Software Quality Journal (2025) 33:9
https://doi.org/10.1007/s11219-024-09698-w

RESEARCH

Quality attributes of test cases and test suites – importance & challenges from practitioners' perspectives

Huynh Khanh Vi Tran[1] · Nauman bin Ali[1] · Michael Unterkalmsteiner[1] · Jürgen Börstler[1] · Panagiota Chatzipetrou[2]

Accepted: 17 October 2024

Abstract
The quality of the test suites and the constituent test cases significantly impacts confidence in software testing. While research has identified several quality attributes of test cases and test suites, there is a need for a better understanding of their relative importance in practice. We investigate practitioners' perceptions regarding the relative importance of quality attributes of test cases and test suites and the challenges that they face in ensuring the perceived important quality attributes. To capture the practitioners' perceptions, we conducted an industrial survey using a questionnaire based on the quality attributes identified in an extensive literature review. We used a sampling strategy that leverages LinkedIn to draw a large and heterogeneous sample of professionals with experience in software testing. We collected 354 responses from practitioners with a wide range of experience (from less than one year to 42 years of experience). We found that the majority of practitioners rated Fault Detection, Usability, Maintainability, Reliability, and Coverage to be the most important quality attributes. Resource Efficiency, Reusability, and Simplicity received the most divergent opinions, which, according to our analysis, depend on the software-testing contexts. Also, we identified common challenges that apply to the important attributes, namely inadequate definition, lack of useful metrics, lack of an established review process, and lack of external support. The findings point out where practitioners actually need further support with respect to achieving high-quality test cases and test suites under different software testing contexts. Hence, the findings can serve as a guideline for academic researchers when looking for research directions on the topic. Furthermore, the findings can be used to encourage companies to provide more support to practitioners to achieve high-quality test cases and test suites.

Keywords Software testing · Test case quality · Test suite quality · Quality assurance

1 Introduction

There have been several studies over the years emphasizing the value of high-quality test cases and test suites (Athanasiou et al., 2014; Grano et al., 2018; Garousi and Küçük, 2018) as well as the effects of design flaws in them (Spadini et al., 2018; Kovács and Szabados, 2014; Garousi and Küçük, 2018; Garousi et al., 2019). Moreover, modern software development

Extended author information available on the last page of the article

Published online: 15 January 2025

with continuous software engineering (Fitzgerald and Stol, 2017) and DevOps (Jabbari et al., 2018) relies extensively on automated and continuous testing. For these reasons, it is important to ensure the quality of test cases and test suites.

To characterize and assess the quality of test cases and test suites, attributes similar to software product quality attributes have been proposed. The most recent systematic literature review on the quality attributes of test cases and test suites (Tran et al., 2021) was conducted by the authors of this study. This review provided a comprehensive overview of 30 quality attributes for test cases and test suites, and can serve as a guideline for both practitioners and researchers when searching for literature on the topic. For high-quality test cases and test suites, one can argue that all 30 quality attributes should be relevant. However, focusing on all quality attributes together can be very challenging, especially when certain trade-offs are expected between the quality attributes, i.e., all the attributes cannot be improved simultaneously. The existence of trade-offs is well motivated by several studies that investigated software quality attributes (Sas and Avgeriou, 2020; Wahler et al., 2017; Feitosa et al., 2015; Ali et al., 2012).

Instead of focusing on all quality attributes or some random attributes, we argue that it is better to understand which quality attributes practitioners perceive as important so that we, as researchers, can focus on providing further support in assessing, maintaining, and improving these important attributes. To provide such assistance to practitioners, there is a need to understand the software-testing contexts in which certain quality attributes are perceived as more or less important, as well as any challenges that practitioners face in defining, measuring, and maintaining these quality attributes. Note that the referred contexts can be characterized by different context dimensions such as testing level, testing practice, and characteristics of the system under test (Tran et al., 2021).

The quality of test cases and test suites has received great attention from the research community for several decades. There also exist studies focusing on challenges in software testing in general (Bertolino, 2007; Garousi et al., 2017, 2020; Santos et al., 2019; Fulcini et al., 2023; Waseem et al., 2021; Ali et al., 2012), or challenges related to the quality of test cases and test suites (Juhnke et al., 2021; Kochhar et al., 2019). However, to the best of our knowledge, there are no studies investigating practitioners' perceptions of the importance and challenges (with respect to defining, measuring, and maintaining) of a wide range of quality attributes of test cases and test suites in multiple contexts. We conducted an industrial survey to address this research gap. Our survey recruited practitioners from 20 LinkedIn groups to draw a large-scale sample and increase subject heterogeneity. As a result, we received responses from 354 practitioners who got involved in various testing activities with a wide range of working experience.

The contribution of this survey study is three-fold: (1) the identification of generally important (i.e., independent of context) quality attributes of test cases and test suites in practice, namely Fault Detection, Maintainability, Reliability, Usability, and Coverage; (2) the confirmation of the influence of software-testing context dimensions (Automation activity, Testing activity, Testing level, Testing practice, Type of development process of System under test (SUT), Application domain of SUT, and Type of SUT) on the perceived importance of certain attributes, namely Resource Efficiency, Reusability, Simplicity, and Usability; and (3) the identification of common challenges with respect to defining, measuring, and maintaining the important attributes: inadequate definition, lack of useful metrics, lack of an established review process, and lack of external support.

Ultimately, our study provides valuable insights into how practitioners perceive the importance of quality attributes of both test cases and test suites. Our findings highlight important quality attributes in practice, the weight of certain context dimensions on the importance

of several quality attributes, and the challenges that apply to the important attributes. Altogether, our study points out where practitioners actually need further support with respect to achieving high-quality test cases and test suites. Hence, our findings can serve as a guideline for academic researchers when looking for research directions on the topic, knowing that their research can meet the actual needs in practice. Additionally, the findings can be used to encourage companies to provide more support to practitioners to achieve high-quality test cases and test suites. Ultimately, our findings provide extra support for the knowledge transfer between academia and industry.

The rest of the paper is structured as follows: Section 2 presents related work on the quality of test cases and test suites, general challenges in software testing, and challenges related to the quality of test cases and test suites. Section 3 describes how we conducted the survey. Section 4 presents the survey's results and our analysis for each research question. In Section 5, we discuss selected findings related to the importance of quality attributes, their correlation to the software-testing contexts, and the challenges that apply to the important quality attributes. Section 6 discusses the limitations and the primary threats to validity. Finally, Section 7 concludes our paper.

2 Related work

In this section, we discuss related work from three angles. First, we look at past research on the quality of test cases and test suites. Then, we discuss related works that address general challenges in software testing (Bertolino, 2007; Garousi et al., 2017, 2020; Santos et al., 2019; Fulcini et al., 2023; Waseem et al., 2021; Ali et al., 2012). Finally, we discuss studies that focus on challenges related to the quality of test cases and test suites (Kochhar et al., 2019; Juhnke et al., 2021).

2.1 Quality of test cases and test suites

The quality of test cases and test suites has been investigated for several decades. One of the earliest related works was conducted by Goodenough and Gerhart (1975), who received attention from the research community when studying test adequacy and proposing a definition for reliable test data. Many years later, Zhu et al. (1997) conducted a literature review on test adequacy criteria. Their reviews encouraged fellow researchers to investigate the topic of structural test quality for specific application domains and programming paradigms (Kapfhammer and Soffa, 2003; Lemos et al., 2007; Pei et al., 2017). Besides, some researchers turned their focus to adapting software quality models to define the quality of test cases and test suites (Neukirchen et al., 2008; Athanasiou et al., 2014). In recent years, researchers started looking into how inputs from practitioners could help to construct useful quality models for test cases and test suites (Bowes et al., 2017; Grano et al., 2020; Tran et al., 2019).

