collected above is prepared for interpretation. In this context, techniques such as different coding strategies represent methods of data analysis.

This categorization is designed to convey information about the method and execution of a study. However, not all aspects of the categorization are explicitly detailed in every study. To address this, we document whether a categorization aspect was explicitly stated in the paper or interpreted by us. Additionally, interpretation was applied in the categorization when the primary study's authors provided details about the strategy, data collection, or data analysis processes but did not specify a particular method.

To develop a list of categories for each segment, we employed an iterative process. This process began with initial lists that included categories outlined by Creswell and Poth [42] and those we identified through our own experiences as researchers. We expanded these lists when something from a primary study did not align with the existing categories. At the end of categorization, we have identified eight research strategies, ten data collection methods, and fifteen data analysis methods. For instance, our data collection methods included semi-structured interviews, surveys, focus groups, and observations in the initial list, but did not initially include workshops or the think-aloud method which were discovered when we went through the primary studies.

We reviewed the entire study to ensure accurate categorization. Once an initial categorization of all primary studies was completed by one of the authors (first rater), we conducted an interrater reliability process to verify our results.

While categorizing multiple aspects with multiple raters (in our case, the authors), interrater reliability played a crucial role in ensuring consistency and objectivity among the assessments made by different raters. In our study, this process begun with an individual assessment of primary studies where the first rater went over the entire set to refine the core lists and create the initial version of the categorization. After this step, three other raters randomly evaluated 10% of the studies from the set. We calculated three Cohen's Kappa scores [43] $(0.508, 0.605, 0.512)$ to measure the agreement levels between each of these raters and the primary rater, specifically focusing on the research strategy aspect. These scores pointed to moderate agreement initially. The disparities identified during this preliminary phase were then methodically discussed and reconciled through a detailed review of the contested categorizations. This iterative process of evaluation, discussion, and adjustment continued until full agreement was reached between all raters to ensure reliability. After these discussions, the first rater re-categorized all primary studies to reflect the agreed-upon understanding. The data collection and analysis aspects required less interpretation; therefore, we did not repeat these steps for these aspects; instead, they were directly integrated during the raters' consensus-building discussions on research strategy evaluations.

3.3.2. ***RQ 1.2*** *What are the purposes of qualitative studies in software testing?*

We investigated this by looking at what the authors reported to be their purposes for their studies in the papers. The reason why they choose to do a qualitative study is usually not explicitly stated. Therefore, we considered the parts of the articles that were concerned with problem formulation, motivation, and research design. We went through every study, marking the passages that were related to the purpose behind method selection. We used the passages we extracted to purposefully code the authors' indicated purposes. We then performed axial coding that ended up in groups of purposes.

This process for eliciting the purposes of the studies was developed by the first and second author collaboratively based on a random sample of a few primary studies. Then, the first author conducted the full passage extraction, open and axial coding process. Both authors discussed the resulting open and axial codes until they reached a negotiated agreement, referring back to the original papers when disagreements arose.

The first author conducted the passage extraction and open coding process. The first and second author then collaboratively reviewed the extracted passages, validated the assigned codes, and jointly carried out the axial coding process.

3.3.3. ***RQ 2*** *Are there any common resources used by the qualitative studies of software testing?*

To address this research question, we looked at the sources each primary study cited and calculated which resources were cited most frequently. This was done by getting a bibliography list for each primary study from SCOPUS and Google Scholar. We then extracted resource titles from bibliographic information and applied prepossessing measures for the cases where a resource would be represented in multiple titles (e.g., one that includes the subtitle for a resource and one that does not). We then looked at the most common resources overall for a given research strategy. The latter was done to discover resources that were influential within a research strategy, for example, a guide on how to conduct case studies.

3.3.4. ***RQ 3.1 – 3.2*** *Which areas of software testing use qualitative methods the most (RQ 3.1) and the least (RQ 3.2)?*

To investigate these research questions, we first needed an established list of research areas in software testing. We utilized a taxonomy of software testing research areas identified in a mapping study by Salahirad et al. [29]. In their study, the subtopics of research within the field of software testing are systematically represented through clusters derived from author-assigned keywords spanning several decades of literature. Clusters were identified by examining the interconnectedness of keywords in published works, thus organizing the field into focused areas of study. Each cluster represents a high-level research topic. By identifying and categorizing these clusters, the authors provide a structured overview of the main themes and directions of software testing research.

During the categorization step for RQ1.1, conducted by the first four authors, we also identified the subject areas of our primary studies related to software testing as we reviewed each study. This information was incorporated into the interrater reliability process described for RQ1.1.

Next, we mapped these identified subject areas to the taxonomy proposed by Salahirad et al. [29] by assigning each subject area–primary study pair to a corresponding subject within the taxonomy. This approach allows us to analyze the distribution of qualitatively focused studies across different areas of software testing. This mapping was done iteratively by the first author with regular discussions with the second author in between iterations. With this method, we identify areas where qualitative methods are prevalent or lacking compared to the overall body of work in software testing.

4. Results

This section presents the results of our systematic mapping study of the studies using qualitative methods in software testing. The data for this study is available in the replication package [44].

4.1. General Information

Following our selection process, we identified 102 primary studies, spanning from 2003 to 2023, which cover the past two decades of research in software testing. It is worth noting that our search, along with subsequent snowballing, did not identify any studies prior to 2003 that met the inclusion criteria specified in our methodology. The number of studies published as primary studies for this mapping has been increasing over the years as demonstrated in Figure 2. This indicates an increase in the popularity of qualitative methods in software testing research. We also examined the publication outlets for our primary studies and found the majority (70) were presented at conferences. Figure 3 illustrates the venues that appeared more than once. "ESEM[1]", "ICSE[2]", and "ICST[3]", while the most common journals are "IST[4]", "SQJ[5]", and "JSS[6]". The full names of the venues in Figure 3 can be found in the Appendix Table A .11.

*4.2. **RQ1.1:** Main Qualitative Methods*

4.2.1. Methodological Definitions

At the end of our 3 segmented categorization of our primary studies, we identified eight research strategies, ten data collection methods, and fifteen data analysis methods that were used within the primary studies. However, the definition of a qualitative method could potentially change between individuals. We share the definitions we have to make our results and findings more transparent and generalizable. The overview of popular options for these 3 segments is demonstrated in Figure 4.

Research Strategy Definitions.

- **Case Study:** This method shares similarities with ethnography but focuses on providing an in-depth exploration of a bounded system (such as an activity, event, process, or individual) through extensive data collection. The goal is to report detailed case descriptions and identify case-specific themes. The primary distinction from ethnography is that case studies aim to develop a deep understanding of a single case or to explore an issue or problem using the case as an example, while ethnography seeks to understand how a culture functions as a whole [42].
- **Ethnography:** Ethnography is a method in which, "the researcher describes and interprets the shared and learned patterns of values, behaviors, beliefs, and language of a culture-sharing group" [45]. In ethnography, the researcher looks for patterns of the group's mental activities, such as their ideas and beliefs or material activities (e.g., their behaviour expressed by their actions which are observed by the researcher) [46].

[1] International Symposium on Empirical Software Engineering and Measurement
[2] International Conference on Software Engineering
[3] International Conference on Software Testing, Verification and Validation
[4] Information and Software Technology
[5] Software Quality Journal
[6] Journal of Systems and Software

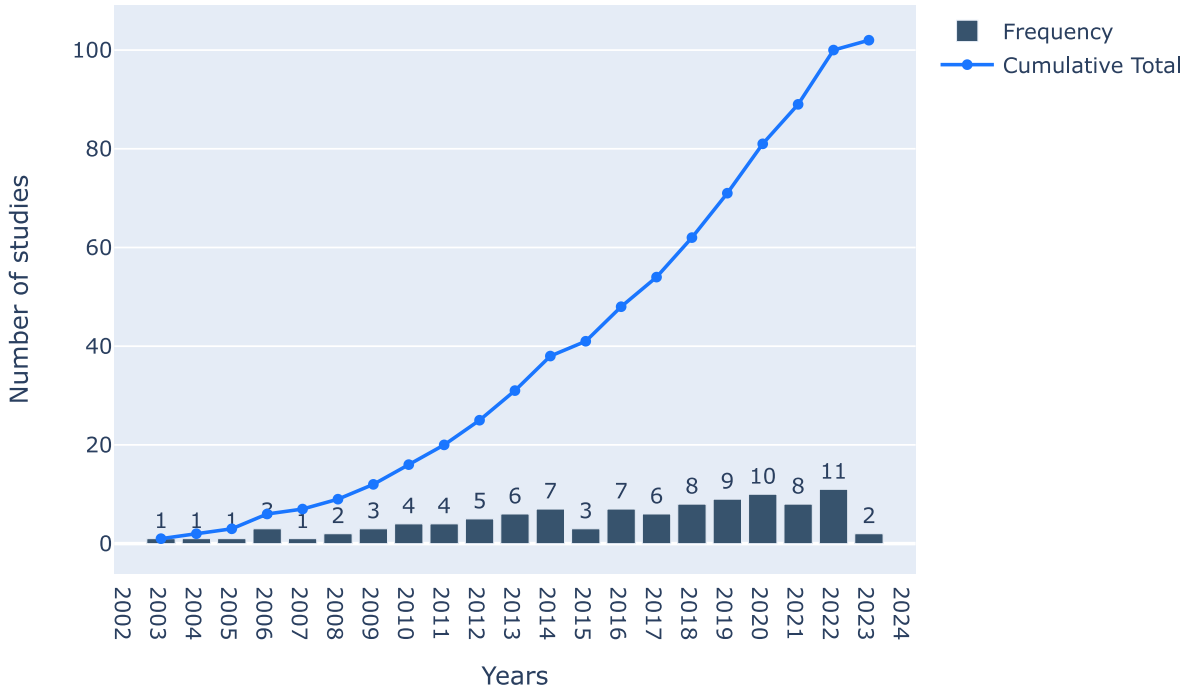

Figure 2. Number of primary studies over years

- **Grounded Theory:** According to Creswell [47] grounded theory is "a systematic, qualitative procedure used to generate a theory that explains, at a broad conceptual level, a process, an action, or an interaction about a substantive topic". The intent of a grounded theory is to move beyond description and to generate or discover a theory, a "unified theoretical explanation" [48] for a process or an action. All participants in the study would have experienced the process in question, and the development of the theory could help explain practice or provide a framework for future research [42]. There are variants of grounded theory such as classical [49], Straussian [50], or constructivist [51] grounded theory.

- **Action Research:** A participatory approach that combines research and action, aimed at solving a problem or improving a situation. The researcher is in direct collaboration with the subject during the research process which means that the researcher is attempting to have an impact while the research is still being conducted. This involves collaboration with stakeholders and typically follows a cyclical process of planning, acting, observing, and reflecting [52].

- **Engineering Research:** Systematic inquiry and experimentation in the field of engineering, aimed at developing new technologies, materials, or processes, or enhancing existing ones, often with a focus on practical applications and solutions. In contrast to action research, engineering research does not actively involve participants during the research process and aims to seek more generalizable solutions to problems. This term is also known as design science in the information science community. However, this is not the case in general. The SIGSOFT empirical standards on software engineering recommend using the term "engineering research"[53] instead.

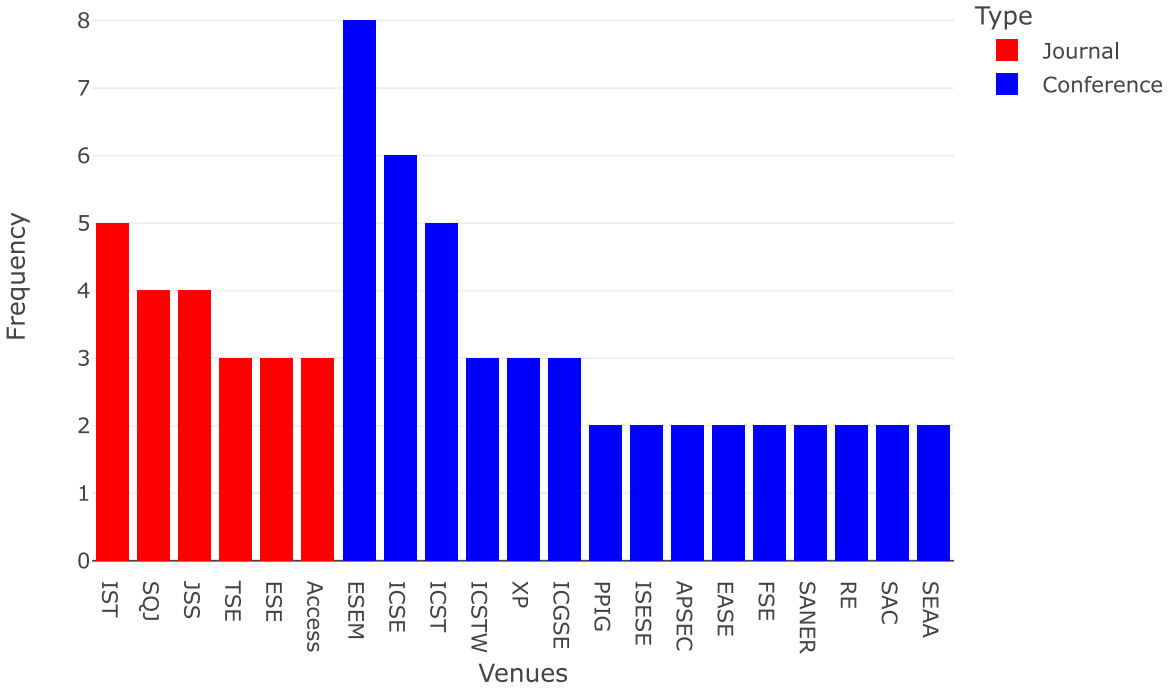

Figure 3. Publication venues of primary studies

- **Knowledge Gathering:** We define this term as referring to studies that do not align with other research strategies and primarily aim to explore and understand specific situations. These studies focus on understanding or describing a particular context, rather than developing or testing theories. Often, their objective is to gather insights, perceptions, or detailed descriptions to build a deeper understanding of the subject. However, we intentionally avoid labeling them as "exploratory" to prevent any negative connotations associated with the term[7], as it is sometimes used to imply lower-effort or less rigorous studies. Instead, we emphasize their value in generating knowledge and providing rich, contextual understanding that complements conventional research frameworks.

- **Correlational Designs:** Measuring variables and determining the degree of relationship that exists between them [54] instead of manipulating an independent variable, as in an experiment. Although it is not as rigorous as an experiment, correlational designs can be used to relate variables or predict outcomes.

- **Narrative Research:** Narrative research collects stories from individuals (can be accompanied by auxiliary sources like documents) about individuals' lived and told experiences. The idea behind creating the collection can be to analyze what was said (thematic), the structure of the story, or who the story is directed towards [55]. Narrative stories often contain specific tensions or interruptions that researchers highlight in conveying these stories [56].

[7]We cannot determine the exploratory nature of each study with absolute certainty. Additionally, papers that fall short of their intended methodology are also included here.

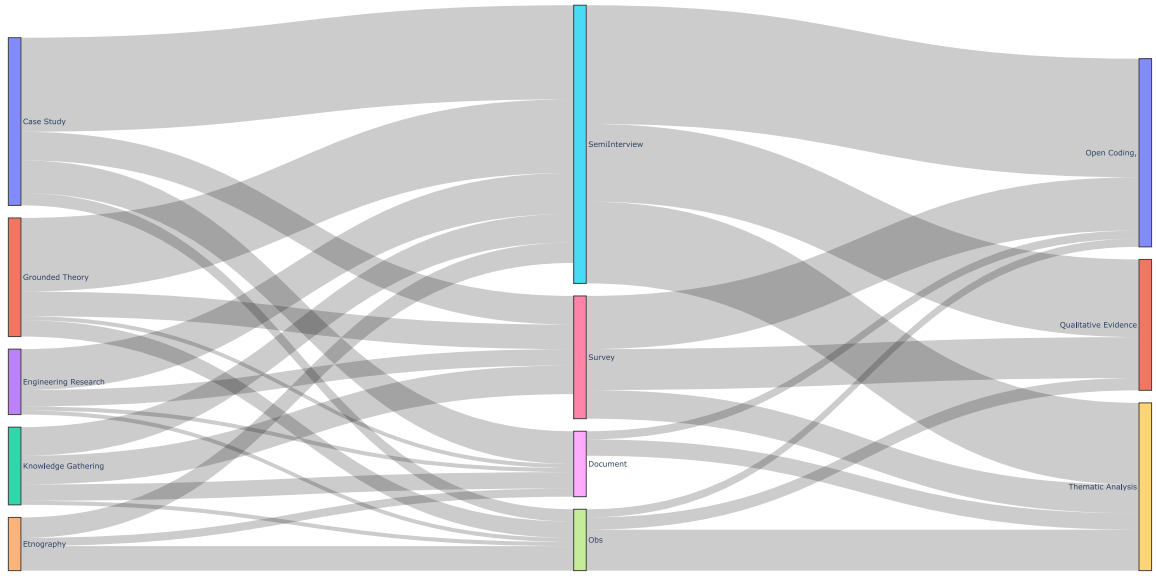

Figure 4. Breakdown of qualitative methods of primary studies using the 3 segmented categorization. From left to right, columns represent the segments: research strategy, data collection, and data analysis. The flows between columns highlight the connections and relationships between segments.

Data Collection Method Definitions. In most cases, the data collection aspects are explicitly stated by the primary studies. We identified the data collection methods listed below, while an overview of the methods and their prevalence is demonstrated in Figure 6.

- **Survey** is a data collection instrument directed at a predefined group of respondents. Surveys can include open-ended questions and they can be conducted via phone, online or in person [57].
- **Observation** This term includes the activity and anything that pertains to observation including notes and recordings of observation, etc [57].
- **Document study** is a broad term we use to refer to the study of data in the form of documents or similar artifacts. These could be anything from tool documentation to code review artifacts to online forum posts.
- **Focus group** is a method that emerged from the extension of open-ended interviews to group discussions. They are designed to obtain perceptions from group members about the predetermined topic of discussion [58].
- **Experiment** is "an empirical inquiry that manipulates one factor or variable of the studied setting [59]". Experiments can be oriented towards either human or technology involvement. When humans participate, there may be a decrease in control due to the impact of human biases [59].
- **Workshop** is a data collection method in which participants collaborate on a specific topic. It is interactive and encourages group discussions, problem-solving, and the generation of new ideas [60].
- **Think aloud** technique involves collecting verbal data from participants as they articulate their thoughts while performing a task assigned by the researcher [61].
- **Interview** is a data collection method where the researcher asks participants questions to gain insights on specific topics. Typically, the researcher uses an interview guide to maintain consistency in the data collection

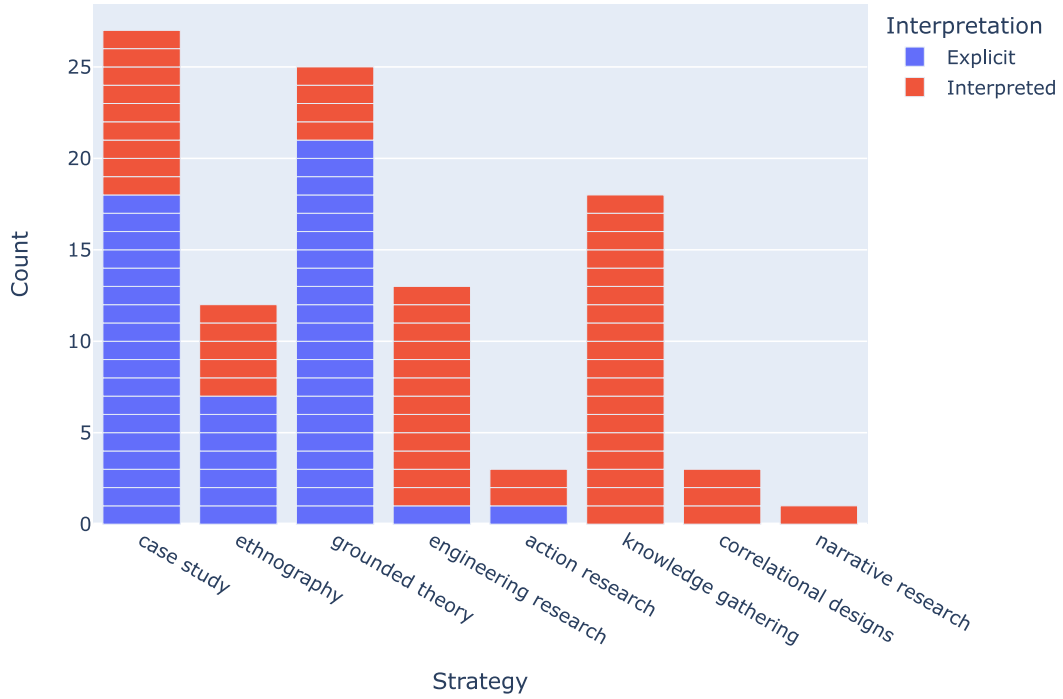

Figure 5. Overview of research strategies

process. The degree to which the researcher follows the guide, and the level of structure or formality involved, determines the type of interview being conducted [62].