The most recent literature review provides a consolidated overview of state of the art in test artifact quality in software testing (Tran et al., 2021), conducted by the authors of this study. The main contribution is a quality model of test cases and test suites containing 30 quality attributes, eleven of which have measurement information. The quality model also includes quality attributes adapted from ISO/IEC 25010:2011 to provide a more extensive overview of the quality of test cases and test suites. In addition, we identified in the literature review eleven context dimensions that have been used to describe the software-testing contexts in

which the quality has been studied. The quality model could assist in developing guidelines or templates for designing test cases and test suites and assessment tools for existing test cases and test suites.

The quality of test cases and test suites have also been discussed in studies on test smells. The concept of test smells was coined by Van Deursen et al. (2001). In the context of unit testing for eXtreme Programming, Van Deursen et al. define test smells as derived from code smells and as indicators of troubles in test code. Later on, Meszaros (2007), while referring to the work of Van Deursen et al., introduced definitions of more test smells for the unit testing framework xUnit. The latest secondary study on definitions of test smells was conducted by Garousi and Küçük (2018). The study provided a list of 139 test smells, as well as a summary of approaches and tools to deal with the test smells. Further in this direction, Aljedaani et al. (2021) provided an overview of the characteristics of over 22 test smell detection tools developed in the research community.

2.2 General challenges in software testing

Bertolino (2007) provided one of the first broad roadmaps of achievements and challenges to address in the field of software testing research. Particularly, the author highlighted certain important milestones, including matured research in (1) techniques and tools to support test design; (2) test criteria and their effectiveness; (3) different testing techniques. The author emphasized the importance of a universal test theory, i.e., a coherent and rigorous framework that practitioners can refer to for selecting adequate testing approaches. Besides, model-based testing, 100% automatic testing, and cost-effective test engineering were also mentioned as what the future software-testing research should focus on. For each of these foci, several challenges were highlighted. For example, *test effectiveness* (How effective is a test selection criterion for finding faults?) was listed as one of the main challenges for achieving a universal test theory. For achieving 100% automatic testing, the research in test input generation led to advancements in theory but has limited industrial impact. Likewise, understanding testing costs was reported as a challenge for "efficacy-maximized test engineering".

Ten years later, Garousi et al. (2017, 2020) presented 105 practitioners' opinions on challenges in different testing activities, including test-case design (criteria-based or based on human expertise), test scripting, test execution, test evaluation, test-result reporting, test management, test automation/tools, and others. The authors asked practitioners to rank the level of challenges (from no challenge at all to lots of challenges) in each testing activity, and then to propose concrete topics they wanted researchers to focus on. According to their findings, test management and automation/tools and "other" test activities were considered the most challenging testing activities. Their main conclusion was that industry and academic focus areas were not aligned. While researchers tended to focus on theoretical challenges such as search-based test-case design, practitioners wanted to improve the effectiveness and efficiency of software testing. Going in the same direction as Garousi et al., Santos et al. (2019) conducted a mapping study, a quantitative study, and a focus group to analyze the misalignment regarding important aspects for improvement between the software-testing research community and practitioners. Their main conclusion was that even though practitioners and researchers were both interested in test automation, practitioners would like to focus on identifying, understanding, and modifying existing testing tools and strategies while researchers paid more attention to developing new ones.

Some studies focus on testing challenges for specific types of system under tests (SUT), application domains, or testing approaches. Fulcini et al. (2023) conducted a multivocal lit-

erature review that assessed techniques, tools, and challenges related to gamified software testing. According to the authors, gamification was a promising testing approach but was not yet well-established, and several challenges needed to be addressed. The authors categorized the challenges into three groups: design improvements (related to adding new gamified mechanics or improving existing ones regarding several aspects such as ethical concerns or tool simplification), implementation improvements (related to the technical implementation of the gamified tool and frameworks such as tool reusability and scalability), and evaluation (related to evaluating gamified mechanics such as the needs for automatic analysis or expert evaluation). The study of Waseem et al. (2021) was based on 106 survey responses and six interviews and focuses on microservice systems. Besides findings regarding designing and monitoring such systems, the authors reported that system complexity was one of the main reasons making the testing activities (creating and implementing manual tests, integration testing) challenging and there were no dedicated solutions. Different from these aforementioned studies (Fulcini et al., 2023; Waseem et al., 2021), which broadly looked into either general aspects of software testing or challenges related to specific software-testing contexts, we focused on the quality attributes of test cases and test suites and their importance in relation to different context dimensions (testing level, testing practice, type of SUT, etc.).

2.3 Challenges related to the quality of test cases and test suites

The third group of related work includes studies (Juhnke et al., 2021; Kochhar et al., 2019) that are closer to our work as they also discussed the quality of test artifacts. Juhnke et al. (2021) conducted 17 interviews and surveyed 36 practitioners to collect and analyze challenges regarding test case specifications in the automotive domain. They classified the challenges into nine categories: availability problems with input artifacts; content-related problems with input artifacts, and knowledge-related problems; lack of knowledge; the test case description; the test case specification content; processes; communication; quality assurance; and tools. Their study showed that the latter four challenges were assessed as more frequently occurring and more critical by practitioners.

Kochhar et al. (2019) invited 282 practitioners to validate 29 hypotheses regarding characteristics of good test cases and testing practices. The hypotheses covered six aspects: test case contents, size and complexity, coverage, maintainability, and bug detection. Each hypothesis was ranked on a Likert scale from "strongly agree" to "strongly disagree". The authors concluded that some hypotheses received quite controversial opinions, and hence, researchers were suggested to conduct deep empirical studies to analyze them. As for practitioners, their hypotheses described characteristics of good test cases, which the authors claimed could be used as guidelines for novices to design high-quality test cases.

2.4 Research gap

We found that general challenges in software testing have been reported in the literature (Section 2.2). Another aspect related to the quality of test cases and test suites is test smells, which have been intensively investigated in recent years (Section 2.1). We identified two studies from Juhnke et al. (2021) and Kochhar et al. (2019) that investigated challenges related to the quality of test cases and test suites from the practitioners' perspective (Section 2.3).

While Juhnke et al. (2021) proposed nine categories of challenges related to test case specifications, their context is limited to the automotive domain. Our study is not restricted to a single software-testing context. We referred to our tertiary study (Tran et al., 2021) on the

quality of test cases and test suites together with various taxonomies (Forward and Lethbridge, 2008; Kotonya et al., 2003; Vijayasarathy and Butler, 2016; Souza et al., 2017; Ray et al., 2014; Ateşoğulları and Mishra, 2020) and the ISO/IEC/IEEE 29119-1 standard Software and systems engineering (2022) for different context dimensions in order to capture different software-testing contexts in practice. Kochhar et al. (2019) studied practitioners' opinions on different hypotheses on good test cases and testing practices. While their study covered some quality attributes, their attributes set was smaller than ours. Also, it was not clear in their paper how the attributes were chosen. In contrast, our quality attributes were extracted from our tertiary study (Tran et al., 2021) on the quality of test cases and test suites.

Therefore, even though our study also focused on understanding practitioners' perceptions of the quality of test cases and test suites, we systematically studied a wider range of software-testing contexts and a larger number of quality attributes of both test cases and test suites. On top of that, we have not found any study that has investigated the correlation between the software-testing context dimensions and the perceived importance of quality attributes. We filled in this research gap, by means of this survey study.

3 Research method

3.1 Research questions

The goal of our survey was to understand which quality attributes of test cases and test suites are considered important and the challenges in defining, measuring, and maintaining the important attributes from the practitioners' perspectives. To achieve the stated goal, we answered the following research questions.

- RQ1. What is the relative importance of quality attributes of test cases and test suites from practitioners' perspectives?
  We wanted to understand if some quality attributes might be perceived as more important than others by practitioners. To answer this research question, we asked respondents to rank the importance (as three levels: *Important* / *Optional* / *Not relevant at all*) of the quality attributes of either test cases or test suites. We distinguished between test cases and test suites in the ranking task because the quality attributes of test cases and test suites can share the same names but be defined differently. Furthermore, a test suite can contain multiple test cases, and hence, the perceived importance of test-suite quality attributes can be weighted differently than of test-case quality attributes.
- RQ2. To what extent does the software-testing context influence the perceived importance of the quality attributes of test cases and test suites?
  The software-testing context contains the following context dimensions: Testing activity, Testing level, Testing type, Testing practice, Automation activity, Application domain of system under test (SUT), Type of SUT, and Type of development process. We wanted to investigate if some quality attributes might be considered important in some contexts but not in different contexts. To answer this research question, we asked respondents to describe their software-testing context dimensions and then we used statistical tests to analyze the correlation between the context dimensions and the importance level of the quality attributes (RQ1's answer).
- RQ3. What challenges do practitioners face in ensuring the important quality attributes of test cases and test suites?

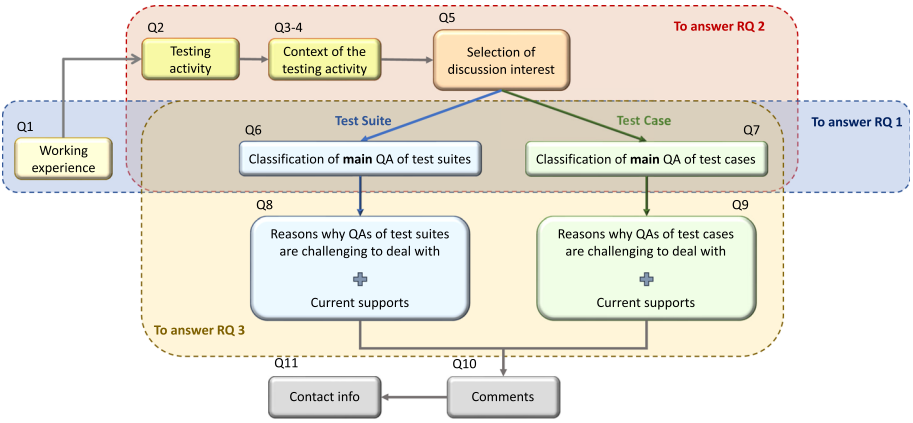

Fig. 1 Overview of the survey questionnaire

To help practitioners achieve high-quality test cases and test suites with respect to the important quality attributes, we wanted to understand the challenges that practitioners face in defining, measuring, and maintaining these attributes. To answer this research question, for each quality attribute, we asked respondents to select among three provided challenges (Juhnke et al., 2021): Challenge C_{def} - Inadequate definition, Challenge C_{metric} - Lack of useful metrics, and Challenge C_{review} - Lack of an established review process. We also offered a free-text field to collect other challenges.

3.2 Survey design

We followed Kitchenham and Pfleeger's guidelines Kitchenham and Pfleeger (2008) to design our survey, a structured cross-sectional web-based survey. This approach helps to reach practitioners from geographically diverse locations and supports automated data collection. The survey was administered via the web-based survey software, Qualtrics Survey[1]. An overview of the questionnaire is shown in Fig. 1.

The questionnaire contains three sections. The first section (Q1–4) collects demographic information about the practitioners (years of experience and software-testing contexts). The second section (Q5–7) collects the perceived importance levels (*Important* / *Optional* / *Not relevant at all*) of the quality attributes of test cases and test suites. It is worth noting that we referred to several taxonomies (Forward and Lethbridge, 2008; Kotonya et al., 2003; Vijayasarathy and Butler, 2016; Souza et al., 2017; Ray et al., 2014; Ateşoğulları and Mishra, 2020), the ISO/IEC/IEEE 29119-1 standard Software and systems engineering (2022) as well as our tertiary study (Tran et al., 2021) on the quality of test cases and test suites to provide the software-testing contexts and the list of quality attributes. More details are presented further below in this section.

The third section (Q8–9) collects two sets of information: (i) challenges that apply to the *important* quality attributes; (ii) information regarding any support the practitioners currently have to tackle the challenges, followed by the closing section (Q10–11). The final version of the questionnaire can be viewed https://doi.org/10.6084/m9.figshare.23309702.v1. In the

[1] https://www.qualtrics.com/uk/core-xm/survey-software/

first section (Q1–4) of the questionnaire, we collected information regarding participants' working experience (Q1) and their software testing-related contexts (Q2–4) in which they have considered the quality of test cases and test suites to be relevant and challenging to achieve, assess, maintain, or improve.

To characterize the software-testing contexts (Q2–4), we first presented a list of categories of common software-testing activities, which were extracted from the ISO/IEC/IEEE 29119-1 standard Software and systems engineering (2022). We then asked each participant to select a category that best describes his or her software-testing activities in which the participant has found it challenging to achieve, assess, maintain, or improve the quality of test cases or test suites (Q2). The subsequent questions (Q3–4) were to gain a better understanding of the context of the selected testing activities. For this purpose, we presented a list of context dimensions, each with common context values from which the participants could select. For example, under the *testing level* context dimension, a participant could select one of the following context values: unit testing, integration testing, system testing, system integration testing, and acceptance testing.

More specifically, we focused on the following context dimensions: testing level, testing type, testing practice, automation activity, and test suite's size, as well as the dimensions related to the systems under test (SUT): size, application domain, software type, and type of development process. These context dimensions were aggregated in our tertiary study (Tran et al., 2021) on the quality of test cases and test suites. Additionally, we relied on different taxonomies (Forward and Lethbridge, 2008; Kotonya et al., 2003; Vijayasarathy and Butler, 2016; Souza et al., 2017; Ray et al., 2014; Ateşoğulları and Mishra, 2020) and the ISO/IEC/IEEE 29119-1 standard Software and systems engineering (2022) to provide common context values under each context dimension in Q3–4. It is worth noting that there were also free text fields where the respondents could enter their own category of testing activities (Q2) and context value (Q3–4) if they found none of the provided categories and values were appropriate.

To help us answer RQ2 (investigating the extent software-testing contexts influence the perceived importance of the quality attributes of test cases and test suites), we stated explicitly in Q3–4 that the respondent should "consider the single context where you found the quality of test cases and test suites one of your main concerns." Due to this reason, we restricted each respondent to select only one context value under each context dimension. Without this design approach, we might have received an answer in which, for example, the respondent provided a mixed context of unit testing and system testing and marked two quality attributes, *Maintainability* and *Fault Detection*, as *Important*. In this example, it would be impossible to tell which quality attribute was associated with which testing level.