- **Semi-structured interviews** is the interview type in which the researcher adheres to an interview guide of planned questions or topics however, depending on the participant's answers questions can change or skipped or new questions, topics can be added [62].
- **Unstructured interviews** are usually conducted informally as a controlled conversation about researchers interests [62].

Data Analysis Method Definitions. Data analysis methods are typically explicitly stated. In cases where the process was described in sufficient detail but no specific method was named, we inferred the method. The overview of the data analysis method we detected can be seen in Figure 7.

- **Thematic Analysis** involves identifying broader patterns or themes within the data and can be applied flexibly across various theoretical frameworks. The process of thematic analysis begins with familiarizing oneself with the data, followed by coding the entire dataset. Next, initial themes are generated and reviewed as potential themes. Finally, the themes are defined and named [63].
- **Open-Axial Coding** is a common method for qualitative data analysis. Open coding is the first step in identifying distinct concepts and themes within the data, aiming to represent the data and phenomena through general concepts. Axial coding is the second stage that focuses on refining, organizing, and categorizing these themes more precisely by using codes from the first step [64].
- **Open-Selective-Theoretical Coding** is a grounded theory specific variant of open-axial coding scheme that has additional selective and theoretical coding steps to support theory building [64, 48].

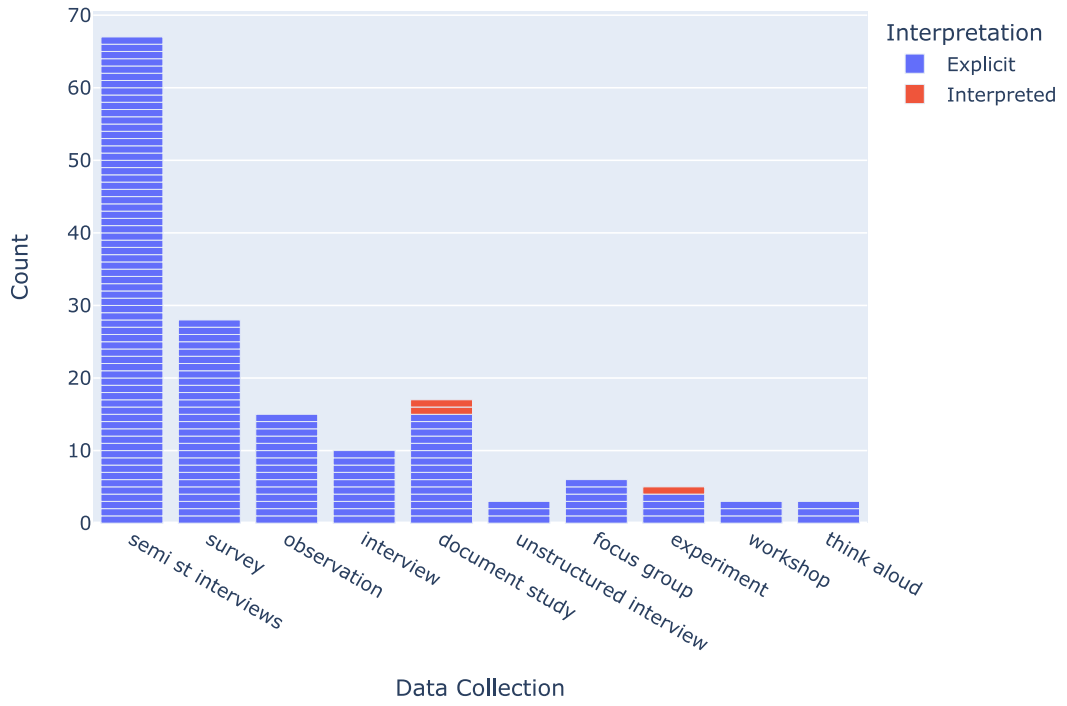

Figure 6. Overview of data collection methods

- **Statistical Analysis** involves the collection and examination of numerical data to uncover trends or patterns. In some primary studies [65, 66, 19, 67], quantitative data is collected as a supplementary measure and is analyzed and validated using descriptive statistics (such as mode, median, and standard deviation) and tests for statistical significance.
- **Qualitative Evidence** is about the instances where studies utilize a segment of qualitative data (e.g., participant quotes) *as is* to substantiate their arguments, provided that this data forms a substantial element of the paper's discourse. Although this is not a data analysis method, presenting data in this manner to carry out a study's discourse serves a similar purpose to data analysis.
- **Qualitative Content Analysis** is a rule based data analysis method that starts with a research question and caries over with inductive and deductive steps which is developed by Mayring and Fenzl [68].
- **Actor-Network Theory** is a framework that focuses on exploring the relationships and interactions between both human and non-human actors in a network based on common interests and goals. Data analysis is built upon actors, actor-network relations, and moments of translation [69, 70].
- **Semantic Equivalence Analysis** is a process of analyzing data by summarizing data points, synthesizing concepts from these summaries, and evaluating which data points support these concepts. The degree of support found in the data for each concept is then used to draw conclusions [71].
- **Notice Collect Think** is a process that simplifies the basics of qualitative data analysis in order to provide a foundation for more complex methods. It consists of three steps. The first step is about recording things that are noticed in data (coding) followed by sorting and categorizing what is noticed. Finally, conclusions are drawn by looking for patterns or relationships within or between collections [72].

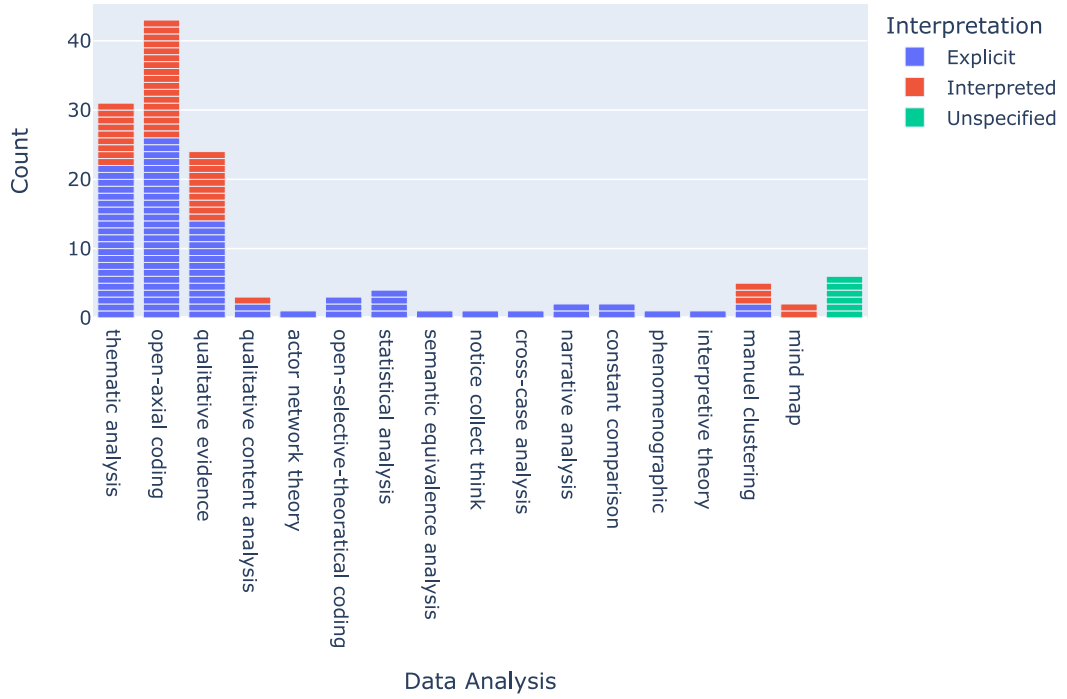

Figure 7. Overview of data analysis methods

- **Cross-Case Analysis** is a qualitative analysis method to compare and contrast data across different cases or settings in order to identify patterns, similarities, and differences. It is particularly useful when the data can be naturally divided into cases, which are then treated as individual units of analysis [7].
- **Narrative Analysis** is a method that interprets stories or experiences shared by individuals. Researchers begin by closely reading the text and taking margin notes. The narrative is then organized into a chronological sequence to understand the flow of events or experiences. Lastly, data is further classified to identify stories, special events and contextual materials [42].
- **Constant Comparison** is a core technique mainly used in Grounded Theory. Data is continuously compared against other data points or concepts to develop and refine emerging categories and theories. As new data is collected, it is systematically compared with previous incidents and evolving concepts to sharpen understanding and verify categories [73].
- **Phenomenography.** Phenomenographic analysis focuses on understanding how a group of people collectively experience or comprehend a phenomenon, emphasizing the second-order perspective. The result is a set of categories that describe qualitatively different ways of understanding or experiencing the phenomenon, organized into outcome spaces that often reflect a hierarchy of complexity [74].
- **Interpretive Theory** is one of the approaches to theory building in Grounded theory. It combines abstract conceptualization, rich theoretical claims, subjective reflection, and creative interpretation to offer nuanced insights into complex social phenomena [51].
- **Manual Clustering.** Grouping data manually into clusters or themes based on researcher judgment. This approach may be considered an ad hoc alternative to thematic analysis, as it lacks a standardized procedure for

implementation.

- **Mind Map.** A diagram used for visually organizing information, often used in qualitative research for brainstorming and idea generation. They can be used by researchers to identify themes and ultimately interpret meaning [75].

4.2.2. Results of categorization

In our population of primary studies, we can observe that the most prevalent research strategies are "case study", "knowledge gathering", and "grounded theory" as demonstrated in Figure 5. The breakdown of primary studies according to their respective research strategy can be found in Table 2. Moreover, Table 3 demonstrates the distribution of data collection and data analysis methods for each research strategy.

Strategy	**Primary Studies**
Action Research	[76, 77, 78]
Case Study	[79, 80, 81, 69, 82, 83, 84, 37, 85, 86, 87, 88, 89, 74, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102]
Correlational Designs	[103, 104, 67]
Engineering Research	[105, 65, 35, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115]
Ethnography	[116, 36, 117, 118, 66, 119, 120, 121, 122, 123, 124, 61]
Grounded Theory	[125, 126, 127, 128, 129, 34, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 4, 140, 141, 142, 143, 144, 145, 146, 147]
Knowledge Gathering	[148, 149, 71, 150, 151, 152, 153, 19, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163]
Narrative Research	[164]

Table 2: Primary Studies by Research Strategy

Table 3. Research Strategies Grouped by Data Collection and Analysis Methods

Strategy	**Case Study (27)**	**Grounded Theory (25)**	**Knowledge Gathering (18)**	**Engineering Research (13)**	**Ethnography (12)**	**Correlational Designs (3)**	**Action Research (3)**	**Narrative Research (1)**
Data Collection Methods								
Semi structured interview	23	18	7	10	5	2	1	1
Survey	7	6	7	4	0	2	1	1
Observation	3	4	1	1	6	0	0	0
Interview	0	3	3	1	3	0	0	0
Document	8	1	4	1	2	0	1	0
Unstructured interview	0	0	0	0	3	0	0	0
Focus group	2	1	1	1	0	0	1	0
Experiment	1	0	0	1	1	2	0	0
Workshop	0	1	0	1	0	0	0	1
Think aloud	0	1	0	1	1	0	0	0
Data Analysis Methods								
Thematic analysis	9	5	5	3	7	0	1	1
Open axial coding	9	16	7	6	3	1	1	0
Qualitative evidence	1	12	4	1	2	1	2	1
Qualitative content analysis	0	0	1	2	0	0	0	0
Actor network theory	1	0	0	0	0	0	0	0
Open selective theoretical coding	0	3	0	0	0	0	0	0
Statistical analysis	0	0	1	1	1	1	0	0
Semantic equivalence analysis	0	0	1	0	0	0	0	0
Notice collect think	1	0	0	0	0	0	0	0
Cross-case analysis	0	0	1	0	0	0	0	0
Narrative analysis	2	0	0	0	0	0	0	0
Constant comparison	0	1	0	1	0	0	0	0
Phenomenographic	1	0	0	0	0	0	0	0
Interpretive theory	0	1	0	0	0	0	0	0
Manual clustering	4	0	0	0	1	0	0	0
Mind map	1	0	0	0	1	0	0	0
Not available	2	0	2	1	0	1	0	0

Case Study. Case study is the most prevalent strategy in our primary studies with 27 instances; 18 of these are explicitly stated to be case studies and 9 are interpreted by us. Among these studies, the most common data collection method is semi-structured interviews (23), followed by documents (8) and surveys (7). For data analysis,

coding-based methods are predominant, with thematic analysis and open-axial coding each being used in 9 studies. Additionally, some studies employ different techniques, including manual clustering (4) and narrative analysis (2).

Grounded Theory. Grounded Theory is the second most common research strategy with 25 instances. Most of these studies explicitly state that they use grounded theory (21). The most frequent data collection methods are semi-structured interviews (18), surveys (6), and observation (4). The most popular data analysis method is open-axial coding (19) followed by thematic analysis (5).

Knowledge Gathering. Our initial list of research strategies did not include the category "knowledge gathering". After reviewing the studies, we refined our definition of ethnography to place greater emphasis on immersion. The studies that were not included in the new version were labeled as knowledge-gathering studies. After readjustments, we made another pass throughout the whole list of primary studies and found 18 studies. The purpose behind this adjustment was to create a category for software engineering studies that utilize qualitative methods that only focus on understanding or describing a situation. Since this is not a previously established strategy, all instances of it are interpreted. Common data collection methods are semi-structured interviews (7), surveys (7), and documents (4). Data analysis is conducted with open-axial coding (7) and thematic analysis (5), however, two studies do not report on their data analysis methods.

Engineering Research. Engineering research represents another research approach that we have introduced, therefore it is mostly interpreted. Within the 13 occurrences of this strategy, semi structured interviews are the most frequently used method for data collection, appearing in 10 instances, followed by surveys, which are used 4 times. Additionally, more interactive data collection techniques, such as think-aloud sessions, focus groups, workshops, and experiments, have each been employed once. For data analysis, the most common methods are open axial coding (6), thematic analysis (3), and qualitative content analysis (3).

Ethnography. In our collection of primary studies, we identified 7 studies that explicitly mention conducting ethnography, and we interpreted an additional 5 studies as employing ethnography as their research approach, bringing the total to 12 studies. Observation (6) emerges as the most frequently utilized method for data collection in these studies, followed by different types of interviews, including semi-structured (5) and unstructured (3). For the analysis of collected data in ethnographic studies, thematic analysis (7) and open-axial coding (3) are the common methodologies employed.

Correlational Designs. We interpreted 3 studies as correlational designs. The data collection methods for these studies were semi-structured interviews (2), surveys (2), and experiments (2). For analyzing the data, open axial coding and statistical analysis methods were used once, and one of these studies did not report on how they conducted their analysis.

Action Research. We classified 3 studies under the category of action research, where two were interpreted and one was explicit. Data collection in these studies was conducted using: document analysis, focus groups, surveys, and semi-structured interviews, each method being employed once. Data analysis was done with open axial coding and thematic analysis, and one study did not report on their data analysis.

Narrative Research. We interpreted one study to be narrative research. This study employed semi-structured interviews, a workshop, and a survey for data collection, with thematic analysis being used to analyze the data.

RQ1.1: What are the main qualitative methods used in software testing studies?

The most commonly used research strategies in qualitative software testing studies are case studies, grounded theory, and knowledge-gathering approaches. Semi-structured interviews were the most frequent data collection method, followed by surveys and document analysis. Open coding and thematic analysis were the primary methods for data analysis.

*4.3. **RQ1.2:**Purposes of the qualitative studies in software testing*

To investigate why studies utilize qualitative methods, we went through each primary study in our set and collected relevant passages. We detected two types of passages that are typically relevant to this part of the analysis.

- passages that present a research gap followed by the aim of the research.
- passages that describe a method and why that method fits the research problem at hand.

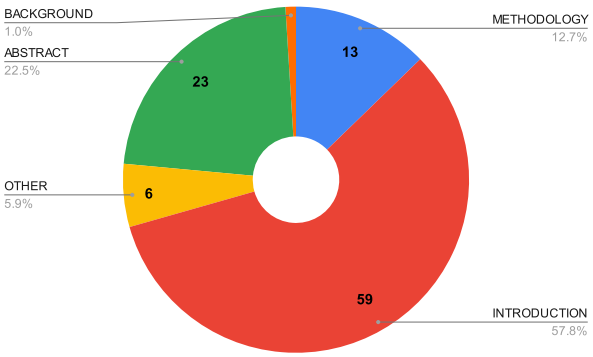

Figure 8. Sections of paragraphs that state purpose

Most of the time, we found the first type of passage in the introduction sections and the second type in the methodology sections.

Figure 8 presents a breakdown of where these paragraphs are located within their respective studies. The *introduction* sections provided the most information, followed by the *abstracts* and *methodology* sections. In addition, six other studies included relevant information in sections such as the research process or study details.

The second type of passage is more useful to see how studies choose their methods, however having only the first type of passage was the most common way of reporting we came across. After collecting the relevant passages, we performed open and axial coding on these passages until we reached larger code groups to get an overlook of the purposes as they are reported in our primary studies. These groups of purposes are demonstrated in Table 4.

The coding process resulted in five key themes. Better Understanding, the most frequent group (82), focuses on deepening subject knowledge, particularly how practitioners work and their typical measures, alongside topics like industry-academia alignment. Perspective (59) explores various viewpoints, emphasizing practitioners' perspectives and industry insights. Human Aspects (20) delve into interpersonal and cultural dimensions, such as communication, teamwork, and practitioner motivations. Education and Training (14) centers on skill development, learning processes, and enhancing professional competencies. Lastly, Tools & Methods (11) evaluation of tools, methods, and best practices, with an emphasis on their effectiveness and utility.

RQ1.2: What are the purposes of qualitative studies in software testing?

Our analysis indicates that the main motivations for conducting qualitative studies in software testing are to gain a deeper understanding of practitioner activities, explore industry practices from practitioners' perspectives, and enhance researchers' comprehension of the research subject.

4.4. ***RQ2:*** *Common Resources*

We are motivated to find guidelines that are popular in the area of qualitative research in software testing. Although common resources do not directly indicate guidelines we expect them to show up among the common resources. Identifying these guidelines would enable researchers who are new to qualitative research in software testing to become familiar with the state-of-the-art and thus decrease the barrier of entry for contributing to this area. In contrast, the lack of guidelines (or the existing guidelines not being used) could indicate a lack of support for applying qualitative methods in the area; it can also indicate the existence of ad hoc approaches to qualitative research.

4.4.1. *Resources for All Primary Studies*

First, we look at the overall picture by considering all primary studies. Then we divide primary studies by research strategy and inspect these separately.