The second section (Q5–7) of the questionnaire collected the backbone data of the study. Q5 asked the respondents to select whether they wanted to share their opinions on the quality of either test cases or test suites. The reason for having Q5 is explained under RQ1 in Section 3.1. Depending on their answers to Q5, we asked the respondents to rank the importance of each main quality attribute of test cases (Q6) or test suites (Q7) as *Important* / *Optional* / *Not relevant at all*. Note that in Q6 and Q7, we reminded each respondent to think about the importance of each quality attribute in the software-testing context that he or she specified in Q2–4. To facilitate this ranking task, we provided the respondents with a test-case quality model (to answer Q6) and a test-suite quality model (to answer Q7). Each quality model consists of a list of main quality attributes and their sub-quality attributes, similar to the software quality models in ISO/IEC 25010:2011 ISO/IEC JTC 1/SC 7 (2011). Additionally, each quality attribute comes with a short description. The quality models and the descriptions of all provided quality attributes were aggregated from our tertiary study (Tran et al., 2021)

and the related work as shown in Table 1. Note that further quality attributes were presented in our tertiary study. However, some of them did not have descriptions as they were not described in the selected secondary studies or in the related work. Hence, these attributes were excluded from this survey.

Table 1 Descriptions of main quality attributes and their sub-quality attributes of test suites (TS) and test cases (TC)

Quality Attribute	For	Description in case of test suites	Description in case of test cases	Ref
Coverage *	TS	The degree of completeness and effectiveness with which a test suite ensures that SUT code is executed and the system requirements are verified.	N/A	S1
↳Code coverage completeness	TS	The degree to which a test suite ensures that the SUT code is executed.	N/A	S1
↳Requirement/ feature coverage completeness	TS	The degree to which a test suite ensures that the system requirements are addressed.	N/A	S2
Diversity *	TS	The degree of dissimilarity among test cases in terms of static (e.g., test data, test steps, execution traces, code statements) or dynamic (e.g., execution history) information.	N/A	S1, S3
Fault detection *	TS, TC	The degree of effectiveness and efficiency of identifying defects that a [test suite | test case] can achieve.		S1
↳Fault detection capability/ effectiveness	TS, TC	The more defects a [test suite | test case] finds, the more effective it is.		S1, S4
↳Fault detection efficiency	TS, TC	How quickly a [test suite | test case] can discover faults.		S1, S2
Maintainability *	TS, TC	The capability of a [test suite | test case] to be modified for error correction, improvement, or adaption to changes in the environment or the requirements.		S5, S2
↳Analyzability	TS, TC	The property of a [test suite | test case] to be diagnosed for deficiencies or causes of failures in the [test suite | test case], or for identifying which parts to be modified.		S6, S2
↳Co-evolution & maintenance	TS, TC	[Test suites | Test cases] and production code should be developed and maintained synchronously		S7, S8
↳Homogeneity	TS, TC	The degree to which the design of test cases in a test suite follows the same rules.	The degree to which the template for test cases follows the same rules.	S9
↳Self-contained	TS, TC	The degree to which test cases in a test suite can be executed without relying on the execution of other test cases.	The degree to which a test case can be executed without relying on the execution of other test cases.	S1, S10, S5
↳Traceability	TS, TC	The degree to which test cases in a test suite can be linked to other related artifacts such as tested requirements, features, source code, and issues.	The degree to which a test case can be linked to other related artifacts such as tested requirements, features, source code, and issues.	S1, S9, S5
Reliability *	TS, TC	The capability of a [test suite | test case] to maintain a specific level of performance under different conditions. "Performance" expresses the degree to which specific needs and requirements towards the [test suite | test case] are satisfied.		S1, S10, S2

Table 1 continued from previous page

Quality Attribute	For	Description in case of test suites	Description in case of test cases	Ref
↳Repeatability	TS, TC	A [test suite | test case] should produce the same results under fixed conditions.		S1, S9, S5
Resource efficiency *	TS, TC	The amount of resource (e.g., man-effort, computation, or test environment) needed by a [test suite | test case] to execute.		S1, S9, S5
Reusability *	TS, TC	The degree to which test cases of a test suite can be used as parts of another test suite.	The degree to which parts of a test case can be used as parts of another test case.	S1, S2
↳Changeability	TS, TC	The degree to which the structure and style of a [test suite | test case] allow changes to be made easily, completely, and consistently.		S1
↳Universal	TS, TC	The degree to which a [test suite | test case] can be used in different test environments.		S1
Simplicity *	TS, TC	How simple a test suite is in terms of the number of contained test cases, and execution steps.	How simple a test case is in terms of the test logic, the number of contained test cases, the number of steps.	S9, S10, S5, S11
↳Single responsibility	TC	N/A	A test case should focus on verifying just one requirement, condition or behavior of SUT.	S10
Usability *	TS, TC	How easy it is to execute a [test suite | test case].		S2
↳Completeness	TS, TC	A [test suite | test case] should contain all relevant information for its execution.		S1, S9, S5, S12
↳Flexibility	TC	N/A	The degree of freedom in test execution that the test engineer can utilize.	S9, S2
↳Learnability	TS, TC	How easy it is to learn how to execute a [test suite | test case].		S1, S2
↳Readability	TS, TC	How expressive a [test suite | test case] can be (using magic numbers, branching, and inexpressive naming conventions in test code can reduce its readability).		S10, S13
↳Understandability	TS, TC	How easy it is for test users to know whether a [test suite | test case] is suitable for their needs.		S9, S2

[*] Main quality attribute (QA)
[↳] Sub-quality attribute. Note that not all main QAs have a sub-QA(s).
S1: (Tran et al., 2021)
S2: (Zeiss et al., 2007)
S3: (De Oliveira Neto et al., 2018)
S4: (Chernak, 2001)
S5: (Adlemo et al., 2018)
S6: (ISO/IEC JTC 1/SC 7, 2011)
S7: (Garousi et al., 2015)
S8: (Zaidman et al., 2008)
S9: (Tran et al., 2019)
S10: (Bowes et al., 2017)
S11: (Kochhar et al., 2019)
S12: (Athanasiou et al., 2014)
S13: (Daka et al., 2015)

In the third section (Q8–9), we presented each respondent with the list of sub-quality attributes of the *important* main attributes of either test cases (Q8) or test suites (Q9). For each corresponding sub-quality attribute, the respondents were asked to (i) select challenges that apply to the sub-quality attribute and (ii) describe any current support they have had to address the challenges. At this point, free text fields were for respondents to describe the current support. Meanwhile, the three challenges that respondents could choose from are: Challenge C_{def} - *Inadequate definition*, Challenge C_{metric} - *Lack of useful metrics*, and Challenge C_{review} - *Lack of an established review process* (Juhnke et al., 2021). Note that in the study of Juhnke et al., these challenges are listed as the types of challenges related to the

quality assurance of test case specification. Even though their context was test specifications, we decided to use them in our survey as they still cover three fundamental aspects: how to define the quality (in terms of quality attributes), how to measure the quality, and how to maintain such quality. Nevertheless, the respondents could share extra information or specify different challenges in a free text field.

Note that Juhnke et al. also included concrete challenges under each of these types. For example, the authors listed "Requirement coverage is the only known quality metric" as a concrete challenge under "Lack of useful metrics". Here, we did not make use of the concrete challenges under each of these types as we did not want to limit the survey participants to fined-grain challenges that might not apply to their contexts while distracting them from seeing the whole picture. Additionally, the respondents could select a "None" option in case they believed the quality attribute was not challenging for them to deal with. Finally, the questionnaire concluded with closing questions for respondents to provide any extra comments (Q10) and contact information (Q11). An example of a survey answer can be viewed via https://doi.org/10.6084/m9.figshare.23309702.v1.

3.3 Survey instrument evaluation

We developed and improved the questionnaire based on discussions among the co-authors, covering the question logic, wording, and understandability. Once the questionnaire was in a presentable state, we conducted a pilot study with seven external assessors (three researchers and four practitioners). The questionnaire was refined based on their feedback, which was mainly about the clarity of terms and questions in the survey.

3.4 Survey distribution

The target population for the survey was software engineering professionals with working experience related to software testing. They are developers, testers, test architects, etc. To recruit participants, we followed suggestions given by de Mello et al. (2014), who proposed a systematic approach to recruit participants for Software Engineering surveys from LinkedIn[2]. Their proposed strategy helped us draw a large-scale sample and increase subject heterogeneity.

First, we used the Group Search feature in LinkedIn to search for groups related to software testing. The search was conducted in July 2022 and returned 243 groups. The first author read the groups' descriptions and excluded 40 groups based on the following exclusion criteria:

- has a vague or no description;
- has its description out of the scope of software testing;
- restricted members to a region or country only;
- not in English;
- focuses on job advertisement or headhunting;
- targets researchers, students, or learners.

LinkedIn limits the maximum number of groups a LinkedIn user can be part of to 100 groups and the total number of pending requests to join to 20 groups only. Hence, we sorted the 203 (243-40) remaining groups according to the number of members and then applied to join groups with the highest numbers of members first. Our reasoning is that the higher the

[2] https://www.linkedin.com/

number of members in a group, the wider the coverage of relevant professionals in the group. After three months of sending requests to join the sorted groups, the first author became a member of 20 groups (details in Table 2).

Considering the gross number of members (905,983) of the 20 accepted groups already covers 81,53% of the gross number of the 203 groups (1,111,218) and the time constraint of the survey study, we believed that recruiting participants from the 20 groups would be sufficient; hence, we stopped sending requests to join the other groups.

Having the 20 groups (details in Table 2) as our sampling frame, the gross number of members of the 20 groups was 905,983 (accessed on 2022-07-15). Similar to de Mello et al. (2014), we used Bartlett et al.'s formula (Kotrlik and Higgins, 2001) to calculate our theoretical sample size. We aimed for a confidence level of 95%, a confidence interval of 8, and the percentage occurrence of a state or condition of 0.5 (recommended by Bartlett et al.), and hence, our theoretical sample size was 151. To avoid selection bias, for each selected group *i*, the number of distinct members $\mathbf{X_i}$ to be added to the sample was in proportion with

Table 2 LinkedIn groups and members for the survey recruitment

ID	Group name	Num of members	Sample size (in theory)	Sample size (in practice)	Distinct members to extract
1	Software Testing & Automation	287,255	48	2,500	2,500
2	Agile and Lean Software Development	192,649	32	1,700	2,500
3	Quality Assurance – QA Professional, Testing, Test Automation	129,788	22	1,149	1,500
4	Software Testing and Quality Assurance group	71,113	12	634	1,000
5	Ministry of Testing – the online software testing community	48163	8	432	1,000
6	Selenium Testing	36,521	6	330	1,000
7	Software Testing and QA	28,925	5	254	700
8	Software Testing Profession	24,114	4	222	700
9	QA/Testing	23,343	4	215	700
10	Quality Assurance Testing	16,060	3	143	500
11	UK & Europe Software Test & QA Forum	13,346	2	117	500
12	Women In Software Engineering (WISE)	11,592	2	102	500
13	Automation & Manual Testing Group	6,534	1	57	100
14	Junior Testers	4,257	1	37	100
15	Software Test Engineering	3,111	1	27	100
16	Test People - Testing Professional Open Networkers	2,787	0	24	100
17	Software Testing Bootcamp	2,525	0	22	100
18	Association for Software Testing	1,551	0	14	100
19	Modern QA & Testing	1,298	0	11	100
20	Software Development Engineer in Test || Test Automation || Quality Assurance Tester/Analyst	1,051	0	9	100
	Total	905,983	151	8,000	13,900

the gross number of members in that group, that is:

$$\mathbf{X_i} = \frac{\mathbf{GS_i}}{\mathbf{GS}} \times \mathbf{SS}$$

where
- $\mathbf{X_i}$ is the number of distinct members in group *i* to be added to the sample
- $\mathbf{GS_i}$ is the gross number of group *i*
- $\mathbf{GS}$ is the total gross number of members of all 20 groups.
- $\mathbf{SS}$ is the theoretical optimum sample size (151).

Table 2 shows the number of distinct members from each selected group added to the sample size (151). Note that a larger sample size gives a lower margin of error and higher confidence. However, it also requires a larger recruitment size, which was, in our case, not feasible considering the time required to send LinkedIn messages manually.

Our next step was to extract distinct members from each selected group for recruitment (invitation). According to de Mello et al. (2014), the average completion rate through professional social networks lies between 3% and 4%. Hence, by setting an expected completion rate of 3%, our first estimation of the recruitment size was 5033 practitioners to achieve the stated sample size (151). After sending the survey invitations to the first selected group, we noticed that while our response rate was around 9.2%, our completion rate lay around 2% instead of 3–4% as estimated initially. Hence, we increased our recruitment size to 8000 to make sure that we could acquire the planned sample size of 151.

In principle, we could have sent survey invitations to the first 8,000 practitioners extracted from the selected groups. However, LinkedIn tends to have a bias in presenting group members (de Mello et al., 2014) (i.e., tend to present members who connect directly to the user (first degree) or to other members who connect directly to the user (second degree)). To mitigate this issue, we extracted more than 8,000 LinkedIn profiles so that we could randomly select the required number of distinct members (Baltes and Ralph, 2022). In this recruitment process, before randomly selecting members, we first removed duplicate members and excluded members who had no description or whose descriptions did not indicate any working experience in software testing, i.e., recruiters and students. The number of distinct members we recruited (invited) from each selected group is shown in the last column in Table 2.

Besides using LinkedIn as a source for sampling, we also used personal networks and searched for relevant practitioners from the Internet to conduct convenience sampling and snowball sampling. We used these non-probabilistic sampling approaches to increase the total number of responses. It is also worth noting that we assigned a different survey link to each LinkedIn group and to each direct contact or source of contacts for traceability.

The data collection started on September 21, 2022 and ended on January 31, 2023. In total, we collected 828 responses, of which 183 were complete (meaning all the questions were answered). Among these 828 responses, 35 were through personal networks, and 793 were through LinkedIn.

3.5 Data analysis

In total, we obtained 828 registered responses, of which 354 are valid, including 183 complete responses and 171 partial responses that completed at least Q6 or Q7 (the questions on which respondents were asked to rank the importance of the main quality attributes of either test cases

(Q6) or test suites (Q7)). We did not discard these 171 partial responses as they potentially contain information to answer RQ2.

For the two context dimensions, namely the size of SUT and the size of the test suite, we received only 31 and 25 responses, respectively, for the test suite data set, and 52 and 40 responses, respectively, for the test case data set. Most respondents commented that it was not easy for them to know the sizes or they could not obtain such information. Since the sizes of the data sets are too small to yield statistically significant results regarding the correlation between these two dimensions and the perceived importance, we decided not to investigate these two dimensions in this study.

Similarly, more than 90% of the respondents did not provide details regarding any current support they have had to tackle the challenging quality attributes. Since this question was asked at the end of the survey, we suspect that the respondents wanted to conclude the survey and, hence, did not answer this question. As the provided information regarding the support is not enough for us to draw any significant conclusion, we also decided not to focus on this aspect in this study.