We begin by presenting the common resources used across the broader set of primary studies. While the five most frequently used research strategies account for 97% of all studies, categories such as "Action Research", "Correlational

Table 4. Summary of Open Coding Results on Purposes of Primary Studies

Groups	Count	Description/Subcodes
Better Understanding	82	Deepening understanding of subjects.
	47	*Understand how practitioners work.*
	34	*Better understand subject.*
	14	*Identify typical measures by practitioners.*
	13	*Industry-academia alignment*
	11	*Identify improvement needs*
Perspective	59	Exploring viewpoints in professional/academic settings.
	42	*Practitioners' perspective*
	15	*Industry's view*
	5	*Investigate students' perceptions*
	2	*Practitioners' perspective on academic work*
Human Aspects	20	Interpersonal and cultural dimensions.
	6	*How practitioners communicate.*
	6	*Investigate interactions*
	6	*Practitioner motivations*
	3	*Effects of organizational activities*
	3	*Understand the role of experience*
	2	*How to improve teamwork*
	1	*Understand the role of culture*
Education and Training	14	Skill development and learning processes.
	6	*What skills are necessary for practitioners.*
	6	*How do practitioners learn*
	4	*Student learning ability and challenges*
	3	*How to better help professionals' skill acquisition*
Tools & Methods and Guidelines	11	Evaluation of tools, methods, and best practices.
	7	*Tool & Method Evaluation.*
	4	*Best practice guidelines*
	1	*Demonstrate improved effectiveness*

Designs", and "Narrative Research" are not examined on their own due to the limited number of primary studies. We determine the number of common resources to report for each research strategy by using an approach similar to the elbow method, where the number of resources reported varies according to the data. For example, in some research strategies, there are more than 10 distinct resources that appeared three times, but for brevity, we select a minimum number of occurrences that create a manageable table and remain representative.

Table 5: Common Resources in All Primary Studies

Author(s)	Resource Title	Count
Seaman, C.B.	Qualitative methods in empirical studies of software engineering [7]	21
Runeson, P. and Höst M.	Guidelines for conducting and reporting case study research in software engineering [2]	17
Glaser, B. and Strauss A.	Discovery of grounded theory: strategies for qualitative Research [49]	16
Bertolino, A.	Software testing research: achievements, challenges, dreams [165]	15
Rooksby et al.	Testing in the wild: The social and organizational dimensions of real world practice [1]	14
Ng et al.	A preliminary survey on software testing practices in Australia [166]	14
Eisenhardt, K.M.	Building theories from case study research [167]	13
Strauss et al.	Basics of qualitative research [50]	12
Beck, K.	Test-Driven Development: By Example [168]	11

Tassey, G.	The Economic impacts of inadequate infrastructure for software testing [169]	10

4.4.2. Ethnography Studies

The most cited resource among the 12 ethnography studies is "Testing in the wild" [1] (4). There are no commonly cited guides in these studies regarding conducting ethnography besides "Using Thematic Analysis in Psychology" [170], which can be useful for analyzing ethnographic data. There are software engineering specific studies regarding applying ethnography in the literature [171, 172], but our analysis indicates that these were not prevalent resources in our primary studies.

Table 6: Common Resources in Ethnography Studies

Author(s)	Resource Title	Count
Rooksby, J.	Testing in the wild: The social and organizational dimensions of real world practice [1]	4 (33%)
Shah et al.	Studying the influence of culture in global software engineering: thinking in terms of cultural models [173]	3 (25%)
Beck, K.	Test-Driven Development: by Example [168]	3 (25%)
Braun, V. and Clarke V.	Using thematic analysis in psychology [170]	3 (25%)
Runeson, P.	A survey of unit testing practices [174]	3 (25%)
Whittaker, J.A.	What is Software Testing? And why is it so hard? [175]	3 (25%)

4.4.3. Case Studies

Among the 27 case studies, the most popular subject category is "Human Aspects" with 5 studies, followed by "Creation Guidance", "Processes&Risk" and "Testing in Practice" with 4 studies each.

In Table 7 we list the most common citations in this group of studies. We observe that the most commonly used guideline is "Guidelines for conducting and reporting case study research in software engineering" [2].

Table 7: Common Resources in Case Studies

Author(s)	Resource Title	Count
Runeson, P. and Höst M.	Guidelines for conducting and reporting case study research in software engineering [2]	7 (26%)
Rooksby et al.	Testing in the wild: the social and organizational dimensions of real world practice [1]	4 (15%)
Merriam, S.B. and Tisdell E.	Qualitative research: a guide to design and implementation [9]	3 (11%)
Berner et al.	Observations and lessons learned from automated testing [176]	3 (11%)
Bertolino, A.	Software testing research: achievements, challenges, dreams [165]	3 (11%)
Braun, V. and Clarke V.	Using thematic analysis in psychology [170]	3 (11%)
Seaman, C.B.	Qualitative methods in empirical studies of software engineering [7]	3 (11%)

4.4.4. Grounded Theory Studies

There are 25 grounded theory studies. The most popular subject categories are "Testing in Practice" (5) and "Test Automation" (4) respectively. We observe that the most frequently referenced work is Seaman's "Qualitative methods in empirical studies of software engineering" [7]. This indicates a strong reliance on Seaman's framework for qualitative methods, which is foundational for understanding and applying empirical research techniques in the software engineering domain.

In terms of citations, Seaman's work is closed followed by the work of Corbin and Strauss "Basics of qualitative research: grounded theory procedures and techniques" [48]. Corbin and Strauss' work provides comprehensive procedures for grounded theory, guiding researchers through the data collection, coding, and theory development processes.

Table 8: Common Resources in Grounded Theory Studies

Author(s)	Resource Title	Count
Seaman, C.B.	Qualitative methods in empirical studies of software engineering [7]	9 (36%)
Corbin, J. and Strauss, A.	Basics of qualitative research: grounded theory procedures and techniques [48]	8 (32%)
Eisenhardt, K.M.	Building theories from case study research [167]	7 (28%)
Glaser, B. and Strauss, A.	The discovery of grounded theory: strategies for qualitative research [49]	7 (28%)

European Commission	SME definition (2003) [177]	6 (24%)
Klein, H.K. and Myers, M.D.	A set of principles for conducting and evaluating interpretive field studies in information systems [178]	6 (24%)
Paré, G. and Elam, J.J.	Using case study research to build theories of IT implementation [179]	5 (20%)
ISO/IEC	Information Technology - Process Assessment - Part 1: Concepts and Vocabulary [180]	5 (20%)

4.4.5. Engineering Research Studies

Among the 13 engineering research studies in our list, 5 are related to "Creation Guidance" and 4 are related to "Processes and Risk". The most cited study in this subset is about test patterns. However, we still see some guidelines in use with "Experimentation in software engineering" by Wohlin et al. [59], and "Guidelines for conducting and reporting case study research in software engineering" [2] by Runeson and Host occurring three times.

Table 9: Common Resources in Engineering Research Studies

Author(s)	Resource Title	Count
Meszaros, G.	xUnit test patterns: refactoring test code [181]	4 (31%)
Wohlin et al.	Experimentation in software engineering [59]	3 (23%)
Athanasiou et al.	Test code quality and its relation to issue handling performance [182]	3 (23%)
Runeson, P. and Höst M.	Guidelines for conducting and reporting case study research in software engineering [2]	3 (23%)
Jia, Y. and Harman M.	An analysis and survey of the development of mutation testing [183]	3 (23%)

4.4.6. Knowledge Gathering Studies

Among the 18 knowledge gathering studies, we identify 6 that focus on 'Testing Career and Education'. This combination of research strategy and subject area is the most prevalent, as illustrated in Figure 10. It represents the highest percentage, accounting for one-third of all knowledge gathering studies. There is a five-way tie for the most cited studies. We see "Testing in the wild" by Rooksby et al. [1] cited in 4 studies. Important to note is that this resource appears again here after being the second most common resource in case studies. Other common guidelines are the book "The Art of Software Testing" [184] by Myers and "Qualitative Methods in Empirical Studies of Software Engineering" [7] by Seaman.

Table 10: Common Resources in Knowledge Gathering Studies

Author(s)	Resource Title	Count
Myers et al.	The art of software testing [184]	4 (22%)
Rooksby, J.	Testing in the wild: the social and organizational dimensions of real world practice [1]	4 (22%)
Capretz and Ahmed	Making sense of software development and personality types [185]	4 (22%)
Weyuker et al.	Clearing a career path for software testers [186]	4 (22%)
Seaman, C.B.	Qualitative methods in empirical studies of software engineering [7]	4 (22%)

Identifying common resources in qualitative research on software testing provides insights into the foundational literature that guides current practices. These frequently cited works could identify key references for researchers, especially those new to the field. The presence of certain highly referenced guidelines, such as those by Seaman and Runeson, highlights their pivotal role in shaping research methodologies. However, the absence or limited use of guidelines may signal reliance on more ad hoc methods or the need for further development of standardized practices in qualitative research within this domain. These common resources reflect both the strengths and potential gaps in the literature offering future researchers valuable insights as they seek to contribute to this field.

RQ2: Are there any common resources used by the qualitative studies of software testing?

Across our set of primary studies, the most frequently cited resources offer foundational guidance on qualitative research. Highly referenced works include Seaman's "Qualitative methods in empirical studies of software engineering" [7] and Runeson and Höst's guidelines for case study research [2], and "Discovery of grounded theory" by Glaser and Strauss [49] and others. These resources seem to guide qualitative research efforts within software testing.

4.5. ***RQ3:*** *Subjects of the qualitative studies*

To identify the most and least popular subjects in our primary studies, we utilized the software testing subject taxonomy from Salahirad et al. [29]. By mapping the subjects of our primary studies onto this taxonomy, we determined which areas are the most populated and the least populated by our primary studies, revealing the topics that are prevalent or missing in qualitative software testing research.

4.5.1. *Clusters of research topics in software testing*

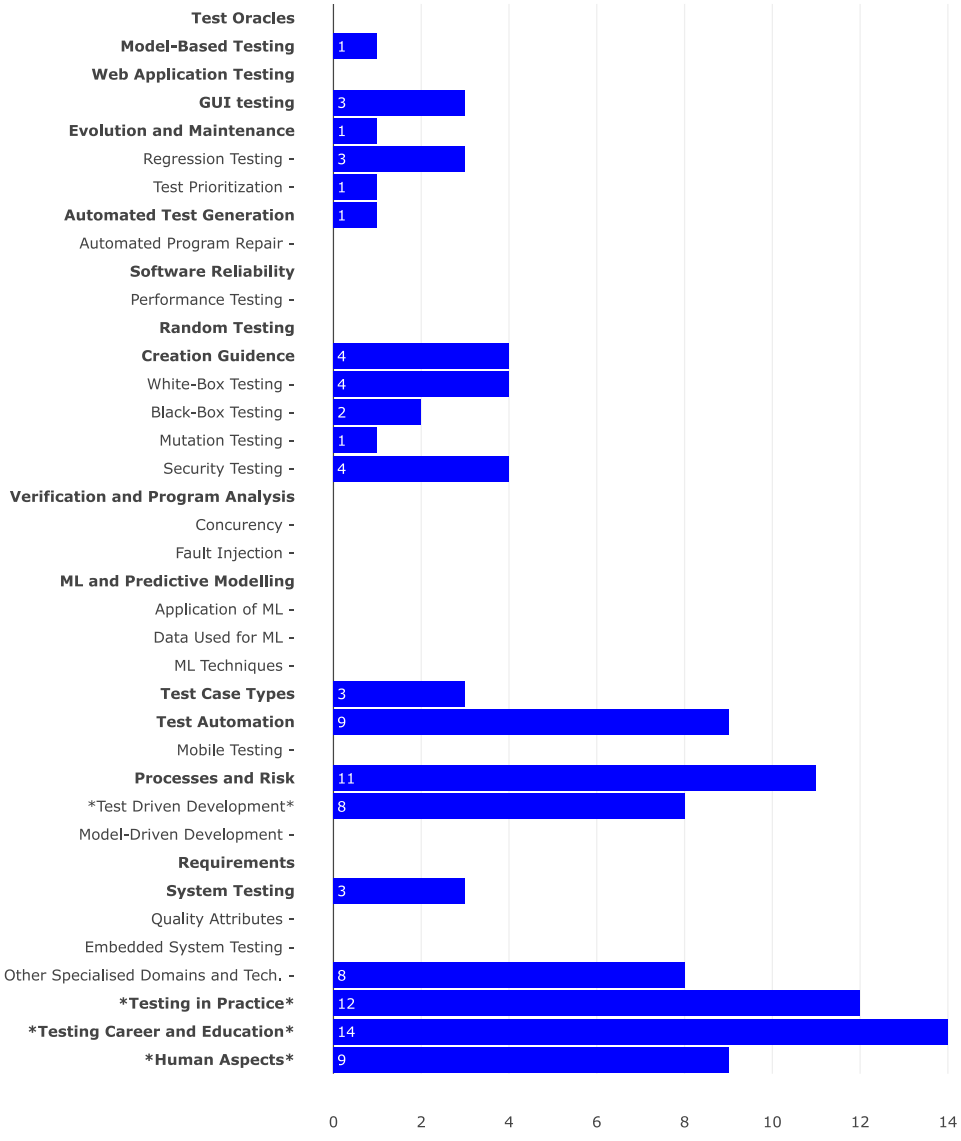

Figure 9. Subject areas of the qualitative studies. [Bolded categories represent broad topics, hyphenated categories indicate subcategories, and asterisks mark our additions to the taxonomy.]

Figure 9 shows the distribution of subjects by utilizing the software testing subjects taxonomy, where the bold cat-

egories indicate the broader categories and the hyphenated ones indicate their respective subcategories. For example, "Creation Guidance" is a broad category that has the subcategories "Security Testing","Mutation Testing","Black-Box Testing" and "White-Box Testing". These subcategories contain four, two, one, and four papers respectively. Additionally, there are four more papers classified under "Creation Guidance" that do not fall into any of the subcategories. Moreover, there were also cases in which our primary studies were left outside of this taxonomy due to their subjects. For example, a study about motivation to pursue a testing career [162] would not fit into this taxonomy.

For the primary studies that did not align with the existing taxonomy, we manually grouped them into new categories and integrated them into the established taxonomy. The newly added categories, which are "Test Driven Development", "Testing in Practice", "Testing Career and Education", and "Human Aspects" are demonstrated with asterisks in Figure 9. Important to note that we introduced the "Test-Driven Development" as a sub-category under the "Processes and Risk" broad category. We created this new sub-category to highlight this subject which has eight primary studies in our mapping study.

RQ3.1 Subjects with qualitatively focused studies

The software testing taxonomy consists of 16 broad categories of research subjects. After mapping our primary studies to these categories, we found that 9 of the categories were populated. Below is a list of these categories along with their representative keywords from Salahirad et al.[29].

- *Model Based Testing:* Model transformation, metamorphic relation
- *GUI Testing:* GUI, finite state machine
- *Evolution and Maintenance:* Program comprehension, change impact analysis
- *Automated Test Generation:* Genetic algorithms, branch coverage
- *Creation Guidance:* Certification, test adequacy
- *Test Case Types:* Unit testing, exploratory testing
- *Test Automation:* Test execution, testing tools
- *Processes and Risk:* Software quality, Test driven development
- *System Testing:* System testing, user interfaces

Among the studies that fit the existing taxonomy, the most prevalent subjects are **Processes and Risk** (11%), **Test Automation** (9%), Test Driven Development (8%, as a subcategory of Processes and Risk), and Specialized Domains (8% as a subcategory of System Testing). Another notable group of studies is in **Creation Guidance** (4%) with the subcategories White-Box (4%), Black-Box (2%), Mutation testing (1%) and Security testing (4%).

34% of our primary studies did not fit into the existing taxonomy. We clustered these studies into three new categories with a similar format of representative keywords:

- *Testing in Practice:* Organisational challenges, testing cultures
- *Testing Career and Education:* Testing career, motivation
- *Human Aspects:* Human aspects, non-technical factors

With the inclusion of these new categories, **Testing Career and Education** and **Testing in Practice** become the most populated. In our adaptation, these categories were introduced to better represent our primary studies within the existing taxonomy. Since the taxonomy we employed, developed by Salahirad et al. [29], was designed to represent the entire body of work in software testing, we initially anticipated mapping more of our primary studies without the need to create additional large categories. This could indicate a potential divide between the focus areas of qualitative software testing studies and the broader body of work in software testing. It indicates that qualitative studies tend to concentrate more on human-oriented aspects of software testing, while there may be valuable opportunities to explore technical aspects of software testing from a qualitative perspective.

Some broad categories in the taxonomy did not have any associated qualitative study in our sample. These categories are represented below with example keywords from Salahirad et al. [29]:

RQ3.2 Subjects absent of qualitatively focused studies

- *Test Oracles:* Test oracle, metamorphic relation

- *Web Application Testing:* Web applications, JavaScript
- *Software Reliability:* Reliability growth, quality control
- *Random Testing:* Adaptive random testing, statistical testing
- *ML and Predictive Modelling:* Defect prediction, neural networks
- *Requirements:* Requirements engineering, traceability

The lack of studies in these areas suggests that qualitative research has not extensively explored the technical or automated aspects of software testing, particularly in relation to machine learning, performance, or advanced techniques such as fault injection and concurrency. This may indicate a gap in the human and practical elements of these more technical topics, or it could suggest that these areas are primarily examined through quantitative methods since they appeared in the taxonomy but we were not able to populate them with our primary studies.

These gaps present potential opportunities for future research, particularly if there is a need to investigate the human, social, or contextual aspects of these areas in software testing. The application of qualitative approaches could offer valuable insight into how practitioners engage or perceive these technical practices.

RQ3: Which areas of software testing use qualitative methods the most?

We mapped qualitative software testing research topics onto an existing taxonomy to identify both frequently and rarely studied areas. Popular subject areas include “Processes and Risk”, “Test Automation”, and “Test Driven Development”, with an added focus on human-centric categories such as “Testing Career and Education* and “Testing in Practice” and “Human Aspects”.
Unexplored areas include technical topics like “Test Oracles”, “Web Application Testing”, “Machine Learning and Predictive Modeling”. The absence of qualitative work in these areas suggests they are primarily studied through quantitative methods.

4.6. Subjects and Strategies

We analyze the research strategies used in subjects to investigate the distribution of research strategies for each subject in the taxonomy with primary studies. This presents a cross-analysis of RQ1 and RQ3.1. Figure 10 shows the distribution of research strategies used across various topics in software testing. Brighter areas (from purple to yellow) highlight where certain research strategies are more popular than others, giving us insight into the research focuses within different areas of software testing. Understanding these patterns can help identify current trends and possibly encourage the use of underutilized research methods in future studies when they are applicable, ultimately enriching the field of software testing.

We made the following key observations based on our analysis:

- “Action Research,” (3) “Correlational Research,” (3) and “Narrative Research” (1) are scarcely used in the general set of primary studies.
- As shown in Figure 5, “Case Study” is the most frequently utilized research strategy overall. The heatmap reveals that this strategy was employed in all subject areas except for “Model Based Testing” and “Automated Test Generation”. The “Human Aspects” subject area most commonly used is “Case Study,” with 5 out of 9 studies employing this method.
- The “Ethnography” research strategy is used by only 5 subject areas out of 12. The most popular subject category is “Processes and Risk” (4 studies with 2 coming from “TDD”) followed by Creation Guidance (3 studies with 2 coming from “Security Testing”).
- Out of the 9 studies in the “Test Automation” subject area, 4 use “Grounded Theory.” Additionally, among the 12 studies on “Testing in Practice,” 5 employed “Grounded Theory”.
- There were 14 “Testing Career and Education” studies and 6 of them had “Knowledge Gathering” as their research strategy, which is the most heated point in the map.