We started the data analysis by examining the "Other" answers, i.e., answers containing free texts, for the survey questions Q2–4 (eliciting context information) and Q8–9 (eliciting challenges). Using a thematic coding approach (Cruzes and Dyba, 2011) to analyze the free text answers, we studied each "Other" answer to see if we could consolidate the answer with an existing answer. If this was not the case, i.e., the answer presented new information, we created a new code under the corresponding question.

Consequently, the free text answers to Q2–4 of 83% responses (293 out of 354 valid responses) provided sufficient information to form reasonable software-testing contexts for the analysis to answer RQ2. The remaining 17% were incomplete or lacked contextual information to interpret the answers and were excluded from the analysis. An example of excluded answers is "Working across multiple teams as well for system testing." for the survey question Q2, which asks participants about their testing activities. Another excluded one is "No control of this" as the answer to the type of development process for the system under test (part of question Q3.1). Note that we did not consider these remaining 17% as invalid responses as it could be that the respondent and us did not share the same understanding of the related concepts. Hence, we still used these responses to answer the other research questions as long as the research questions did not require information regarding software-testing contexts.

With respect to the survey questions Q8–9, we did not receive any responses with only insufficient free text answers. It means that all responses containing answers to Q8–9 were used for the analysis to answer RQ3. Since RQ1 did not require answers (Q2–4) or (Q8–9), a response was included in the data set to answer RQ1 as long as it contained answers to Q6 or Q7. As a result, we had 354 responses to answer RQ1, 293 responses to answer RQ2, and 183 responses to answer RQ3.

It is also worth noting that since our survey required the respondents to select either to answer questions regarding the quality of test cases or of test suites (Q5), the total responses actually contained two exclusive data sets: one data set regarding the test case quality and one data set regarding the test suite quality. To ease the discussion, we call the two data sets the test case data set and the test suite data set. Figure 2 illustrates our responses filtering step with respect to the two mentioned data sets. The two data sets are combined to answer RQ1. With RQ2 and RQ3, we analyzed each data set independently.

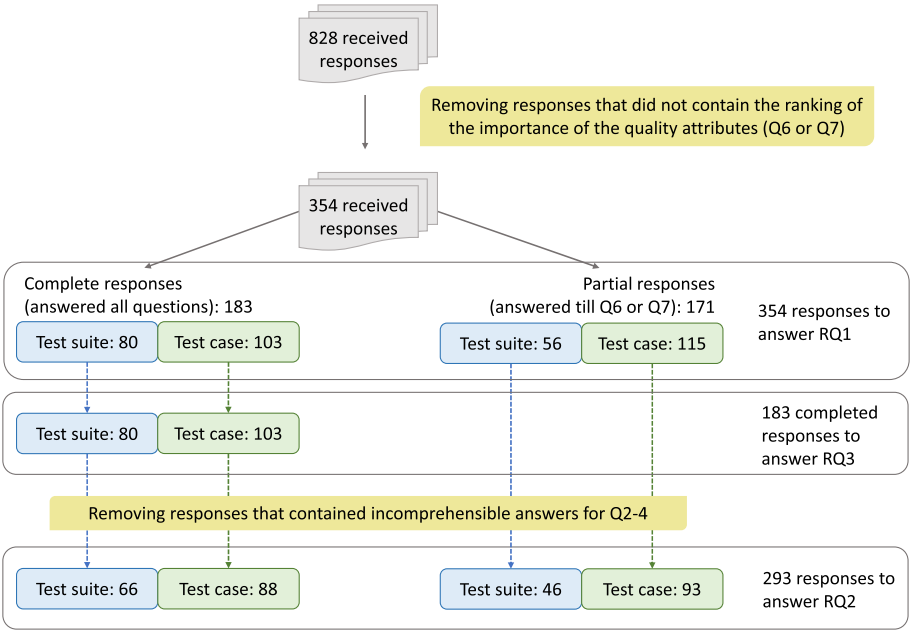

Fig. 2 Survey responses to answer research questions

3.5.1 Statistical tests

We followed Sheskin's guideline (Sheskin, 2020) to select statistical tests and then conducted the tests in IBM SPSS Statistics 28 to answer the research questions. Generally, to decide which statistical tests to use, we assessed three points: (1) the study design (associations, predictions, group differences, reliability, etc.), (2) the type of collected data, and (3) the number of independent and dependent variables (Sheskin, 2020). Below, we explained our selection of tests and our reasoning for each research question.

RQ1

First, we conducted a chi-square test of independence (at a 95% level of significance) to check whether some quality attributes (of either test cases or test suites) were perceived as significantly more important than the other attributes. Then, we performed post hoc comparisons of rates of importance votes by pairs of quality attributes. It is worth noting that each post hoc comparison was a separate chi-square test on each pair of quality attributes. Hence, we used Bonferroni correction to control Type I error across all of these post hoc tests.

Second, to verify if novice practitioners would be more likely to classify more quality attributes as important than senior practitioners, for each data set, we ran a Pearson's correlation (Sheskin, 2020) to assess the relationship between working experience and the percentage of quality attributes classified as important.

RQ2

We first ran TwoStep Cluster analysis (provided by SPSS Statistics 28) on each data set (the test case data set and the test suite data set) to explore any possibility of grouping the respondents according to their ranking of the importance of the quality attributes. This clustering step allowed us to observe similarities and differences in the software-testing contexts

between the groups of respondents with different views on the importance of the quality attributes. To verify if such similarities and differences in the contexts could actually influence how important quality attributes were perceived, we conducted the Ordinal regression models and Multinominal Regression models (Sheskin, 2020). The models were constructed for the test case data set and the test suite data set separately.

The regression model (Sheskin, 2020) took into account all the context dimensions (multiple independent variables) and one quality attribute (one dependent variable). Hence, there were nine regression models for nine quality attributes of test suites and seven models for seven quality attributes of test cases. Note that we did not include the context dimension *Testing tool* in the regression models. It is because this dimension is dependent on the other context dimensions (the testing tool selection depends on the testing level, practice, etc.), and hence, violated one of the assumptions of the ordinal logistic regression test.

With each regression model, we first checked the result of the test of parallel lines to verify whether the assumption of proportional odds is met. If the result of the test of parallel lines was **not** statistically significant ($p > 0.05$), meaning that the required assumption was met, we proceeded with checking the result of the Model Fitting Information, which is an overall measure of whether the model fits the data well. If the significance value indicated in the Model Fitting Information was statistically significant ($p < 0.05$), we examined the results of the Tests of Model Effects, which report whether each independent variable has a statistically significant effect on the dependent variable ($p < 0.05$). We used the results of the Tests of Model Effects to verify the correlation between the context dimensions and the quality attributes' importance levels.

If the required assumption of proportional odds was not met, i.e., the result of the test of parallel lines was statistically significant ($p < 0.05$), we instead ran multinomial logistic regression for the corresponding quality attribute. Note that multinomial logistic regression does not consider the ordinal nature of our dependent variables (the importance levels of the quality attributes). However, it is recommended as an alternative if the ordinal model does not meet the assumption of proportional odds (García-Pérez, 2013). For each multinomial regression model, we first checked the Model Fitting Information, which is an overall measure of whether the model fits the data well (same as for the Ordinal regression models). If the significance value indicated in the Model Fitting Information was statistically significant ($p < 0.05$), we examined the results of the Likelihood Ratio Tests, which report whether each independent variable has a statistically significant effect on the dependent variable ($p < 0.05$). The results of the Likelihood Ratio Tests helped us verify the correlation between software-testing contexts and the quality attributes' importance levels.