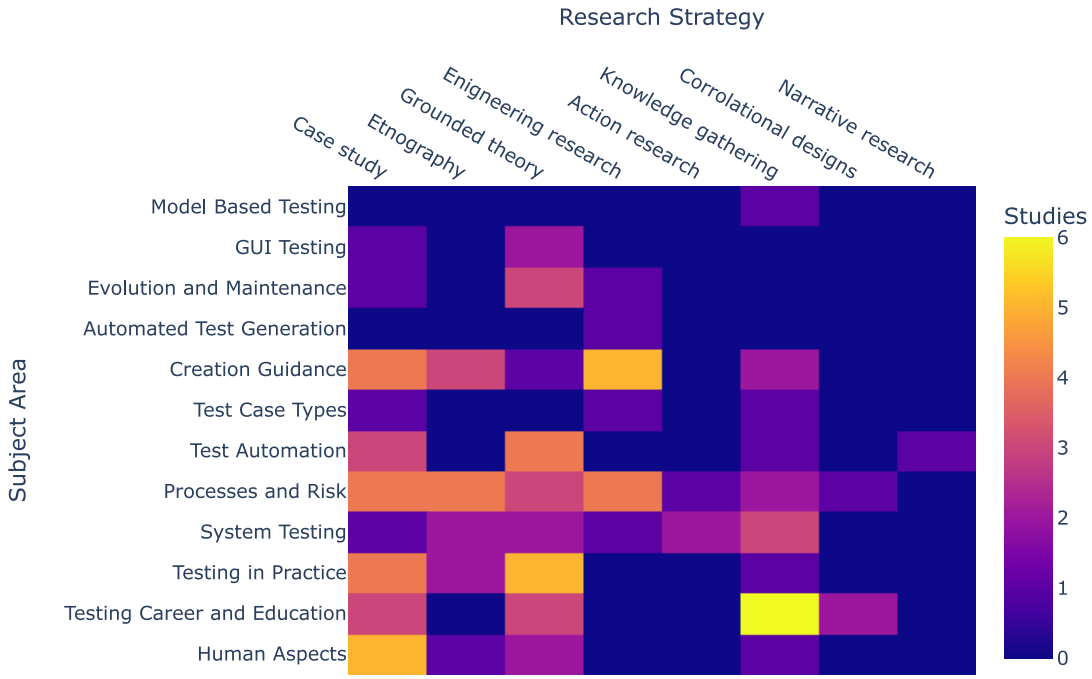

Figure 10. Heatmap of subject areas and research strategy

5. Discussion

In this section, we analyze and interpret the findings from our systematic mapping study. We begin by discussing the characteristics of primary studies that employ case studies, ethnography, grounded theory, or knowledge gathering. These are strategy specific subsections that convey our method related observations and support them with related work when possible. Next, we address the challenges encountered during the identification of primary studies, followed by the common reporting shortcomings observed in these studies. These subsections are accompanied by their respective summaries or a call to action for the community. Finally, we outline our proposed research agenda, informed by the study's results and our own experiences during the research process.

5.1. Case Studies

In our primary studies case study research is the most prevalent research strategy. However, within the broader field of software engineering, there has been notable confusion about the attributes that define a case study [187]. To investigate the presence of this "label confusion" in our identified case studies, we examined the aspects required for a case study. Recent work by Wohlin has provided a more precise definition of a case study for the software engineering community, emphasizing that *"the two aspects most clearly distinguishing a case study from other types of research studies are "contemporary phenomenon" and "real-life context" "* [187]. All studies we classified as case studies satisfied these two aspects.

Further refinement in the identification of case studies has been proposed by Rainer and Wohlin [188]. They proposed a simple indicator for identifying case studies in software engineering. The effectiveness of this indicator suggests that interview-based data collection is very common in software engineering case studies. However, as Rainer and Wohlin cautioned, an overreliance on interviews combined with insufficient data triangulation might lead to case studies being perceived predominantly as interview-based research. Without careful consideration, the research community could inadvertently narrow the concept of a case study exclusively to interview methods, thereby limiting the design and methodological diversity of future studies. This could reduce our concept of case study to only interview studies and could limit the design of future case studies since the community is not considering other possibilities.

To explore the working definition of case study in our primary studies, we applied this indicator to our sample of 18 explicit case studies. We found that 17 satisfied the indicator criteria. The sole exception was a case study that did not conduct interviews but employed an experiment [100]. Additionally, we had 9 studies that we interpreted as case studies. Among these, 2 exceptions did not involve interviews; instead, they analyzed documents created during the case study [93, 82]. Furthermore, two more primary studies claimed to be case studies and involved interviews but were better classified under other research strategies [71, 106].

Although our sample size is relatively small for evaluating the broader issue of labeling, the indicator also works in our sample. This highlights a potential concern. The tendency within the community to conduct case studies solely with interview-based designs could restrict methodological diversity and prevent other methods of collecting data from taking place in case studies. Moreover, the limited diversity in data collection methods contradicts established principles of what defines a robust case study. In fact, another essential attribute of case studies, highlighted in established guidelines [189, 42], is the use of multiple data sources or data triangulation. We analyzed this aspect in our primary studies as well and found that approximately half (13 out of 27) relied on a single method of data collection. While this does not directly measure the extent of data triangulation, it underscores a threat and potential area for methodological improvement in future case studies.

Summary:

Case studies are the most common research strategy in our primary studies, but there appears to be some "label confusion" in the software engineering community about what defines a case study. While case studies should involve a "contemporary phenomenon" and a "real-life context," many in the field tend to rely heavily on interviews as the primary data collection method. Nearly all case studies in our sample aligned with a proposed indicator that demonstrates the ubiquity of interviews in case studies, which might narrow methodological diversity. Additionally, an important aspect of robust case studies is data triangulation, involving multiple data sources. However, about half of the case studies we reviewed used only a single method of data collection—primarily interviews. This reliance on interviews suggests a potential methodological limitation, and future case study designs could benefit from incorporating a broader range of data collection methods to achieve more comprehensive triangulation.

5.2. Ethnography Studies

We have 12 studies that employ ethnography as their research strategy, with seven of them explicitly stating their method as such. For another 5 studies, we interpreted the research strategy to be ethnography. However, these studies are conducted to serve software engineering research purposes and seem quite different from more traditional ethnography studies that we see in social sciences that focus on fieldwork and immersion [42]. The human involvement that we have in software engineering makes ethnographic methods beneficial for SE research [190, 172]. However, the scope of events we investigate is generally smaller compared to the subjects of social sciences. Zhang et al. explained this scope diversion: "When coming to SE, ethnography tends to study its research objects in a relatively constrained and smaller scope, e.g., a project and a software company. In such cases, a panoramic vision may turn to be not a pursuit" [172].

Given that one of the primary aims of this method is to become familiar with a culture [42], conducting a fully ethnographic study with complete cultural immersion may not always be necessary or purposeful. We think this might explain why we saw a lot more case studies than ethnography studies.

We saw two different approaches to this challenge in our studies. The first type is studies that try to adhere to traditional ethnography as much as they can by doing on-site observations through immersion while collecting data by observation and field notes (e.g., see Shah et al. [122]). Second is the studies that mainly collect their data by interviews, but their approach to the problem space is in the realm of ethnography [116, 121]. In these cases, while they may not fully engage in the immersive, observational aspect of ethnography, they apply ethnographic techniques to understand the cultural and social dynamics within their research context. This distinction is significant as it highlights a more flexible application of ethnographic methods. The authors of some primary studies seem to be aware of these differences with applying ethnography, therefore they represent their studies as "ethnographically informed" studies (e.g. Shah et al. [120] or Romano et al. [36]).

A systematic literature review study on ethnographic methods in software engineering by Zhang et al. synthesized challenges for applying ethnography in software engineering research [172]. The challenges they identify for applying ethnography are divided into design, execution, and reporting phases. We review our set of ethnographic studies considering these phases.

For the design phase, Zhang et al.[172] consider scope divergence to be a problem as we have demonstrated at the beginning of this subsection. In addition, selecting the appropriate study duration for these studies is also a point of consideration. The median duration for the studies investigated by Zhang et al. was 8 months [172] and a considerable amount of studies were under 6 months (36%) which is the recommended minimum duration by social sciences [191]. We had 5 ethnography studies that reported their study duration: all of them were shorter than 6 months, and the longest reported duration was 5 months. The median duration for software testing studies appears to be shorter than that of software engineering studies. While this may suggest a potential area for improvement, it is important to consider that the scope of software testing studies may be more limited compared to broader software engineering studies, which could account for the shorter study durations. Additionally, another study design related observation we made in Section 4.4 was that the primary studies using the ethnographic research strategy lacked a common framework or guide for conducting ethnographic research.

We did not directly assess the execution challenges in our primary studies, as they provided limited information on this topic. The challenges identified by Zhang et al. in this context are "collaboration with participating company", "balance of listening and observation", and "researcher-participant relationship" [172]. The challenge of the reporting phase was to provide thick descriptions [192] in a page-limited medium. It is precisely this thick description that is a signature of ethnography [172] and provides value.

We investigated whether the challenge of reporting thick descriptions persisted in our primary studies. Even though our primary studies did not specifically acknowledge page limitations as a problem in reporting ethnography, we see approximately twice the amount of journal articles in the ethnography category compared to the overall list of primary studies (21.5% overall and 41.6% for ethnography studies). One way to allow for more consistent reporting of thick descriptions is to publish in a journal format [172]. Another is to consider the eventual reporting of the study and carefully design the scope of the investigation accordingly.

To enhance the potential of ethnographic studies in software testing, we offer the following suggestions. Gaining a deep understanding of a method by following established guidelines allows a clearer recognition of its strengths and potential challenges, ultimately increasing the effectiveness of a study. The ethnography studies that we investigated did not have any common guiding study that was being utilized.

To leverage the benefits of ethnography, researchers would need to conduct longer and more frequent field studies. These extended efforts will allow researchers to capture the evolving dynamics of their subjects, leading to more impactful and lasting contributions to the field. In addition, careful consideration should be given to providing thick descriptions while staying mindful of reporting limitations. By adopting these practices, researchers can significantly improve the depth and quality of their ethnographic studies, leading to more insightful outcomes.

Summary:

Ethnographic methods in software engineering research are adapted from traditional social sciences to focus on smaller scopes, such as projects or companies, and they are valuable for understanding human dynamics. However, challenges such as scope divergence, short study durations, execution complexities, and difficulty in producing detailed "thick descriptions" limit their effectiveness. To enhance ethnographic studies in SE, researchers should adopt standardized guidelines, conduct longer and more frequent field studies, and carefully design their scope to align with reporting constraints, enabling deeper insights and more impactful contributions to the field.

5.3. Grounded Theory Studies

Grounded theory is a research method that employs data analysis for the systematic generation of theory [193]. It was introduced by sociologists Glaser and Strauss [49] as a result of their collaborative research. Due to the differences in their opinions about how to apply the method, two variants to grounded theory have emerged [194]. These are known as Glaser's version of grounded theory (often referred to as the classical version) and Strauss's

version, known as Straussian grounded theory [195]. The evaluation of grounded theory has its own distinct history, and additional variations of the method have emerged over time. While many resources are available on how to apply these approaches, the choice of which variant to use leads to significant differences in application [194].

For example in Glaserian GT, rather than starting with a specific research question, the researcher explores an area of concern for the subjects, allowing the problem to emerge through data collection and analysis. This is beneficial for the discovery of new, previously unrecognized problems leading to new theory development. This variant does not use axial coding after open coding. Instead, open coding is applied until a core category emerges, after which the researcher moves on to selective coding by using the core category [194, 49]. In contrast, in Straussian GT the researcher begins with a research question, which serves as a guiding statement for studying a particular phenomenon much like in most other research methods. The downside is that it may overlook the subjects' primary concerns due to preconceived ideas. Therefore for any grounded theory study, choosing and stating which variant to utilize is an important factor [194, 50].

We have investigated our primary studies regarding the variant of grounded theory (GT) they used. We started with what was directly reported by the authors of our primary studies. Due to differences in their application, it is possible to infer which variant is being used as well. Therefore, in addition to what was reported, we used the key differences explained above (existence of research questions and axial coding) to infer the grounded theory variant in use.

Among the 25 primary studies that used grounded theory, 21 explicitly stated that they were using grounded theory, and 4 studies were interpreted as such by us. Within the 21 explicit grounded theory studies, 3 specified Glaserian GT, 5 specified Straussian GT, 1 specified a recent variant called socio-technical grounded theory [196] specifically tuned for software engineering and 12 did not specify a variant. In our 3 Glaserian GT studies (in which we do not expect to see explicit research questions or axial coding) 2 of them report research questions, and one of these also includes axial coding. In our 5 Straussian GT studies, where we expect to see research questions and axial coding, 2 of them report research questions but all of them report axial coding. In the 12 studies that did not specify a variant, 4 report research questions, and 7 report axial coding. Using this information, we can deduce that almost all of the studies that did not specify a variant (besides 1 that neither reports research questions nor axial coding) are likely to be Straussian GT studies.

In reviewing 25 primary studies, we found that most lacked clear identification of the specific grounded theory variant used. This is significant because the choice of variant has a direct impact on method application. While some studies were clearly aligned with a specific version, others showed inconsistencies, such as the use of research questions or axial coding, when they might not be expected. These issues, along with a general lack of detail, suggest that grounded theory may have been used more as a "methodological rationale," a point highlighted by Charmaz, who developed a constructivist approach to grounded theory:

"Numerous researchers have invoked grounded theory as a methodological rationale to justify conducting qualitative research rather than adopting its guidelines to inform their studies" [51].

Ad-hoc approaches to grounded theory might suggest that what is being conducted is not actually grounded theory but rather a way to refer to coding qualitative data [8] which is only a part of this method.

Moreover, while conducting the common resource analysis for RQ2, we found software engineering related guidelines for grounded theory. Stol et al. analyzed the use of grounded theory in software engineering and offered guidelines for reporting software engineering grounded theory studies [8]. Hoda introduced a software engineering specific variant to grounded theory [196]. Hoda's Socio-Technical Grounded Theory (STGT) divides the grounded theory process into two stages. The first stage focuses on exploration and is accessible to researchers new to grounded theory, as many in the engineering field are, as it does not require full theory development. The second stage provides guidelines for selecting theory models to structure and develop observations into a coherent theory, addressing a common gap in engineering researchers' familiarity with theory modeling. By offering concrete guidance, Hoda's framework makes grounded theory more practical and applicable for software engineering researchers. Neither of these developments are reflected in our common resource analysis. We attribute this to the relatively recent publication of these studies (2016 for Stol et al.'s paper [8] and 2021 for Hoda's work [196]), as only 9 out of the 25 studies in our analysis were published after 2016.

Summary:

Clear and transparent reporting is essential in grounded theory research. In our study, we observed that most primary studies did not specify the variant of grounded theory used, leading to inconsistencies and ambiguity in their methodological applications. For example, some studies that explicitly stated they were using Glaserian grounded theory included elements like research questions or axial coding, which are not aligned with its guidelines. Similarly, many studies that did not specify a variant showed characteristics typical of Straussian grounded theory. This lack of clarity potentially undermines the reliability of their findings. These observations align with concerns raised in the literature that grounded theory is sometimes used more as a "methodological rationale" rather than being rigorously applied as intended. Our findings emphasize that failing to specify and adhere to the chosen variant introduces methodological ambiguity and can blur the distinction between grounded theory and general qualitative coding approaches.

5.4. Knowledge Gathering Studies

As we described earlier, knowledge gathering studies are studies that utilize qualitative methods that only focus on understanding or describing a situation. This research strategy emerged as we went through our list of primary studies because we kept seeing studies that were qualitative but did not necessarily adhere to any research strategy we previously identified in the literature. Furthermore, this category mainly had exploratory studies which could be related to not directly adhering to other research strategies. We observed that this category of studies sometimes had a lower quality of reporting in terms of research design and data analysis compared to others. Knowledge gathering studies had the lowest ratio of explicitly stated data analysis (39%) method compared to other research strategies (case study 59%, ethnography 67%, grounded theory 80%, engineering research 69%) with more than 3 occurrences. Flexibility of method might be desired for exploratory studies; however, this should not translate into not ensuring rigor in qualitative methodology. Exploratory studies are valuable, particularly when addressing research topics that have not yet been thoroughly investigated. Sharing preliminary results from these studies can also be beneficial to the community. However, it is important to clearly indicate when the findings are preliminary, such as when the theory being developed is still in its early stages. When a study is exploratory, this should be explicitly stated, allowing readers to adjust their expectations accordingly. This transparency enables other researchers to build on these findings rather than dismissing the study due to concerns about the research process.

Summary:

Knowledge gathering studies focus on understanding specific contexts, providing valuable insights and descriptions. For exploratory knowledge gathering studies, clearly stating their exploratory nature and the preliminary status of findings enhances transparency and allows the research community to build on their contributions. For non-exploratory knowledge gathering studies, ensuring methodological rigor and clear reporting of research design and data analysis strengthens their impact and reliability.

5.5. Choosing the Correct Research Methodology

Selecting an appropriate research methodology is critical for effectively addressing the specific nature of a research question. In software engineering, a socio-technical discipline, the diversity of research problems demands varied approaches [197]. Not every research question or subdomain is inherently suitable for qualitative exploration. For example, technology-driven areas such as automated test generation or fault detection often emphasize computational metrics and technical experimentation [198], which naturally limit the applicability of qualitative strategies. This is reflected in our results, where subdomains like "Test Oracles", "Web Application Testing" and "ML and Predictive Modelling" did not yield any qualitatively focused research in our investigations (see RQ3, Section 4.5).

The feasibility of employing certain research strategies also varies across subdomains. Case study or grounded theory, for instance, is well-suited for examining human-centric or organizational aspects, as demonstrated by the number of qualitative studies in areas like "Testing in Practice" and "Human Aspects" (see Figure 10 and Section 4.6). In contrast, these strategies are less applicable in purely technical domains where there is limited interaction

with human factors. Our findings emphasize that such areas remain primarily quantitative in focus, underscoring the challenges of applying qualitative methodologies universally across software engineering research.

Summary:

Aligning the research question with an appropriate methodology is essential for generating meaningful insights. The results of this study provides insights for researchers in choosing strategies that fit both the nature of their research questions and the characteristics of the subdomains they are investigating.

5.6. Methodology Specific References

Based on the common resource analysis presented in RQ2, it is evident that some of the commonly referenced works across various research strategies are not directly related to general or software engineering specific methodological guidelines. The common resource analysis for Ethnography (Table 6) and knowledge gathering (Table 10) studies do not involve prevalent methodological guidelines, while grounded theory studies only include general qualitative guidelines on software engineering or non-software engineering specific guidelines on grounded theory.

This observation highlights a potential gap in the adoption or availability of domain-specific methodological sources. We propose three potential causes for this issue:

General guideline work exists but lack specificity to software engineering. Many widely recognized guidelines, such as those by Glaser and Strauss on grounded theory [49], or general qualitative research methodologies [9], are designed for broader application across disciplines. While these works offer valuable insights, their lack of direct contextualization for software engineering can limit their perceived relevance or usability in addressing software testing-specific challenges. This does not seem to be the case, as we have been able to detect software engineering specific guidelines for all research strategies we conducted the resource analysis on besides knowledge gathering. As knowledge gathering is a research strategy defined by us, we do not expect to see relevant guidelines for it.

Software engineering-specific guideline works exist but are either unpopular or too recent. While some methodological guidelines specifically targeted at software engineering do exist, they may not yet have achieved widespread adoption, or they are too recent to appear in our analysis. We have conducted resource analyses on case studies, ethnography, grounded theory, engineering research, and knowledge gathering. We detected software engineering specific guidelines for the following research strategies:

Case study. Runeson and Höst [2] have created software engineering specific guidelines for case studies published in 2009. This is the most prevalent resource in our resource analysis for case studies.

Ethnography. Zhang et al. [172] have created software engineering specific guidelines for ethnography in 2019. However, we did not observe this resource to be prevalent in our resource analysis.

Grounded theory. On top of the many existing guidelines for grounded theory, there are also software engineering specific guidelines by Stol and Fitzgerald [8] and Hoda [196], which were published in respectively 2016 and 2021. We did not observe these guidelines in our resource analysis.

The common pattern that emerges from these resources is the disproportionate adoption of domain-specific guidelines relative to their recency. While it is possible that newer guidelines will gain more widespread adoption over time, we aim to actively support their adoption by highlighting their existence. This aligns with our goal of assisting researchers entering the domain of qualitative methods in software engineering.

Many works exist without a standout reference or are published under multiple titles or versions. In some cases, the abundance of related works can dilute the prominence of any single resource. Additionally, certain guidelines or resources may exist in multiple forms or titles, further complicating their identification and consistent use. This fragmentation could prevent researchers from converging on a singular, well-established methodological reference. This phenomenon would also hide such resources from our common resource analysis. A good example would be the grounded theory research strategy. As we have elaborated earlier, there are many versions of guidelines for this method from the same authors.

Summary:

For most of the research strategies, we did not identify one, or a set of prevalent methodological guidelines. We hypothesize that some guidelines might not be specific enough to software engineering, some works are too recent to have been cited frequently, or many works exist causing no standout reference work.

5.7. Qualitative Evidence as an Alternative for Data Analysis

The remainder of this section continues with our broader discussion points that are consistent across the entire set of primary studies. Throughout our cataloging attempts of the primary studies, we have observed a pattern in which studies would employ an extensive sharing of their raw qualitative data in the form of participant quotes. This is used as a way to support the arguments or findings of these studies by demonstrating direct links between the arguments and evidence. We have decided to refer to this "qualitative evidence" as a data analysis method for the purposes of this study since the extensive use of it in some studies seems to replace the need to aggregately analyze the entire qualitative data or diminish the need to report it in that manner. We have detected 24 studies using qualitative evidence, which makes it the 3rd most popular data analysis method in our primary studies. For two of these studies, this was the only method of data analysis that we were able to detect. The use of participant quotations is highly encouraged in ethnography and grounded theory methodologies [42]. However, relying too heavily on quotations as the primary means of presenting qualitative data is not the intended practice. When quotes dominate, data synthesis or theory building may be neglected, as arguments are built primarily around individual quotes. This issue has also been identified in other studies that investigated qualitative software engineering work.

For example, Stol et al. in *Grounded Theory in Software Engineering Research* write: "In many cases, the study's results are structured based on a set of research questions, which are answered in detail using quotes from participants. This type of output is quite common for those studies that only used coding techniques, but do not make a theoretical contribution [8]." Similarly, Zhang et al. notes that "Thick description should be used to portray a variety of scenes and episodes during a software process. But verbatim quotations should not be used everywhere, limited verbatim quotations should be used in some uncommon details [172]."

We found that some of our primary studies suffer from the same concerns that are highlighted by Stol et al. [8] and Zhang et al. [172]. We argue that these problems could be related to the lack of intimate knowledge of the intended outcomes of qualitative methods, which can be mitigated by using already existing guidelines for methodological rigor.

Summary:

The use of participant quotations to support arguments is common practice, valued for its ability to establish clear links between findings and data. However, an overreliance on quotations as the primary means of presenting data can limit deeper analysis, synthesis, or theory building. To address this, researchers should balance the use of quotations with thorough data aggregation and interpretation. Following established methodological guidelines can help ensure that qualitative methods are used effectively to achieve their intended outcomes, enhancing the overall quality and contribution of the research.

5.8. Study Discovery Challenges

Secondary studies like ours usually investigate studies on a common topic. The amount of research done around software testing is quite extensive. Therefore, conducting a secondary study on software testing is a huge endeavor. In our case, we specifically looked at qualitatively focused software testing studies. While conducting our initial search for papers in databases, we observed that author-assigned keywords were not working well for our use case. These keywords sometimes do not reflect what methods a paper uses. Backward and forward snowballing helped with this problem immensely and allowed us to gather a respectable set of primary studies. We think there are a couple of reasons why methods are sometimes overlooked in author-assisted keywords.

- Usually, there is a limit to the number of keywords an author can assign. Authors might prioritize keywords that reflect the primary findings or concepts of the study, possibly leaving out or oversimplifying the methodological aspects. Furthermore, keyword classification systems, such as the ACM Computing Classification System [199], can also reduce the ability to convey method-related information through keywords.
- Authors use keywords that are more likely to be used in search queries to ensure visibility.
- Authors might assume that the method is understood within the context of the study, therefore omitting specific references to it in their keyword selection.

Call to Action:

We call to both authors and the community to encourage mentioning one's research strategy, and potentially the primary data collection and analysis method, in the author assigned keywords. Fostering a culture of explicitly naming one's method or strategy helps the community to create and uphold standards in the review process [53] and encourages the authors to actively reflect on the chosen approach. Being encouraged to explicitly identify and name one's research strategy, also helps authors to discover relevant guideline papers to use to learn how to rigorously apply their chosen method.

5.9. Shortcomings of Reporting

We have observed problems with reporting in some studies. All of these problems are tied to methodological rigor; however, some reporting problems cause bigger concerns than others. We map our primary studies on three dimensions, which are "research strategy", "data collection", and "data analysis". For simplicity, each study is assigned one strategy. In data collection or analysis, it is possible to have more than one method. When a dimension of a study was not stated or described in detail, we tried to interpret it from the contents of the paper at hand.

We found that out of 102 primary studies, 65 explicitly stated their data analysis method. For 31 studies, we were able to interpret the method(s) of analysis from the authors' descriptions. For 6 studies we did not have enough information about data analysis to make a confident interpretation, and 2 studies had qualitative evidence as their only data analysis method. As discussed earlier, we consider qualitative evidence to be the use of direct quotes to drive the arguments of a study without additional data analysis on the subject. This is normal for points that are details, but we consider it as potentially problematic, since a quote does not necessarily reflect the overall data points, and overuse it has been discussed as problematic in the literature as well [172, 8]

Although uncommon in our set of 102 primary studies, we had to interpret the data collection method for three studies. We argue that, for transparency and replicability, the data collection method should always be explicitly stated and never left open to interpretation.

Moreover, there are primary studies in which relevant information about study participants was also lacking. However, we find this harder to evaluate, as it depends a lot more on the specific study context, and it is possible that in some cases some information on this might have been omitted to protect personal data. However, for many studies, the information regarding the relationship of participants with software testing is crucial for the study. These could be roles, amount of experience, frequency of testing activities, or which testing activities. This information is often necessary for the reader to be able to understand the background of the study participants, which in turn enables the reader to put the study results into context.

The combined observations may indicate shortcomings in reporting how data is collected and processed in some qualitative research studies on software testing. Especially related to clearly indicating data collection and analysis methods. When data collection or analysis methods are not reported in sufficient detail in research articles, several critical problems can emerge. The lack of transparency makes it challenging for readers to evaluate the reliability and replicability of the findings, which can erode trust in the study. Mendez et al. [200] have also emphasized the importance of trustworthiness in qualitative studies within software engineering. In their guidelines for open science, they recommend sharing data analysis procedures and codebooks to enhance transparency.

Furthermore, without an adequate description of the methodology, potential biases or limitations cannot be accurately evaluated, which may lower the academic value of the paper. Inadequate reporting reduces the contribution of the study to the scientific community and weakens its overall impact.

Call to Action:

We urge researchers engaged in qualitative work to maintain a higher standard of reporting, to increase the trust of the community in the results of qualitative studies. This starts with explicitly indicating the research strategy, data collection, and analysis methods, not exclusively relying on quotes to support interpretation and outcomes, and reporting the relevant professional experience of the participants.

5.10. Research Agenda

Based on the insights from our systematic mapping study, we present a refined research agenda aimed at tackling the challenges and seizing the opportunities identified in qualitative software testing research. This agenda emphasizes improving methodological rigor and reporting practices while also encouraging the exploration of under-researched areas to increase the impact and value of qualitative studies in this domain.

Expanding Qualitative Methods to Technical Domains

Our study identified a lack of qualitative research in areas such as "Test Oracles", "Performance Testing", and "Concurrency". These subjects appear in the mapping study by Salahirad et al. [29] indicating that they are studied frequently in general. The fact that these subjects do not appear in our primary studies indicates that they may not have been investigated with qualitative focus. These gaps present opportunities to conduct new studies to explore the socio-technical aspects that shape the application of these areas in real-world scenarios.

Future studies could apply qualitative methods to these technical areas, focusing on how practitioners engage with and perceive these advanced techniques. For example, qualitative research could investigate how testers and developers perceive the effectiveness and limitations of Test Oracles in practice, exploring questions such as:

- *How do practitioners decide on oracle selection, and what are the trade-offs they consider when making oracle-related decisions?*

In Performance Testing, researchers could explore practitioner workflows and decision-making processes by considering:

- *How do practitioners set objectives for performance testing, and what strategies do they use to design performance test cases?*

Similarly, in Concurrency Testing, qualitative methods could address questions like:

- *How do teams manage the complexity of concurrent systems, and what best practices do they follow to detect race conditions?*

Expanding Qualitative Methods to ML and Predictive Modeling for Software Testing

Our study also highlights a gap in qualitative research focused on Machine Learning (ML) and Predictive Modeling in the context of software testing, since we were unable to discover any qualitative studies conducted on machine learning or predictive modeling but the subjects appear in the taxonomy of Salahirad et al. [29]. With machine learning technologies increasing their domain of application, many tools and techniques have emerged that are relevant for software testing. These techniques and tools can help with tasks such as test case preparation, predicting and classifying software vulnerabilities, repairing defects, and more [201, 202]. How these developments impact practitioner workflows and approaches to testing activities is important to understand for software testing research. Future research could use qualitative methods to investigate:

- *"Whether and how the adoption of generative machine learning tools has altered the roles and responsibilities of testers and developers."*

Moreover, understanding how testers adapt their workflows when integrating machine learning tools is equally important. Studies could explore questions such as:

- *"How do practitioners adapt their testing strategies to incorporate machine learning tools, and what challenges do they face in doing so?"*

Another important area of focus is the training and development of testing professionals in light of these advancements. Research could address questions such as:

- *What new skills or knowledge are required for testers to effectively work with data-driven tools, and how are organizations addressing these training needs?*

Finally, studies could explore broader organizational perspectives by asking:

- *What factors influence the adoption of machine learning techniques in software testing, and how do organizations decide when and where to implement these tools?*

By employing qualitative approaches to these topics, researchers can uncover valuable insights into the evolving landscape of software testing, providing a deeper understanding of how ML is reshaping testing practices and how organizations can better support these transitions.

In Section 4.5, we identified areas of software testing that lack qualitatively focused studies, which served as the foundation for our research agenda. The research topics included in our agenda are directly derived from our analysis, and we have provided example research questions to illustrate how these topics can be explored from a qualitative perspective. By applying qualitative approaches to these domains that we discussed, researchers can uncover valuable insights into the decision making, challenges, and practical experiences of practitioners, offering a more intimate understanding of how these advanced testing techniques are perceived or applied in industry and practitioner-oriented contexts. The research questions that we suggest are meant to exemplify and inspire rather than to be used as literal directives for future studies. Our aim here is to acknowledge the potential contributions of studies that focus on practitioner expertise to learn more about industrial applications of these techniques, which in turn allows software testing research to develop more ways to support and contribute to these practices.

Methodical considerations for future studies

Our mapping study has highlighted the key subjects for investigation, and the research agenda we propose offers examples of research questions at the intersection of human and technical aspects of software testing. In this section, we briefly underline methodological recommendations that could be advantageous in addressing these questions.

Our analysis revealed that ethnographic studies in software testing often did not involve extensive fieldwork and when field studies were conducted, they were typically shorter than what is recommended in both social sciences and software engineering. Longer ethnographic field studies could offer deeper insights that may not be achievable through interviews alone. Therefore, we recommend considering longer ethnographic studies for future research.

Moreover, as we identified case studies to be the most common qualitative method in our primary studies, we would like to suggest a variant that would be beneficial for future qualitative work.

Longitudinal case studies, which allow for data collection over extended periods, provide a stronger ability to identify changes and agents of change [203]. Longitudinal studies have been previously suggested to the software engineering community but the necessary resources to carry out such studies are rarely allocated. Although they are resource intensive, these studies could track the evolution of testing practices within specific organizations or teams, offering valuable insights into the long-term impacts of workflow shifts or tool integration. We believe this approach would be particularly useful for studying the evolution of software testing with the integration of machine learning tools into practitioner workflows.

As new technologies are adopted in testing environments, there are likely to be multiple ways of working and varying levels of adoption. To explore these differences, future research could benefit from cross-case studies that compare testing practices across different organizations or projects, providing a broader understanding of how different contexts influence the use and effectiveness of new testing methodologies.

While both extended ethnographic field studies and longitudinal case studies can yield valuable insights, it is crucial to recognize that such qualitative studies require substantial time commitments. To strengthen ethnographic field studies and longitudinal case studies, we as a research community have to address this mismatch with our incentive structure. For example, funders and institutions could dedicatedly support longitudinal work that encompasses multiple, subsequent PhD students.

6. Threats to Validity

This section outlines the potential threats to the validity of our study. We acknowledge that various limitations might have affected our findings and have taken steps to mitigate these where possible. In the following, we detail the key threats related to database selection, search completeness, study inclusion criteria, information filtering, and the potential for subjectivity in data extraction and categorization.

Database limitations:

Finding the majority of relevant studies is of great importance for mapping studies. We selected SCOPUS [30] from Elsevier as our search database, since it indexes all major digital libraries that are relevant to software engineering research like IEEE Xplore and ACM Digital Library [31, 32]. This is a good start, but it does not guarantee us the coverage we strive for. Therefore we supplement with Google Scholar [204] as a search engine in parallel to SCOPUS to mitigate this threat. We detected that five of our primary studies were not directly accessible by SCOPUS and were found by Google Scholar. Any potential indexing delay in SCOPUS is also mitigated by complementing the search with Google Scholar and snowballing to identify additional recent relevant studies through citations.

Search completeness:

Because of the method based scope of this study, ensuring proper search terms was not trivial. We have refined our search terms until a set of known primary studies were detected successfully. However, the studies we were able to retrieve are still limited by the content provided by authors in studies "Title&Keywords&Abstract" which are sometimes not completely representative of study contents. Although we had a list of refined terms for retrieving articles with qualitative methods, there is no guarantee that we were able to detect all qualifying qualitative studies. We mitigate this threat with search term refinement.

Exclusion of moderately qualitative studies:

Moreover, there is another completeness related threat regarding inspecting qualitative work done in software testing. In many studies, qualitative methods are used as an auxiliary research method. Potentially, there are things that can be learned from inspecting these studies as well; however, for this study, we left such work out of scope by only considering studies where the main research method is qualitative.

We believe that working with primary studies that use qualitative methods as their main research focus also allows us to detect qualifying studies with more precision. To arrive at our list of primary studies, we have checked about a thousand articles, including snowballing. We consider a general study that includes all uses of qualitative methods in software testing to be unfeasible due to the scope, as the number of studies to inspect and evaluate would be much higher.

Risks of information filtering by inclusion criteria:

By focusing on software testing research, we may overlook interdisciplinary studies where qualitative insights on software testing were secondary but still significant. This restriction might result in a bias towards more software testing community-focused outputs. This was a necessary scoping measure that allowed us to look for studies systematically, leaving out studies that are not directly concerned with software testing.

We are potentially missing relevant information from the gray literature or non-English studies as these were not included. Furthermore, the exclusion of multiple versions of a study, opting only for the most comprehensive version, could potentially lead to the loss of details presented in earlier work. However, in our impressions, this is not the case.

Lastly, while our decision to impose no time restrictions was intended to capture the full spectrum of qualitative research focused on software testing, it may have inadvertently included older studies whose methodologies and contexts may no longer be relevant to current practices.

Subjectivity in data extraction and categorization:

A primary threat to the validity of our study arises from the processes of data extraction and categorization. The initial data extraction for study selection categorization was performed by the main interpreter, which could raise concerns about potential bias in selecting and interpreting relevant data from the primary studies. The risk of subjective interpretation cannot be completely eliminated, and this subjectivity could affect the consistency and precision of the extracted data, influencing the study findings. However, multiple interpreters independently evaluated data extraction samples (10% to 20% for each of the three additional interpreters) in each stage, and discrepancies were discussed to reach a consensus. This collaborative approach serves as a mitigation strategy to ensure greater accuracy and reduce bias, increasing the robustness of the findings.

Additionally, the categorization of data based on research strategy, data collection methods, and data analysis techniques involved a level of interpretation, particularly when these categorizations were not explicitly stated in primary studies. This interpretive aspect introduces the possibility of miss-classification, as different researchers might have varying perspectives on categorizing the same data. To mitigate this risk, we have taken various actions throughout the study. We have reported the definitions of concepts for all aspects of categorization in Section 4.2.1 for transparency and to ensure that all researchers in the study share the same understanding of concepts. We performed inter-rater reliability analysis to detect disagreements. In any case of disagreements regarding concept definitions or data analysis, we discuss these disagreements until a consensus is reached and update the relevant segment.

We acknowledge that our interrater reliability process was conducted using a single round of sampling with Cohen's Kappa scores calculated only once for each interrater. While the moderate agreement scores were resolved through iterative discussions, the lack of repeated rounds or larger sample sets may affect the reliability and representativeness of our results. Future work could benefit from a more robust approach, such as additional sampling rounds to improve reliability and mitigate potential biases.

Furthermore, studies can potentially encompass multiple aspects that our assessments might overlook. We assign one research strategy to each study, identifying and proceeding with the strategy that is dominant in each study. However, elements of other strategies might be present. This could limit the depth of our analysis, particularly in capturing the diversity and complexity of qualitative research methods in software testing.

7. Conclusion

This systematic mapping study aimed to present a holistic view of qualitatively focused research in software testing by systematically analyzing studies that use these qualitative methodologies. Our research was guided by four key questions: What are the main qualitative methods used in software testing? (RQ1.1) What are the purposes behind these studies? (RQ1.2) What resources are commonly cited in qualitative software testing research? (RQ2) Which areas of software testing have been investigated most and least using qualitative methods? (RQ3)

Our analysis revealed case study and grounded theory to be the most frequently employed qualitative research strategies. Semi-structured interviews emerged as the dominant data collection method, with open-axial coding and thematic analysis as the most common approaches to analyzing these data. This combination of methods is well suited for gathering insights directly from practitioners. In addition, we also found a potential lack of methodological diversity, with strategies such as ethnography and action research being less likely to be utilized in software testing contexts.

The key motivation for our primary studies was to deepen the understanding of practitioner activities and processes. Many studies also aimed to capture the practitioner's perspective on their subjects. We also observed studies that used qualitative methods to investigate the human aspects of software testing, such as communication, decision-making, and team dynamics, but these were less prevalent than the former purposes.

When investigating primary studies for common resources, we found that certain foundational works, such as "Qualitative Methods in Empirical Software Engineering" by Seaman [7] and guidelines on case study research by Runeson and Höst [2], were frequently cited. These resources seem to be influential in shaping the methodological approaches used by software testing research. However, we observed a relative lack of utilization for specialized guides for method specific guides in software testing as well, which suggests that currently researchers often rely on general qualitative guidelines. We have discussed the recent emergence of method-specific studies in software engineering [8, 172, 196]. These relatively recent studies and more future work to be conducted in this area could help provide better method specific support for qualitative studies of software testing.

Our mapping showed that while areas like test automation and test-driven development are well studied, significant gaps remain in domains such as test oracles, web application testing, and the integration of machine learning in testing practices. These underexplored areas present significant opportunities for future qualitative research, particularly as these technical domains involve human decision-making that could benefit from deeper investigation with qualitative methods.

This study also highlighted a few key takeaways from the discussion of the analysis results. There is a consistent issue with methodological transparency in qualitative software testing research, particularly in reporting data analysis processes. This hinders reliability and could potentially reduce the credibility of the findings. Moreover, many

studies overly rely on participant quotes as evidence without sufficient theoretical synthesis, limiting the depth of insights that qualitative research can offer. Additionally, there is a notable imbalance in the areas where qualitative research is applied, with a heavy focus on practitioner-centered issues and less attention given to the interplay between human factors and more complex technical challenges like machine learning or test oracles. We also emphasized that qualitative research has great potential to address these areas, but this requires methodological rigor and better reporting practices.