RQ3

Statistically, we needed a test to determine whether there was a statistically significant difference in the challenges votes among the quality attributes. Hence, the one-way repeated measures analysis of variance (ANOVA), which is also referred to as a within-subjects ANOVA, was the most suitable test (Sheskin, 2020). Our continuous, dependent variable was the percentage of respondents voting for each challenge for each quality attribute, while our within-subject factor (the independent variable) consisted of categorical levels, i.e., the challenges. Before running the one-way repeated measures ANOVA on each of the two data sets (test cases and test suites), we used Shapiro-Wilk's test ($p > 0.05$) to check whether each data set had outliers and was normally distributed. If the normality assumption is not met (Sheskin, 2020), we used the Friedman test, which is commonly used instead of ANOVA.

4 Results and analysis

In this section, we first describe the respondents' demographics (working experience and software-testing contexts), then present results and analysis according to our three research questions. Note that the analysis centers around the two data sets, the test case data set and the test suite data set. For clarity, the test case data set contains answers from respondents discussing test case quality, while the test suite data set contains answers from respondents discussing test suite quality (as explained earlier in Section 3.5).

4.1 Demographics

Overall, the working experience of 354 respondents varied from less than one year to 42 years of experience, with a median of 8 years and a mean of 10.23 years; 90% of the respondents had two to 25 years of experience. Figure 3 illustrates the distribution of the respondents' working experience. In this figure, we divide the respondents into bins (groups) with a uniform range of 5 years of experience.

Figures 4 and 5 present the context information of the respondents under eight context dimensions (Testing activity, Testing level, Testing type, Testing practice, Automation activity, Application domain of SUT, Type of SUT, and Type of development process) for the test case data set and the test suite data set.

Our general observation, based on Figs. 4 and 5 is that the context information is highly similar from both data sets. More specifically, the respondents from both data sets were mainly working at the *system testing level* for *functional testing*, applying *automation in tests execution*. They primarily tested *systems software* in the *finance* domain and used *agile practices* in their development process. The differences only come from their testing activities and testing practices. The most common testing activities were *test environment and data management activities* from the test suite data set and *test design and implementation activities* from the test case data set. For the testing practice, the respondents from the test suite data

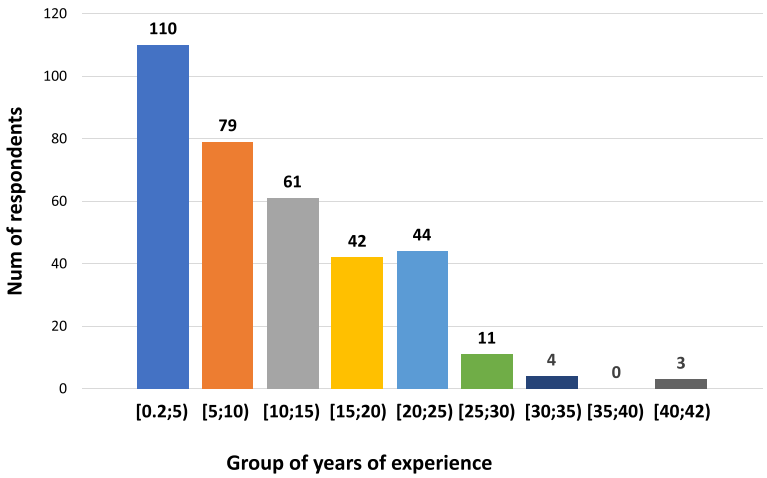

Fig. 3 Responses distribution over the working experience

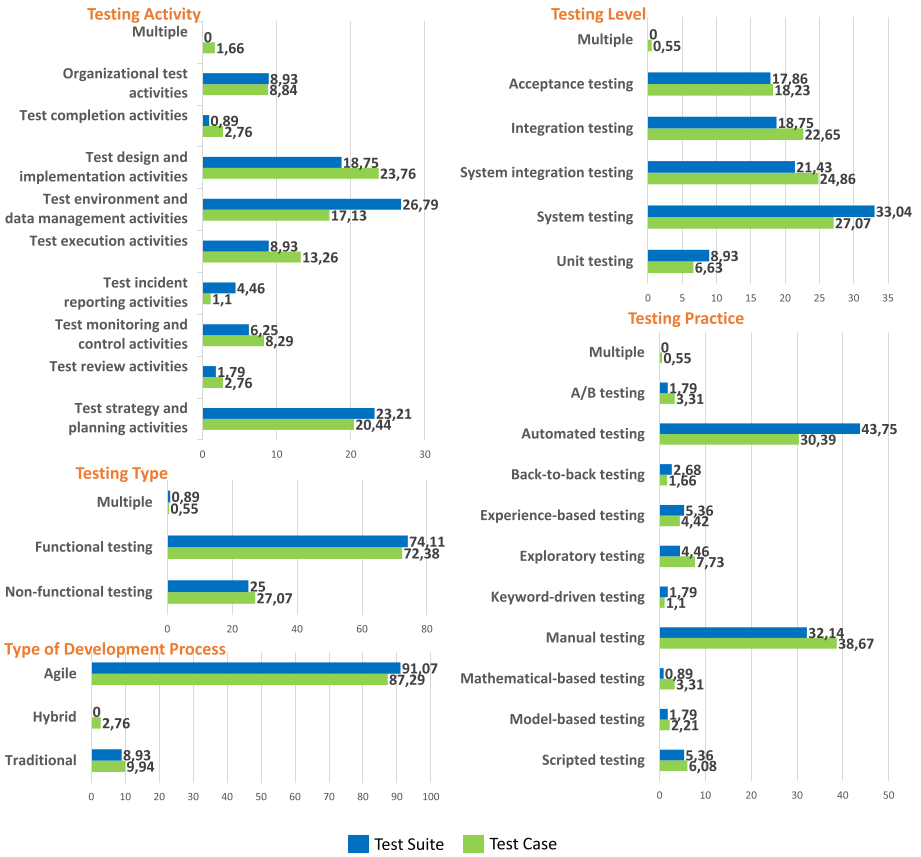

Fig. 4 Respondents' software-testing contexts in terms of Testing activity, Testing type, Type of development process, Testing level, and Testing practice

set did more automated testing, whereas those from the test case data set mainly did manual testing.

4.2 RQ1 - Important quality attributes of test cases and test suites

With this research question, our focus was to investigate whether the importance of the quality attributes of test cases and test suites is perceived differently in practice.