Based on our findings, we proposed a future research agenda aimed at expanding qualitative research into underexplored technical areas, such as machine learning integrated software testing. Future studies should also seek to diversify qualitative methodologies by incorporating more ethnographic approaches, which could provide deeper insights into testing cultures and workflows. Additionally, enhancing the rigor of reporting, particularly in data analysis and theory development, would be beneficial for the results of future studies. By tackling these challenges, qualitative research can play a more pivotal role in software testing, especially in enriching academic understanding of industry practices.

Acknowledgments

This research is sponsored by the Swiss National Science Foundation (SNSF Grant 200021M 205146), and the Dutch Science Foundation NWO through the Vici "TestShift" project (No. VI.C.182.032).

Table .11. Venues with at Least Two Primary Studies

Venue	**Acronym**
International Symposium on Empirical Software Engineering and Measurement	ESEM
International Conference on Software Engineering	ICSE
International Conference on Software Testing Verification and Validation	ICST
Information and Software Technology	IST
Software Quality Journal	SQJ
Journal of Systems and Software	JSS
International Conference on Software Testing Verification and Validation Workshops	ICSTW
Agile Processes in Software Engineering and Extreme Programming	XP
IEEE Transactions on Software Engineering	TSE
International Conference on Global Software Engineering	ICGSE
Empirical Software Engineering	ESE
IEEE Access	Access
The Psychology of Programming Interest Group	PPIG
International Symposium on Empirical Software Engineering	ISESE
Asia-Pacific Software Engineering Conference	APSEC
International Conference on Evaluation and Assessment in Software Engineering	EASE
Symposium on the Foundations of Software Engineering	FSE
IEEE International Conference on Software Analysis Evolution and Reengineering	SANER
International Requirements Engineering Conference	RE
Proceedings of the ACM Symposium on Applied Computing	SAC
Conference on Software Engineering and Advanced Applications	SEAA

Table .12: List of Primary Studies

Cite Key	**Titles**
nascimento2018the-adoption[79]	The Adoption of Open Source Projects in Engineering Education: A Real Software Development Experience
tran2019test-case[80]	Test-Case Quality – Understanding Practitioners' Perspectives
votipka2018hackers[116]	Hackers vs. Testers: A Comparison of Software Vulnerability Discovery Processes
romano2016results[36]	Results from an ethnographically-informed study in the context of test driven development
nascimento2019does[81]	Does FLOSS in Software Engineering Education Narrow the Theory-Practice Gap? A Study Grounded on Students' Perception
roy2017spreadsheet[125]	Spreadsheet testing in practice

Continued on next page

Table .12 – continued from previous page

Cite Key	Titles
habchi2022a-qualitative[126]	A Qualitative Study on the Sources, Impacts, and Mitigation Strategies of Flaky Tests
jamil2020the-current[117]	The current practices of changing secure software: An empirical study
santos2018computer[76]	Computer games are serious business and so is their quality: Particularities of software testing in game development from the perspective of practitioners
prado2018towards[105]	Towards cognitive support for unit testing: A qualitative study with practitioners
sekgweleo2022understanding[69]	Understanding the factors that influence software testing through moments of translation
florea2020a-qualitative[148]	A Qualitative Study of the Background, Skill Acquisition, and Learning Preferences of Software Testers
hotomski2016an-exploratory[149]	An Exploratory Study on Handling Requirements and Acceptance Test Documentation in Industry
choma2018developers[82]	Developers' initial perceptions on TDD practice: A thematic analysis with distinct domains and languages
salman2022what[127]	What Leads to a Confirmatory or Disconfirmatory Behavior of Software Testers?
hernandez2014understanding[118]	Understanding software testers in the automotive industry: A mixed-method case study
romano2017findings[66]	Findings from a multi-method study on test-driven development
grano2020pizza[65]	Pizza versus Pinsa: On the Perception and Measurability of Unit Test Code Quality
pham2017onboarding[128]	Onboarding inexperienced developers: struggles and perceptions regarding automated testing
santos2020bug-falha[83]	Bug! Falha! Bachi! Fallo! Défaut!! What about internationalization testing in the software industry?
alegroth2022practitioners[71]	Practitioners' best practices to Adopt, Use or Abandon Model-based Testing with Graphical models for Software-intensive Systems
brandt2021developer-centric[35]	Developer-centric test amplification: The interplay between automatic generation human exploration
minhas2020regression[84]	Regression testing for large-scale embedded software development – Exploring the state of practice
betka2021extreme[106]	Extreme mutation testing in practice: An industrial case study
ismail2015a-qualitative[150]	A qualitative empirical investigation of contributing success factors for software testing outsourcing projects
kasurinen2010software[129]	Software test automation in practice: Empirical observations
pham2014enablers[34]	Enablers, inhibitors, and perceptions of testing in novice software teams
stray2022exploring[37]	Exploring human factors of the agile software tester
strandberg2019information[85]	Information Flow in Software Testing - An interview Study with Embedded Software Engineering Practitioners
lou2022testing[107]	Testing of autonomous driving systems: where are we and where should we go?
parsons2014influences[130]	Influences on regression testing strategies in agile software development environments
aragao2019testdcat[108]	TestDCat: Catalog of Test Debt Subtypes and Management Activities
bijlsma2021how-do-students[103]	How do Students Test Software Units?
shah2016robustness[86]	Robustness Testing of Embedded Software Systems: An Industrial interview Study
aniche2015does[109]	Does test-driven development improve class design? A qualitative study on developers' perceptions
afzal2020a-study[119]	A Study on Challenges of Testing Robotic Systems
vianna2019an-exploratory[151]	An Exploratory Study of How Specialists Deal with Testing in Data Stream Processing Applications
illes-seifert2008on-the-role[152]	On the role of communication, documentation and experience during system testing - An interview study
riungu2010software[131]	Software testing as an online service: Observations from practice
eldh2012robustness[87]	Robustness testing of mobile telecommunication systems: A case study on industrial practice and challenges
buchan2021causal[88]	Causal factors, benefits and challenges of test-driven development: Practitioner perceptions
shah2011outsourced[120]	Outsourced, offshored software-testing practice: Vendor-side experiences
shah2010studying[89]	Studying human and social aspects of testing in a service-based software company: Case study
taipale2006improving[132]	Improving software testing by observing practice
kasurinen2011how-test[133]	How test organizations adopt new testing practices and methods?
kasurinen2010test[134]	Test case selection and prioritization: Risk-based or design-based?
lonnberg2012back[74]	Back to school - How professional software developers develop and test software in an educational context
wiklund2012technical[90]	Technical debt in test automation
dahlstedt2006challenges[153]	Challenges in system testing—an interview study
kassab2021exploring[19]	Exploring the profiles of software testing jobs in the United States

Continued on next page

Table .12 – continued from previous page

Cite Key	Titles
gopal2022the-impact[135]	The impact of POGIL-like learning on student understanding of software testing and DevOps: A qualitative study
gopal2021student[136]	Student difficulties in Unit Testing, Integration Testing and Continuous Integration: An exploratory pilot qualitative study
deak2016challanges[121]	Challenges and strategies for motivating software testing personnel
aniche2011what-concerns[137]	What concerns beginner test-driven development practitioners: a qualitative analysis of opinions in an agile conference
dolezel2020defining[154]	Defining TestOps: collaborative behaviors and technology-driven workflows seen as enablers of effective software testing in DevOps
hotomski2018a-qualitative[110]	A qualitative study on using GuideGen to keep requirements and acceptance tests aligned
karhu2009empirical[138]	Empirical observations on software testing automation
khan2018issues[91]	Issues/Challenges of Automated Software Testing: A Case Study
deak2014a-comparative[155]	A comparative study of testers' motivation in traditional and agile software development
seth2014organizational[139]	Organizational and customer related challenges of software testing: An empirical study in 11 software companies
beer2008the-role[92]	The role of experience in software testing practice
swillus2023sentiment[4]	Sentiment Overflow in the Testing Stack: Analysing Software Testing Posts on Stack Overflow
garousi2020exploring[156]	Exploring the industry's challenges in software testing: An empirical study
lizama2020the-unpopularity[157]	The Unpopularity of the Software Tester Role Among Software Practitioners: A Case Study
kochhar2019practitioners[158]	Practitioners' Views on Good Software Testing Practices
aniche2019pragmatic[93]	Pragmatic software testing education
cruzes2019testing[94]	Testing in a DevOps Era: Perceptions of Testers in Norwegian Organisations
spadini2018when[111]	When testing meets code review: Why and how developers review tests
memar2018gamifying[77]	Gamifying information system testing– Qualitative validation through focus group discussion
santos2017would[159]	Would You Like to Motivate Software Testers? Ask Them How
alegroth2017on-the-long[140]	On the long-term use of visual gui testing in industrial practice: a case study
cruzes2016communication[78]	Communication between developers and testers in distributed continuous agile testing
scanniello2016students[160]	Students' and professionals' perceptions of test-driven development: A focus group study
pham2015communicating[104]	Communicating software testing culture through visualizing testing activity
kasurinen2014what[141]	What do game developers test in their products?
shah2014global[122]	Global software testing under deadline pressure: Vendor-side experiences
liebel2013state[95]	State-of-practice in GUI-based system and acceptance testing: An industrial multiple-case study
pham2013creating[142]	Creating a shared understanding of testing culture on a social coding site
deak2013organization[161]	Organization of testing activities in Norwegian software companies
itkonen2013the-role[143]	The role of the tester's knowledge in exploratory software testing
shah2013culture[123]	Culture and testing: What is the relationship?
ali2012testing[96]	Testing highly complex system of systems: An industrial case study
mantyla2012who[97]	Who tested my software? Testing as an organizationally cross-cutting activity
kettunen2010a-study[144]	A study on agility and testing processes in software organizations
kasurinen2009analysis[145]	Analysis of problems in testing practices
marchenko2009long[98]	Long-term effects of test-driven development A case study
martin2007good[124]	'Good' organisational reasons for 'bad' software testing: An ethnographic study of testing in a small software company
taipale2006factors[67]	Factors affecting software testing time schedule
itkonen2005exploratory[99]	Exploratory testing: A multiple case study
yu2004on-the-testing[100]	On the testing methods used by beginning software testers
runeson2003test[101]	Test processes in software product evolution - A qualitative survey on the state of practice
minhas2023checklists[112]	Checklists to support decision-making in regression testing
aniche2022how-developers[61]	How Developers Engineer Test Cases: An Observational Study
polepalle2022evidence[146]	Evidence and perceptions on GUI test automation - An exploratory study
brandt2022how-does[113]	How Does This New Developer Test Fit In? A Visualization to Understand Amplified Test Cases
irshad2021adapting[114]	Adapting Behavior Driven Development (BDD) for large-scale software systems
evans2021scared[164]	Scared, frustrated and quietly proud: Testers' lived experience of tools and automation
waychal2021practitioners[162]	Practitioners' Testimonials about Software Testing
barraood2021test[163]	Test Case Quality Factors: Content Analysis of Software Testing Websites
fischbach2020what-makes[115]	What makes agile test artifacts useful? An activity-based quality model from a practitioners' perspective

Continued on next page

Table .12 – continued from previous page

Cite Key	Titles
tyagi2017adopting[147]	Adopting test automation on agile development projects: A grounded theory study of Indian software organizations
tervonen2013outsourcing[102]	Outsourcing software testing: A case study in the Oulu area

References

[1] J. Rooksby, M. Rouncefield, I. Sommerville, Testing in the wild: The social and organisational dimensions of real world practice, Computer Supported Cooperative Work (CSCW) 18 (2009) 559–580.

[2] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empirical software engineering 14 (2009) 131–164.

[3] C. E. Brandt, A. Khatami, M. Wessel, A. Zaidman, Shaken, not stirred: How developers like their amplified tests, IEEE Trans. Software Eng. 50 (5) (2024) 1264–1280. doi:10.1109/TSE.2024.3381015.

[4] M. Swillus, A. Zaidman, Sentiment overflow in the testing stack: Analyzing software testing posts on stack overflow, J. Syst. Softw. 205 (2023) 111804. doi:10.1016/J.JSS.2023.111804.

[5] M. P. Prado, E. Verbeek, M.-A. Storey, A. M. Vincenzi, Wap: Cognitive aspects in unit testing: The hunting game and the hunter's perspective, in: 2015 ieee 26th international symposium on software reliability engineering (issre), IEEE, 2015, pp. 387–392.

[6] T. Dybå, R. Prikladnicki, K. Rönkkö, C. Seaman, J. Sillito, Qualitative research in software engineering, Empirical Software Engineering 16 (2011) 425–429.

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

[8] K.-J. Stol, P. Ralph, B. Fitzgerald, Grounded theory in software engineering research: a critical review and guidelines, in: Proceedings of the 38th International conference on software engineering, 2016, pp. 120–131.

[9] S. B. Merriam, E. J. Tisdell, Qualitative research: A guide to design and implementation, John Wiley & Sons, 2015.

[10] H. F. Wolcott, Transforming qualitative data: Description, analysis, and interpretation, Sage, 1994.

[11] J. F. Gilgun, Definitions, methodologies, and methods in qualitative family research., SAGE Publications, Inc, 1992.

[12] E. Fossey, C. Harvey, F. Mcdermott, L. Davidson, Understanding and evaluating qualitative research, Australian & New Zealand Journal of Psychiatry 36 (6) (2002) 717–732, pMID: 12406114. arXiv:https://doi.org/10.1046/j.1440-1614.2002.01100.x, doi:10.1046/j.1440-1614.2002.01100.x.

[13] S. J. Taylor, R. Bogdan, M. L. DeVault, Introduction to qualitative research methods: A guidebook and resource, John Wiley & Sons, 2015.

[14] N. K. Denzin, Y. S. Lincoln, The Sage handbook of qualitative research, sage, 2011.

[15] B. Gough, A. Lyons, The future of qualitative research in psychology: Accentuating the positive, Integrative Psychological and Behavioral Science 50 (2016) 234–243.

[16] R. W. Belk, Handbook of qualitative research methods in marketing, Edward Elgar Publishing, 2007.

[17] S. Sawyer, Software development teams, Communications of the ACM 47 (12) (2004) 95–99.

[18] J. Sutton, Z. Austin, Qualitative research: Data collection, analysis, and management, The Canadian journal of hospital pharmacy 68 (3) (2015) 226.

[19] M. Kassab, P. A. Laplante, J. F. DeFranco, V. V. G. Neto, G. Destefanis, Exploring the profiles of software testing jobs in the united states, IEEE Access 9 (2021) 68905–68916. doi:10.1109/ACCESS.2021.3077755.

[20] B. Ardiç, A. Zaidman, Hey teachers, teach those kids some software testing, in: 5th IEEE/ACM International Workshop on Software Engineering Education for the Next Generation, SEENG@ICSE 2023, Melbourne, Australia, May 16, 2023, IEEE, 2023, pp. 9–16. doi:10.1109/SEENG59157.2023.00007.

[21] N. Dziugaite, B. Ardiç, A. Zaidman, What are massive open online courses (moocs) teaching about software testing?, in: F. Lonetti, A. Guerriero, M. Saadatmand, C. J. Budnik, J. Li (Eds.), Proceedings of the 5th ACM/IEEE International Conference on Automation of Software Test (AST 2024), Lisbon, Portugal, April 15-16, 2024, ACM, 2024, pp. 204–208. doi:10.1145/3644032.3644469.

[22] K. Petersen, R. Feldt, S. Mujtaba, M. Mattsson, Systematic mapping studies in software engineering, in: 12th international conference on evaluation and assessment in software engineering (EASE), BCS Learning & Development, 2008.

[23] B. Kitchenham, S. Charters, et al., Guidelines for performing systematic literature reviews in software engineering (2007).

[24] D. Budgen, M. Turner, P. Brereton, B. A. Kitchenham, Using mapping studies in software engineering., in: Ppig, Vol. 8, 2008, pp. 195–204.

[25] E. Engström, P. Runeson, Software product line testing–a systematic mapping study, Information and Software Technology 53 (1) (2011) 2–13.

[26] C. Catal, D. Mishra, Test case prioritization: a systematic mapping study, Software Quality Journal 21 (2013) 445–478.

[27] V. Garousi, A. Mesbah, A. Betin-Can, S. Mirshokraie, A systematic mapping study of web application testing, Information and Software Technology 55 (8) (2013) 1374–1396.

[28] V. H. Durelli, R. S. Durelli, S. S. Borges, A. T. Endo, M. M. Eler, D. R. Dias, M. P. Guimarães, Machine learning applied to software testing: A systematic mapping study, IEEE Transactions on Reliability 68 (3) (2019) 1189–1212.

[29] A. Salahirad, G. Gay, E. Mohammadi, Mapping the structure and evolution of software testing research over the past three decades, Journal of Systems and Software 195 (2023) 111518. doi:https://doi.org/10.1016/j.jss.2022.111518.

[30] Elsevier, Scopus content, accessed: July 25, 2024 (2024).

[31] IEEE Author Center, Abstracting & indexing (a&i) databases, accessed: July 25, 2024 (2024).

[32] ACM, Acm partnerships with indexing services, accessed: July 25, 2024 (2024).

[33] E. Mourão, J. F. Pimentel, L. Murta, M. Kalinowski, E. Mendes, C. Wohlin, On the performance of hybrid search strategies for systematic literature reviews in software engineering, Information and software technology 123 (2020) 106294.

[34] R. Pham, S. Kiesling, O. Liskin, L. Singer, K. Schneider, Enablers, inhibitors, and perceptions of testing in novice software teams, in: S. Cheung, A. Orso, M. D. Storey (Eds.), ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE), ACM, 2014, pp. 30–40. doi:10.1145/2635868.2635925.
[35] C. E. Brandt, A. Zaidman, Developer-centric test amplification, Empir. Softw. Eng. 27 (4) (2022) 96. doi:10.1007/s10664-021-10094-2.
[36] S. Romano, D. Fucci, G. Scanniello, B. Turhan, N. Juristo, Results from an ethnographically-informed study in the context of test-driven development, PeerJ Prepr. 4 (2016) e1864. doi:10.7287/peerj.preprints.1864v1.
[37] V. Stray, R. Florea, L. Paruch, Exploring human factors of the agile software tester, Softw. Qual. J. 30 (2) (2022) 455–481. doi:10.1007/s11219-021-09561-2.
[38] C. Wohlin, Guidelines for snowballing in systematic literature studies and a replication in software engineering, in: Proceedings of the 18th international conference on evaluation and assessment in software engineering, 2014, pp. 1–10.
[39] K.-J. Stol, B. Fitzgerald, A holistic overview of software engineering research strategies, in: 2015 IEEE/ACM 3rd International Workshop on Conducting Empirical Studies in Industry, IEEE, 2015, pp. 47–54.
[40] M.-A. Storey, N. A. Ernst, C. Williams, E. Kalliamvakou, The who, what, how of software engineering research: a socio-technical framework, Empirical Software Engineering 25 (2020) 4097–4129.
[41] P. J. Runkel, J. E. McGrath, Research on human behavior : a systematic guide to method, Holt, Rinehart and Winston, 1972.
[42] J. W. Creswell, C. N. Poth, Qualitative inquiry and research design: Choosing among five approaches, Sage publications, 2016.
[43] N. Gisev, J. S. Bell, T. F. Chen, Interrater agreement and interrater reliability: key concepts, approaches, and applications, Research in Social and Administrative Pharmacy 9 (3) (2013) 330–338.
[44] B. Ardic, C. Brandt, A. Khatami, M. Swillus, A. Zaidman, Data package for "the qualitative factor in software testing: A systematic mapping study of qualitative methods" (2025).
[45] M. Harris, The rise of anthropological theory: A history of theories of culture, AltaMira Press, 2001.
[46] D. M. Fetterman, Ethnography: Step-by-step, Sage publications, 2019.
[47] J. W. Creswell, Grounded theory designs, Educational research: Planning, conducting, and evaluating quantitative and qualitative research (2012) 422–500.
[48] J. Corbin, A. Strauss, Basics of qualitative research: Techniques and procedures for developing grounded theory, Sage publications, 2014.
[49] B. Glaser, A. Strauss, Discovery of grounded theory: Strategies for qualitative research, Routledge, 2017.
[50] A. Strauss, J. Corbin, et al., Basics of qualitative research, Vol. 15, sage Newbury Park, CA, 1990.
[51] K. Charmaz, Constructing grounded theory: A practical guide through qualitative analysis, Sage, 2006.
[52] M. Staron, Action research in software engineering, Springer, 2020.
[53] P. Ralph, S. Baltes, D. Bianculli, Y. Dittrich, M. Felderer, R. Feldt, A. Filieri, C. A. Furia, D. Graziotin, P. He, R. Hoda, N. Juristo, B. A. Kitchenham, R. Robbes, D. Méndez, J. S. Molléri, D. Spinellis, M. Staron, K. Stol, D. A. Tamburri, M. Torchiano, C. Treude, B. Turhan, S. Vegas, ACM SIGSOFT empirical standards, CoRR abs/2010.03525.
[54] A. Höfer, W. F. Tichy, Status of empirical research in software engineering, in: Empirical Software Engineering Issues. Critical Assessment and Future Directions: International Workshop, Dagstuhl Castle, Germany, June 26-30, 2006. Revised Papers, Springer, 2007, pp. 10–19.
[55] C. Riessman, Narrative methods for the human sciences, Sage, 2008.
[56] N. K. Denzin, Interpretive biography, Sage, 1989.
[57] E. R. Babbie, The practice of social research, Cengage Au, 2020.
[58] J. Kontio, J. Bragge, L. Lehtola, The focus group method as an empirical tool in software engineering, in: Guide to advanced empirical software engineering, Springer, 2008, pp. 93–116.
[59] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, et al., Experimentation in software engineering, Vol. 236, Springer, 2012.
[60] S. Ahmed, R. M. Asraf, The workshop as a qualitative research approach: lessons learnt from a "critical thinking through writing" workshop, The Turkish Online Journal of Design, Art and Communication 2018 (2018) 1504–1510.
[61] M. Aniche, C. Treude, A. Zaidman, How developers engineer test cases: An observational study, IEEE Trans. Software Eng. 48 (12) (2022) 4925–4946. doi:10.1109/TSE.2021.3129889.
[62] E. Adhabi, C. B. Anozie, Literature review for the type of interview in qualitative research, International Journal of Education 9 (3) (2017) 86–97.
[63] C. Willig, W. S. Rogers, The SAGE handbook of qualitative research in psychology, Sage, 2017.
[64] M. Williams, T. Moser, The art of coding and thematic exploration in qualitative research, International management review 15 (1) (2019) 45–55.
[65] G. Grano, C. D. Iaco, F. Palomba, H. C. Gall, Pizza versus pinsa: On the perception and measurability of unit test code quality, in: IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, 2020, pp. 336–347. doi:10.1109/ICSME46990.2020.00040.
[66] S. Romano, D. Fucci, G. Scanniello, B. Turhan, N. Juristo, Findings from a multi-method study on test-driven development, Inf. Softw. Technol. 89 (2017) 64–77. doi:10.1016/j.infsof.2017.03.010.
[67] O. Taipale, H. Kälviäinen, K. Smolander, Factors affecting software testing time schedule, in: 17th Australian Software Engineering Conference (ASWEC 2006), 18-21 April 2006, Sydney, Australia, IEEE Computer Society, 2006, pp. 283–291. doi:10.1109/ASWEC.2006.27.
[68] P. Mayring, T. Fenzl, Qualitative inhaltsanalyse, Springer, 2019.
[69] T. Sekgweleo, T. Iyamu, Understanding the factors that influence software testing through moments of translation, J. Syst. Inf. Technol. 24 (3) (2022) 202–220. doi:10.1108/JSIT-07-2021-0125.
[70] L. Justesen, Actor-network theory as analytical approach, Qualitative Analysis: Eight Approaches for the Social Sciences. Thousand Oaks, Ca: SAGE (2020) 327–244.
[71] E. Alégroth, K. Karl, H. Rosshagen, T. Helmfridsson, N. Olsson, Practitioners' best practices to adopt, use or abandon model-based testing with graphical models for software-intensive systems, Empir. Softw. Eng. 27 (5) (2022) 103. doi:10.1007/s10664-022-10145-2.
[72] J. V. Seidel, Qualitative Data Analysis, Colorado Springs, Colorado, originally published as Qualitative Data Analysis, in The Ethnograph v5.0: A Users Guide, Appendix E (1998).

[73] S. Adolph, W. Hall, P. Kruchten, Using grounded theory to study the experience of software development, Empirical Software Engineering 16 (2011) 487–513.
[74] J. Lönnberg, L. Malmi, Back to school: How professional software developers develop and test software in an educational context, in: M. Laakso, R. McCartney (Eds.), Koli Calling International Conference on Computing Education Research, ACM, 2012, pp. 47–56. doi:10.1145/2401796.2401802.
[75] J. Wheeldon, J. Faubert, Framing experience: Concept maps, mind maps, and data collection in qualitative research, International journal of qualitative methods 8 (3) (2009) 68–83.
[76] R. E. S. Santos, C. V. C. de Magalhães, L. F. Capretz, J. S. C. Neto, F. Q. B. da Silva, A. Saher, Computer games are serious business and so is their quality: Particularities of software testing in game development from the perspective of practitioners, in: M. Oivo, D. M. Fernández, A. Mockus (Eds.), ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), ACM, 2018, pp. 33:1–33:10. doi:10.1145/3239235.3268923.
[77] N. Memar, A. Krishna, D. A. McMeekin, T. Tan, Gamifying information system testing-qualitative validation through focus group discussion, in: B. Andersson, B. Johansson, C. Barry, M. Lang, H. Linger, C. Schneider (Eds.), Information Systems Development: Designing Digitalization, ISD 2018 Proceedings, Lund, Sweden August 22-24, 2018, Lund University / Association for Information Systems, 2018.
[78] D. S. Cruzes, N. B. Moe, T. Dybå, Communication between developers and testers in distributed continuous agile testing, in: 11th IEEE International Conference on Global Software Engineering, ICGSE 2016, Orange County, CA, USA, August 2-5, 2016, IEEE Computer Society, 2016, pp. 59–68. doi:10.1109/ICGSE.2016.27.
[79] D. M. C. Nascimento, C. F. G. Chavez, R. A. Bittencourt, The adoption of open source projects in engineering education: A real software development experience, in: IEEE Frontiers in Education Conference (FIE), IEEE, 2018, pp. 1–9. doi:10.1109/FIE.2018.8658908.
[80] H. K. V. Tran, N. B. Ali, J. Börstler, M. Unterkalmsteiner, Test-case quality - understanding practitioners' perspectives, in: X. Franch, T. Männistö, S. Martínez-Fernández (Eds.), International Conference on Product-Focused Software Process Improvement (PROFES), Vol. 11915 of Lecture Notes in Computer Science, Springer, 2019, pp. 37–52. doi:10.1007/978-3-030-35333-9_3.
[81] D. M. C. Nascimento, C. von Flach Garcia Chavez, R. A. Bittencourt, Does FLOSS in software engineering education narrow the theory-practice gap? A study grounded on students' perception, in: F. Bordeleau, A. Sillitti, P. Meirelles, V. Lenarduzzi (Eds.), IFIP WG International Conference on Open Source Systems (OSS), Vol. 556 of IFIP Advances in Information and Communication Technology, Springer, 2019, pp. 153–164. doi:10.1007/978-3-030-20883-7_14.
[82] J. Choma, E. M. Guerra, T. S. da Silva, Developers' initial perceptions on TDD practice: A thematic analysis with distinct domains and languages, in: J. Garbajosa, X. Wang, A. Aguiar (Eds.), International Conference on Agile Processes in Software Engineering and Extreme Programming (XP), Vol. 314 of Lecture Notes in Business Information Processing, Springer, 2018, pp. 68–85. doi:10.1007/978-3-319-91602-6_5.
[83] R. E. S. Santos, J. R. Cordeiro, Y. Labiche, C. V. C. de Magalhães, F. Q. B. da Silva, Bug! falha! bachi! fallo! défaut!: What about internationalization testing in the software industry?, in: M. T. Baldassarre, F. Lanubile, M. Kalinowski, F. Sarro (Eds.), ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), ACM, 2020, pp. 29:1–29:6. doi:10.1145/3382494.3422167.
[84] N. M. Minhas, K. Petersen, J. Börstler, K. Wnuk, Regression testing for large-scale embedded software development - exploring the state of practice, Inf. Softw. Technol. 120 (2020) 106254. doi:10.1016/j.infsof.2019.106254.
[85] P. E. Strandberg, E. P. Enoiu, W. Afzal, D. Sundmark, R. Feldt, Information flow in software testing - an interview study with embedded software engineering practitioners, IEEE Access 7 (2019) 46434–46453. doi:10.1109/ACCESS.2019.2909093.
[86] S. M. A. Shah, D. Sundmark, B. Lindström, S. F. Andler, Robustness testing of embedded software systems: An industrial interview study, IEEE Access 4 (2016) 1859–1871. doi:10.1109/ACCESS.2016.2544951.
[87] S. Eldh, D. Sundmark, Robustness testing of mobile telecommunication systems: A case study on industrial practice and challenges, in: G. Antoniol, A. Bertolino, Y. Labiche (Eds.), IEEE International Conference on Software Testing, Verification and Validation (ICST), IEEE Computer Society, 2012, pp. 895–900. doi:10.1109/ICST.2012.228.
[88] J. Buchan, L. Li, S. G. MacDonell, Causal factors, benefits and challenges of test-driven development: Practitioner perceptions, CoRR abs/2101.12393. arXiv:2101.12393.
[89] H. Shah, M. J. Harrold, Studying human and social aspects of testing in a service-based software company: Case study, in: Y. Dittrich, C. R. B. de Souza, M. Korpela, H. Sharp, J. Singer, H. Winshiers-Theophilus (Eds.), ICSE Workshop on Cooperative and Human Aspects of Software Engineering (CHASE), ACM, 2010, pp. 102–108. doi:10.1145/1833310.1833327.
[90] K. Wiklund, S. Eldh, D. Sundmark, K. Lundqvist, Technical debt in test automation, in: G. Antoniol, A. Bertolino, Y. Labiche (Eds.), Fifth IEEE International Conference on Software Testing, Verification and Validation, ICST 2012, Montreal, QC, Canada, April 17-21, 2012, IEEE Computer Society, 2012, pp. 887–892. doi:10.1109/ICST.2012.192.
[91] A. Z. Khan, S. Iftikhar, R. H. Bokhari, Z. I. Khan, Issues/challenges of automated software testing: A case study, Pak. J. Comput. Inf. Syst. 3 (2) (2018) 61–75.
[92] A. Beer, R. Ramler, The role of experience in software testing practice, in: 34th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2008, September 3-5, 2008, Parma, Italy, IEEE Computer Society, 2008, pp. 258–265. doi:10.1109/SEAA.2008.28.
[93] M. Aniche, F. Hermans, A. Van Deursen, Pragmatic software testing education, in: Proceedings of the 50th ACM Technical Symposium on Computer Science Education, 2019, pp. 414–420.
[94] D. S. Cruzes, K. Melsnes, S. Marczak, Testing in a devops era: Perceptions of testers in norwegian organisations, in: S. Misra, O. Gervasi, B. Murgante, E. N. Stankova, V. Korkhov, C. M. Torre, A. M. A. C. Rocha, D. Taniar, B. O. Apduhan, E. Tarantino (Eds.), Computational Science and Its Applications - ICCSA 2019 - 19th International Conference, Saint Petersburg, Russia, July 1-4, 2019, Proceedings, Part IV, Vol. 11622 of Lecture Notes in Computer Science, Springer, 2019, pp. 442–455. doi:10.1007/978-3-030-24305-0_33.
[95] G. Liebel, E. Alégroth, R. Feldt, State-of-practice in gui-based system and acceptance testing: An industrial multiple-case study, in: O. Demirörs, O. Türetken (Eds.), 39th Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2013, Santander, Spain, September 4-6, 2013, IEEE Computer Society, 2013, pp. 17–24. doi:10.1109/SEAA.2013.29.
[96] N. B. Ali, K. Petersen, M. Mäntylä, Testing highly complex system of systems: an industrial case study, in: P. Runeson, M. Höst, E. Mendes,

A. A. Andrews, R. Harrison (Eds.), 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '12, Lund, Sweden - September 19 - 20, 2012, ACM, 2012, pp. 211–220. doi:10.1145/2372251.2372290.

[97] M. Mäntylä, J. Itkonen, J. Iivonen, Who tested my software? testing as an organizationally cross-cutting activity, Softw. Qual. J. 20 (1) (2012) 145–172. doi:10.1007/s11219-011-9157-4.

[98] A. Marchenko, P. Abrahamsson, T. Ihme, Long-term effects of test-driven development A case study, in: P. Abrahamsson, M. Marchesi, F. Maurer (Eds.), Agile Processes in Software Engineering and Extreme Programming, 10th International Conference, XP 2009, Pula, Sardinia, Italy, May 25-29, 2009. Proceedings, Vol. 31 of Lecture Notes in Business Information Processing, Springer, 2009, pp. 13–22. doi:10.1007/978-3-642-01853-4_4.

[99] J. Itkonen, K. Rautiainen, Exploratory testing: a multiple case study, in: 2005 International Symposium on Empirical Software Engineering (ISESE 2005), 17-18 November 2005, Noosa Heads, Australia, IEEE Computer Society, 2005, pp. 84–93. doi:10.1109/ISESE.2005.1541817.

[100] Y. Yu, S. P. Ng, P. Poon, T. Y. Chen, On the testing methods used by beginning software testers, Inf. Softw. Technol. 46 (5) (2004) 329–335. doi:10.1016/j.infsof.2003.09.006.

[101] P. Runeson, C. Andersson, M. Höst, Test processes in software product evolution - a qualitative survey on the state of practice, J. Softw. Maintenance Res. Pract. 15 (1) (2003) 41–59. doi:10.1002/smr.265.

[102] I. Tervonen, A. Haapalahti, L. Harjumaa, J. Similä, Outsourcing software testing: A case study in the oulu area, in: 2013 13th International Conference on Quality Software, Najing, China, July 29-30, 2013, IEEE, 2013, pp. 65–74. doi:10.1109/QSIC.2013.53.

[103] L. Bijlsma, N. Doorn, H. Passier, H. Pootjes, S. Stuurman, How do students test software units?, in: IEEE/ACM International Conference on Software Engineering: Software Engineering Education and Training ICSE (SEET), IEEE, 2021, pp. 189–198. doi:10.1109/ICSE-SEET52601.2021.00029.

[104] R. Pham, J. Mörschbach, K. Schneider, Communicating software testing culture through visualizing testing activity, in: I. Hammouda, A. Sillitti (Eds.), Proceedings of the 7th International Workshop on Social Software Engineering, SSE 2015, Bergamo, Italy, September 1, 2015, ACM, 2015, pp. 1–8. doi:10.1145/2804381.2804382.

[105] M. P. Prado, A. M. R. Vincenzi, Towards cognitive support for unit testing: A qualitative study with practitioners, J. Syst. Softw. 141 (2018) 66–84. doi:10.1016/j.jss.2018.03.052.

[106] M. Betka, S. Wagner, Extreme mutation testing in practice: An industrial case study, in: IEEE/ACM International Conference on Automation of Software Test (AST@ICSE), IEEE, 2021, pp. 113–116. doi:10.1109/AST52587.2021.00021.

[107] G. Lou, Y. Deng, X. Zheng, M. Zhang, T. Zhang, Testing of autonomous driving systems: Where are we and where should we go?, in: A. Roychoudhury, C. Cadar, M. Kim (Eds.), ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE), ACM, 2022, pp. 31–43. doi:10.1145/3540250.3549111.

[108] B. S. Aragão, R. M. C. Andrade, I. S. Santos, R. N. S. Castro, V. Lelli, T. G. R. Darin, Testdcat: Catalog of test debt subtypes and management activities, in: C. Gaston, N. Kosmatov, P. L. Gall (Eds.), IFIP WG International Conference on Testing Software and Systems (ICTSS), Vol. 11812 of Lecture Notes in Computer Science, Springer, 2019, pp. 279–295. doi:10.1007/978-3-030-31280-0_18.

[109] M. F. Aniche, M. A. Gerosa, Does test-driven development improve class design? A qualitative study on developers' perceptions, J. Braz. Comput. Soc. 21 (1) (2015) 15:1–15:11. doi:10.1186/s13173-015-0034-z.

[110] S. Hotomski, M. Glinz, A qualitative study on using guidegen to keep requirements and acceptance tests aligned, in: G. Ruhe, W. Maalej, D. Amyot (Eds.), 26th IEEE International Requirements Engineering Conference, RE 2018, Banff, AB, Canada, August 20-24, 2018, IEEE Computer Society, 2018, pp. 29–39. doi:10.1109/RE.2018.00-54.

[111] D. Spadini, M. F. Aniche, M. D. Storey, M. Bruntink, A. Bacchelli, When testing meets code review: why and how developers review tests, in: M. Chaudron, I. Crnkovic, M. Chechik, M. Harman (Eds.), Proceedings of the 40th International Conference on Software Engineering, ICSE 2018, Gothenburg, Sweden, May 27 - June 03, 2018, ACM, 2018, pp. 677–687. doi:10.1145/3180155.3180192.

[112] N. M. Minhas, J. Börstler, K. Petersen, Checklists to support decision-making in regression testing, J. Syst. Softw. 202 (2023) 111697. doi:10.1016/j.jss.2023.111697.

[113] C. E. Brandt, A. Zaidman, How does this new developer test fit in? A visualization to understand amplified test cases, in: Working Conference on Software Visualization, VISSOFT 2022, Limassol, Cyprus, October 3-4, 2022, IEEE, 2022, pp. 17–28. doi:10.1109/VISSOFT55257.2022.00011.

[114] M. Irshad, R. Britto, K. Petersen, Adapting behavior driven development (BDD) for large-scale software systems, J. Syst. Softw. 177 (2021) 110944. doi:10.1016/j.jss.2021.110944.

[115] J. Fischbach, H. Femmer, D. Méndez, D. Fucci, A. Vogelsang, What makes agile test artifacts useful?: An activity-based quality model from a practitioners' perspective, in: M. T. Baldassarre, F. Lanubile, M. Kalinowski, F. Sarro (Eds.), ESEM '20: ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, Bari, Italy, October 5-7, 2020, ACM, 2020, pp. 41:1–41:10. doi:10.1145/3382494.3421462.

[116] D. Votipka, R. Stevens, E. M. Redmiles, J. Hu, M. L. Mazurek, Hackers vs. testers: A comparison of software vulnerability discovery processes, in: IEEE Symposium on Security and Privacy (SP), IEEE Computer Society, 2018, pp. 374–391. doi:10.1109/SP.2018.00003.

[117] A. Jamil, L. B. Othmane, A. Valani, M. AbdelKhalek, A. Tek, The current practices of changing secure software: An empirical study, in: C. Hung, T. Cerný, D. Shin, A. Bechini (Eds.), ACM/SIGAPP Symposium on Applied Computing (SAC), ACM, 2020, pp. 1566–1575. doi:10.1145/3341105.3373922.

[118] T. P. R. y Hernández, N. Marsden, Understanding software testers in the automotive industry - A mixed-method case study, in: A. Holzinger, T. Libourel, L. A. Maciaszek, S. J. Mellor (Eds.), International Conference on Software Engineering and Applications (ICSOFT-EA), SciTePress, 2014, pp. 305–314. doi:10.5220/0004992503050314.

[119] A. Afzal, C. L. Goues, M. Hilton, C. S. Timperley, A study on challenges of testing robotic systems, in: IEEE International Conference on Software Testing, Verification and Validation (ICST), IEEE, 2020, pp. 96–107. doi:10.1109/ICST46399.2020.00020.

[120] H. Shah, S. Sinha, M. J. Harrold, Outsourced, offshored software-testing practice: Vendor-side experiences, in: IEEE International Conference on Global Software Engineering (ICGSE), IEEE Computer Society, 2011, pp. 131–140. doi:10.1109/ICGSE.2011.32.

[121] A. Deak, T. Stålhane, G. Sindre, Challenges and strategies for motivating software testing personnel, Inf. Softw. Technol. 73 (2016) 1–15.

doi:10.1016/j.infsof.2016.01.002.
[122] H. Shah, M. J. Harrold, S. Sinha, Global software testing under deadline pressure: Vendor-side experiences, Inf. Softw. Technol. 56 (1) (2014) 6–19. doi:10.1016/j.infsof.2013.04.005.
[123] H. Shah, M. J. Harrold, Culture and testing: What is the relationship?, in: 8th IEEE International Conference on Global Software Engineering, ICGSE 2013, Bari, Italy, August 26-29, 2013, IEEE Computer Society, 2013, pp. 51–60. doi:10.1109/ICGSE.2013.15.
[124] D. B. Martin, J. Rooksby, M. Rouncefield, I. Sommerville, 'good' organisational reasons for 'bad' software testing: An ethnographic study of testing in a small software company, in: 29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007, IEEE Computer Society, 2007, pp. 602–611. doi:10.1109/ICSE.2007.1.
[125] S. Roy, F. Hermans, A. van Deursen, Spreadsheet testing in practice, in: M. Pinzger, G. Bavota, A. Marcus (Eds.), IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER), IEEE Computer Society, 2017, pp. 338–348. doi:10.1109/SANER.2017.7884634.
[126] S. Habchi, G. Haben, M. Papadakis, M. Cordy, Y. L. Traon, A qualitative study on the sources, impacts, and mitigation strategies of flaky tests, in: IEEE International Conference on Software Testing, Verification and Validation (ICST), IEEE, 2022, pp. 244–255. doi:10.1109/ICST53961.2022.00034.
[127] I. Salman, P. Rodríguez, B. Turhan, A. Tosun, A. Gureller, What leads to a confirmatory or disconfirmatory behavior of software testers?, IEEE Trans. Software Eng. 48 (4) (2022) 1351–1368. doi:10.1109/TSE.2020.3019892.
[128] R. Pham, S. Kiesling, L. Singer, K. Schneider, Onboarding inexperienced developers: Struggles and perceptions regarding automated testing, Softw. Qual. J. 25 (4) (2017) 1239–1268. doi:10.1007/s11219-016-9333-7.
[129] J. Kasurinen, O. Taipale, K. Smolander, Software test automation in practice: Empirical observations, Adv. Softw. Eng. 2010 (2010) 620836:1–620836:18. doi:10.1155/2010/620836.
[130] D. Parsons, T. Susnjak, M. Lange, Influences on regression testing strategies in agile software development environments, Softw. Qual. J. 22 (4) (2014) 717–739. doi:10.1007/s11219-013-9225-z.
[131] L. M. Riungu, O. Taipale, K. Smolander, Software testing as an online service: Observations from practice, in: IEEE International Conference on Software Testing, Verification and Validation (ICST), IEEE Computer Society, 2010, pp. 418–423. doi:10.1109/ICSTW.2010.62.
[132] O. Taipale, K. Smolander, Improving software testing by observing practice, in: G. H. Travassos, J. C. Maldonado, C. Wohlin (Eds.), International Symposium on Empirical Software Engineering (ISESE), ACM, 2006, pp. 262–271. doi:10.1145/1159733.1159773.
[133] J. Kasurinen, O. Taipale, K. Smolander, How test organizations adopt new testing practices and methods?, in: IEEE International Conference on Software Testing, Verification and Validation (ICST), IEEE Computer Society, 2011, pp. 553–558. doi:10.1109/ICSTW.2011.63.
[134] J. Kasurinen, O. Taipale, K. Smolander, Test case selection and prioritization: risk-based or design-based?, in: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, 2010, pp. 1–10.
[135] B. Gopal, R. Bockmon, S. Cooper, The impact of pogil-like learning on student understanding of software testing and devops: A qualitative study, in: S. Holland, M. Petre, L. Church, M. Marasoiu (Eds.), Proceedings of the 33rd Annual Workshop of the Psychology of Programming Interest Group, PPIG 2022, The Open University, Milton Keynes, UK & Online, September 5-9, 2022, Psychology of Programming Interest Group, 2022, pp. 241–250.
[136] B. Gopal, S. Cooper, J. Olmanson, R. Bockmon, Student difficulties in unit testing, integration testing and continuous integration: An exploratory pilot qualitative study., in: PPIG, 2021.
[137] M. F. Aniche, T. M. Ferreira, M. A. Gerosa, What concerns beginner test-driven development practitioners: a qualitative analysis of opinions in an agile conference, in: 2nd Brazilian Workshop on Agile Methods, Vol. 19, 2011, p. 22.
[138] K. Karhu, T. Repo, O. Taipale, K. Smolander, Empirical observations on software testing automation, in: Second International Conference on Software Testing Verification and Validation, ICST 2009, Denver, Colorado, USA, April 1-4, 2009, IEEE Computer Society, 2009, pp. 201–209. doi:10.1109/ICST.2009.16.
[139] F. P. Seth, O. Taipale, K. Smolander, Organizational and customer related challenges of software testing: An empirical study in 11 software companies, in: M. Bajec, M. Collard, R. Deneckère (Eds.), IEEE 8th International Conference on Research Challenges in Information Science, RCIS 2014, Marrakech, Morocco, May 28-30, 2014, IEEE, 2014, pp. 1–12. doi:10.1109/RCIS.2014.6861031.
[140] E. Alégroth, R. Feldt, On the long-term use of visual gui testing in industrial practice: a case study, Empir. Softw. Eng. 22 (6) (2017) 2937–2971. doi:10.1007/s10664-016-9497-6.
[141] J. Kasurinen, K. Smolander, What do game developers test in their products?, in: M. Morisio, T. Dybå, M. Torchiano (Eds.), 2014 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14, Torino, Italy, September 18-19, 2014, ACM, 2014, pp. 1:1–1:10. doi:10.1145/2652524.2652525.
[142] R. Pham, L. Singer, O. Liskin, F. M. F. Filho, K. Schneider, Creating a shared understanding of testing culture on a social coding site, in: D. Notkin, B. H. C. Cheng, K. Pohl (Eds.), 35th International Conference on Software Engineering, ICSE '13, San Francisco, CA, USA, May 18-26, 2013, IEEE Computer Society, 2013, pp. 112–121. doi:10.1109/ICSE.2013.6606557.
[143] J. Itkonen, M. Mäntylä, C. Lassenius, The role of the tester's knowledge in exploratory software testing, IEEE Trans. Software Eng. 39 (5) (2013) 707–724. doi:10.1109/TSE.2012.55.
[144] V. Kettunen, J. Kasurinen, O. Taipale, K. Smolander, A study on agility and testing processes in software organizations, in: P. Tonella, A. Orso (Eds.), Proceedings of the Nineteenth International Symposium on Software Testing and Analysis, ISSTA 2010, Trento, Italy, July 12-16, 2010, ACM, 2010, pp. 231–240. doi:10.1145/1831708.1831737.
[145] J. Kasurinen, O. Taipale, K. Smolander, Analysis of problems in testing practices, in: S. Sulaiman, N. M. M. Noor (Eds.), 16th Asia-Pacific Software Engineering Conference, APSEC 2009, 1-3 December 2009, Batu Ferringhi, Penang, Malaysia, IEEE Computer Society, 2009, pp. 309–315. doi:10.1109/APSEC.2009.17.
[146] C. Polepalle, R. S. Kondoju, D. Badampudi, Evidence and perceptions on GUI test automation - an exploratory study, in: S. Tiwari, S. Chaudhary, C. K. Roy, M. D'Souza, R. Sharma, L. Kumar (Eds.), ISEC 2022: 15th Innovations in Software Engineering Conference, Gandhinagar, India, February 24 - 26, 2022, ACM, 2022, pp. 14:1–14:10. doi:10.1145/3511430.3511442.
[147] S. Tyagi, R. Sibal, B. Suri, Adopting test automation on agile development projects: A grounded theory study of indian software organizations, in: H. Baumeister, H. Lichter, M. Riebisch (Eds.), Agile Processes in Software Engineering and Extreme Programming - 18th

International Conference, XP 2017, Cologne, Germany, May 22-26, 2017, Proceedings, Vol. 283 of Lecture Notes in Business Information Processing, 2017, pp. 184–198. doi:10.1007/978-3-319-57633-6_12.

[148] R. Florea, V. Stray, A qualitative study of the background, skill acquisition, and learning preferences of software testers, in: J. Li, L. Jaccheri, T. Dingsøyr, R. Chitchyan (Eds.), Evaluation and Assessment in Software Engineering (EASE), ACM, 2020, pp. 299–305. doi:10.1145/3383219.3383252.

[149] S. Hotomski, E. B. Charrada, M. Glinz, An exploratory study on handling requirements and acceptance test documentation in industry, in: IEEE International Requirements Engineering Conference (RE), IEEE Computer Society, 2016, pp. 116–125. doi:10.1109/RE.2016.37.

[150] F. F. Ismail, R. Razali, A qualitative empirical investigation of contributing success factors for software testing outsourcing projects, Jurnal Teknologi 77 (9).

[151] A. Vianna, W. Ferreira, K. Gama, An exploratory study of how specialists deal with testing in data stream processing applications, in: ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), IEEE, 2019, pp. 1–6. doi:10.1109/ESEM.2019.8870186.

[152] T. Illes-Seifert, B. Paech, On the role of communication, documentation and experience during system testing-an interview study., in: PRIMIUM, 2008.

[153] Å. Dahlstedt, Challenges in system testing—an interview study, in: Advances in Information Systems Development: Bridging the Gap between Academia and Industry, Springer, 2006, pp. 1043–1052.

[154] M. Dolezel, Defining testops: Collaborative behaviors and technology-driven workflows seen as enablers of effective software testing in devops, in: M. Paasivaara, P. Kruchten (Eds.), Agile Processes in Software Engineering and Extreme Programming - Workshops - XP 2020 Workshops, Copenhagen, Denmark, June 8-12, 2020, Revised Selected Papers, Vol. 396 of Lecture Notes in Business Information Processing, Springer, 2020, pp. 253–261. doi:10.1007/978-3-030-58858-8_26.

[155] A. Deak, A comparative study of testers' motivation in traditional and agile software development, in: A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch, M. Raatikainen (Eds.), Product-Focused Software Process Improvement - 15th International Conference, PROFES 2014, Helsinki, Finland, December 10-12, 2014. Proceedings, Vol. 8892 of Lecture Notes in Computer Science, Springer, 2014, pp. 1–16. doi:10.1007/978-3-319-13835-0_1.

[156] V. Garousi, M. Felderer, M. Kuhrmann, K. Herkiloglu, S. Eldh, Exploring the industry's challenges in software testing: An empirical study, J. Softw. Evol. Process. 32 (8). doi:10.1002/smr.2251.

[157] Y. Lizama, D. Varona, P. Waychal, L. F. Capretz, The unpopularity of the software tester role among software practitioners: A case study, CoRR abs/2007.08375. arXiv:2007.08375.

[158] P. S. Kochhar, X. Xia, D. Lo, Practitioners' views on good software testing practices, in: H. Sharp, M. Whalen (Eds.), Proceedings of the 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE (SEIP) 2019, Montreal, QC, Canada, May 25-31, 2019, IEEE / ACM, 2019, pp. 61–70. doi:10.1109/ICSE-SEIP.2019.00015.

[159] R. E. de Souza Santos, C. V. C. de Magalhães, J. da Silva Correia-Neto, F. Q. B. da Silva, L. F. Capretz, R. Souza, Would you like to motivate software testers? ask them how, in: A. Bener, B. Turhan, S. Biffl (Eds.), 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2017, Toronto, ON, Canada, November 9-10, 2017, IEEE Computer Society, 2017, pp. 95–104. doi:10.1109/ESEM.2017.16.

[160] G. Scanniello, S. Romano, D. Fucci, B. Turhan, N. Juristo, Students' and professionals' perceptions of test-driven development: a focus group study, in: S. Ossowski (Ed.), Proceedings of the 31st Annual ACM Symposium on Applied Computing, Pisa, Italy, April 4-8, 2016, ACM, 2016, pp. 1422–1427. doi:10.1145/2851613.2851778.

[161] A. Deak, T. Stålhane, Organization of testing activities in norwegian software companies, in: Sixth IEEE International Conference on Software Testing, Verification and Validation, ICST 2013 Workshops Proceedings, Luxembourg, Luxembourg, March 18-22, 2013, IEEE Computer Society, 2013, pp. 102–107. doi:10.1109/ICSTW.2013.18.

[162] P. Waychal, L. F. Capretz, J. Jia, D. Varona, Y. Lizama, Practitioners' testimonials about software testing, in: 28th IEEE International Conference on Software Analysis, Evolution and Reengineering, SANER 2021, Honolulu, HI, USA, March 9-12, 2021, IEEE, 2021, pp. 582–589. doi:10.1109/SANER50967.2021.00070.

[163] S. O. Barraood, H. Mohd, F. Baharom, Test case quality factors: Content analysis of software testing websites, Webology 18 (SI01) (2021) 75–87. doi:10.14704/web/v18si01/web18007.

[164] I. Evans, C. Porter, M. Micallef, Scared, frustrated and quietly proud: Testers' lived experience of tools and automation, in: P. Marti, O. Parlangeli, A. Recupero (Eds.), ECCE 2021: European Conference on Cognitive Ergonomics 2021, Virtual Event / Siena, Italy, 26-29 April, 2021, ACM, 2021, pp. 16:1–16:7. doi:10.1145/3452853.3452872.

[165] A. Bertolino, Software testing research: Achievements, challenges, dreams, in: Future of Software Engineering (FOSE'07), IEEE, 2007, pp. 85–103.

[166] S. P. Ng, T. Murnane, K. Reed, D. Grant, T. Y. Chen, A preliminary survey on software testing practices in australia, in: 2004 Australian Software Engineering Conference. Proceedings., IEEE, 2004, pp. 116–125.

[167] K. M. Eisenhardt, Building theories from case study research, Academy of management review 14 (4) (1989) 532–550.

[168] K. Beck, Test-driven development: by example, Addison-Wesley Professional, 2003.

[169] G. Tassey, The economic impacts of inadequate infrastructure for software testing. national institute of standards and technology, RTI Project 7007 (11) (2002) 1–309.

[170] V. Braun, V. Clarke, Using thematic analysis in psychology, Qualitative research in psychology 3 (2) (2006) 77–101.

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

[172] H. Zhang, X. Huang, X. Zhou, H. Huang, M. A. Babar, Ethnographic research in software engineering: a critical review and checklist, in: Proceedings of the 2019 27th ACM joint meeting on European software engineering conference and symposium on the foundations of software engineering, 2019, pp. 659–670.

[173] H. Shah, N. J. Nersessian, M. J. Harrold, W. Newstetter, Studying the influence of culture in global software engineering: thinking in terms of cultural models, in: Proceedings of the 4th international conference on Intercultural Collaboration, 2012, pp. 77–86.

[174] P. Runeson, A survey of unit testing practices, IEEE software 23 (4) (2006) 22–29.
[175] J. A. Whittaker, What is software testing? and why is it so hard?, IEEE software 17 (1) (2000) 70–79.
[176] S. Berner, R. Weber, R. K. Keller, Observations and lessons learned from automated testing, in: Proceedings of the 27th international conference on Software engineering, 2005, pp. 571–579.
[177] European Commission (EC), The new SME Definition: User guide and model declaration, Enterprise and Industry Publications, European Commission, 2003.
[178] H. K. Klein, M. D. Myers, A set of principles for conducting and evaluating interpretive field studies in information systems, MIS quarterly (1999) 67–93.
[179] G. Paré, J. J. Elam, Using case study research to build theories of it implementation, in: Information Systems and Qualitative Research: Proceedings of the IFIP TC8 WG 8.2 International Conference on Information Systems and Qualitative Research, 31st May–3rd June 1997, Philadelphia, Pennsylvania, USA, Springer, 1997, pp. 542–568.
[180] I. .-. 2004, Information technology—process assessment—part 1: Concepts and vocabulary (2004).
[181] G. Meszaros, xUnit test patterns: Refactoring test code, Pearson Education, 2007.
[182] D. Athanasiou, A. Nugroho, J. Visser, A. Zaidman, Test code quality and its relation to issue handling performance, IEEE Transactions on Software Engineering 40 (11) (2014) 1100–1125.
[183] Y. Jia, M. Harman, An analysis and survey of the development of mutation testing, IEEE transactions on software engineering 37 (5) (2010) 649–678.
[184] G. J. Myers, C. Sandler, T. Badgett, The art of software testing, John Wiley & Sons, 2011.
[185] L. F. Capretz, F. Ahmed, Making sense of software development and personality types, IT professional 12 (1) (2010) 6–13.
[186] E. J. Weyuker, T. J. Ostrand, J. Brophy, B. Prasad, Clearing a career path for software testers, IEEE software 17 (2) (2000) 76–82.
[187] 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 133 (2021) 106514.
[188] A. Rainer, C. Wohlin, Case study identification: A trivial indicator outperforms human classifiers, Information and Software Technology 161 (2023) 107252.
[189] I. Benbasat, D. K. Goldstein, M. Mead, The case research strategy in studies of information systems, MIS quarterly (1987) 369–386.
[190] H. Sharp, Y. Dittrich, C. R. De Souza, The role of ethnographic studies in empirical software engineering, IEEE Transactions on Software Engineering 42 (8) (2016) 786–804.
[191] S. Herbert, For ethnography, Progress in human geography 24 (4) (2000) 550–568.
[192] C. Geertz, Thick description: Toward an interpretive theory of culture, in: The cultural geography reader, Routledge, 2008, pp. 41–51.
[193] B. G. Glaser, Theoretical sensitivity: Advances in the methodology of grounded theory, Sociology Pr.
[194] J. C. Van Niekerk, J. Roode, Glaserian and straussian grounded theory: similar or completely different?, in: Proceedings of the 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, 2009, pp. 96–103.
[195] B. G. Glaser, J. Holton, et al., Remodeling grounded theory 5 (2).
[196] R. Hoda, Socio-technical grounded theory for software engineering, IEEE Transactions on Software Engineering 48 (10) (2021) 3808–3832.
[197] C. Wohlin, P. Runeson, Guiding the selection of research methodology in industry–academia collaboration in software engineering, Information and software technology 140 (2021) 106678.
[198] T. Potuzak, R. Lipka, Current trends in automated test case generation, in: 2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS), IEEE, 2023, pp. 627–636.
[199] B. Rous, Major update to acm's computing classification system, Communications of the ACM 55 (11) (2012) 12–12.
[200] D. Mendez, D. Graziotin, S. Wagner, H. Seibold, Open science in software engineering, Contemporary empirical methods in software engineering (2020) 477–501.
[201] M. Fu, C. Tantithamthavorn, T. Le, Y. Kume, V. Nguyen, D. Phung, J. Grundy, Aibughunter: A practical tool for predicting, classifying and repairing software vulnerabilities, Empirical Software Engineering 29 (1) (2024) 4.
[202] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, Q. Wang, Software testing with large language models: Survey, landscape, and vision, IEEE Transactions on Software Engineering.
[203] L. McLeod, S. G. MacDonell, B. Doolin, Qualitative research on software development: a longitudinal case study methodology, Empirical software engineering 16 (2011) 430–459.
[204] Google Scholar, Google scholar metrics, accessed: July 25, 2024 (2024).