4.2.1 Differences in importance votes among quality attributes

We wanted to study if practitioners considered some quality attributes to be more important than others. Hence, we focused on analyzing the similarities and differences in the rates of importance votes (under three importance levels: "Important", "Optional", and "Not relevant at all") among the quality attributes. Since the quality attributes of test cases are different from those of test suites, we analyzed the two data sets: (1) the test case data set and (2) the test suite data set separately. Figure 6 shows the percentage of votes (responses) each quality

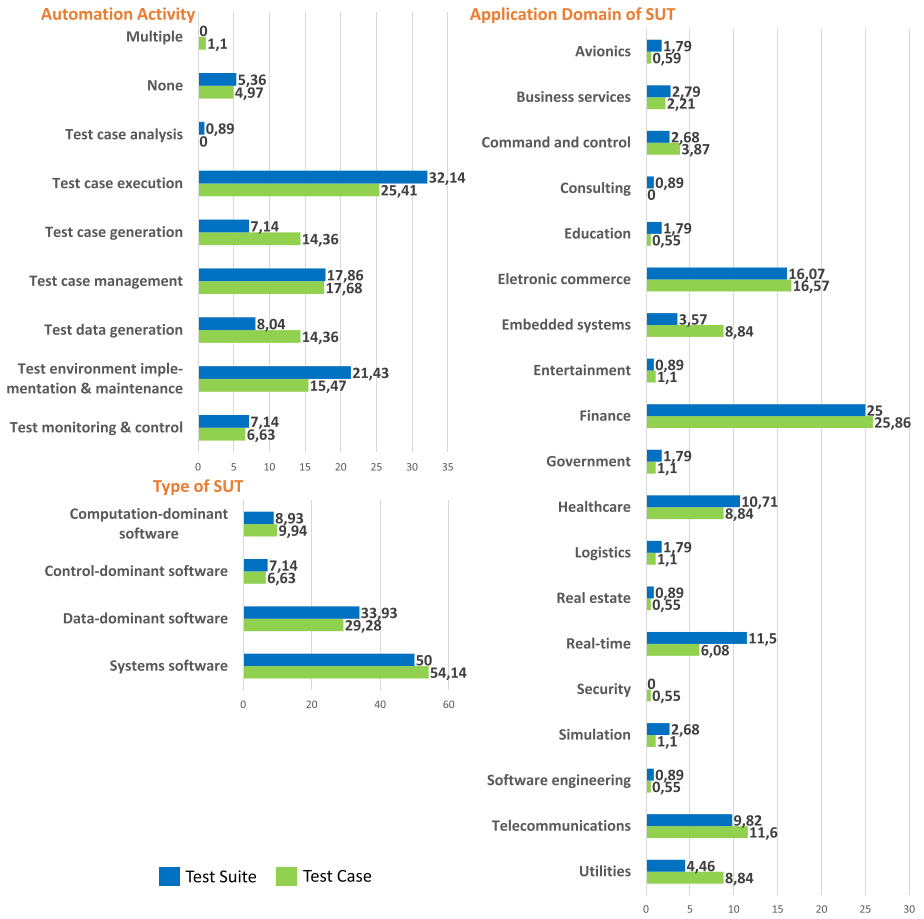

Fig. 5 Respondents' software-testing contexts in terms of Automation activity, Type of SUT, and Application domain of SUT

attribute of test cases and test suites receives under three importance levels: "Important", "Optional", and "Not relevant at all".

For test-suite quality, we can observe that the majority of respondents agreed that all test-suite quality attributes are important. Based on Fig. 6, the quality attribute that the respondents regarded as the most important was Fault Detection (agreed by 85% of the respondents), followed closely by Usability (83%). Likewise, Coverage, Reliability, and Maintainability were also regarded as important, as agreed by at least 75% of the respondents. On the other hand, Simplicity and Resource Efficiency were voted as the least important ones (52% and 54%, respectively).

We can also see that there were 15 respondents (out of 136) who marked at least two (out of nine) main quality attributes as "Not relevant at all". Among these 15 respondents (ranging from two years to 25 years of experience), only two respondents (with twelve and 24 years of experience) considered six and five quality attributes as "Not relevant at all", three other respondents classified three attributes as "Not relevant at all", while the other ten marked two attributes as "Not relevant at all". Based on this observation, we can see that the respondents

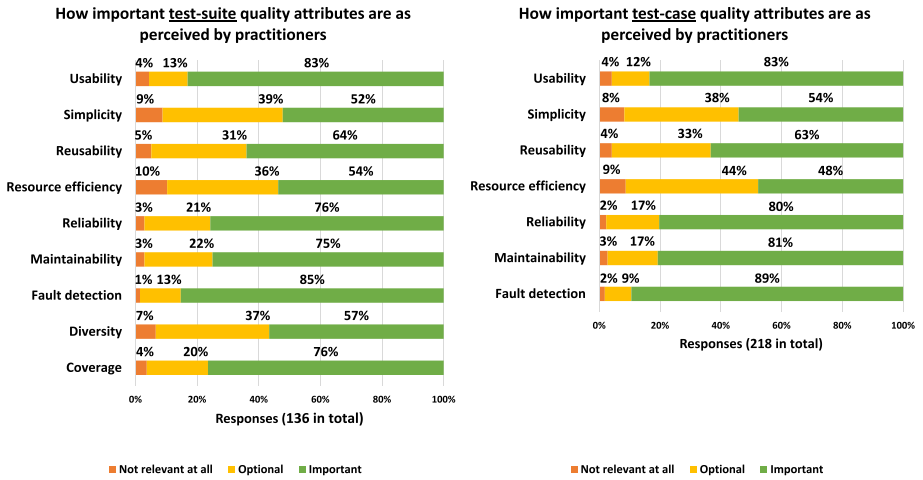

Fig. 6 Perceived importance of the quality attributes of test suites and test cases by practitioners

who wanted to emphasize the importance of only a few quality attributes of test suites were from the minority group (2 respondents), while the rest of them regarded more attributes as important or at least optional.

We observe similar results for test-case quality. While all quality attributes were perceived as important by many respondents, Fault Detection was regarded as the most important one (by 89% of respondents), followed by Usability (83%), Maintainability (81%), and Reliability (80%). The data also shows Simplicity and Resource Efficiency were considered the least important attributes (54% and 48%, respectively).

Also, we note that 14 respondents (out of 218) classified at least two main quality attributes of test cases as "Not relevant at all". Among these 14 respondents (ranging from 3 months to 25 years of experience), only two respondents (with 4 months and seven years of experience) considered six quality attributes as "Not relevant at all", five other respondents classified three quality attributes as "Not relevant at all", while the other seven marked two quality attributes as "Not relevant at all". With this observation, similar to the case of test-suite quality, we can see that the respondents who wanted to emphasize the importance of only a few quality attributes of test cases were from the minority group (2 respondents), while the rest of them considered more attributes as optional or important.

The chi-square test of independence (at a 95% level of significance) revealed statistically significant differences between seven quality attributes of *test cases* regarding the importance votes (grouped by the three importance levels) (χ^2 (12) = 170.504, $p < 0.001$). Post hoc comparisons of rates of importance votes by pairs of *test-case* quality attributes indicated that Fault Detection, Maintainability, Reliability, and Usability had significantly higher rates of votes for the highest importance level than Resource Efficiency, Reusability, and Simplicity (the test results can be viewed https://doi.org/10.6084/m9.figshare.23309702.v1).

Likewise, the chi-square test of independence showed significant differences between nine quality attributes of *test suites* regarding the importance votes (χ^2 (16) = 87.282, $p <$ 0.001). Post hoc comparisons of rates of importance votes by pairs of *test-suite* quality attributes revealed that Fault Detection, Usability, Coverage, Maintainability, and Reliability had significantly higher rates of votes for the highest importance level than Resource Efficiency and Simplicity (the test results can be viewed via https://doi.org/10.6084/m9.figshare.

23309702.v1). Additionally, Fault Detection and Usability had significantly higher rates of votes for the highest importance level than Diversity and Reusability.

4.2.2 Perceived importance of quality attributes and practitioners' working experience

We wanted to study if novice practitioners would be more likely to classify more quality attributes as important than senior practitioners. Since novice practitioners have less experience, and might have limited perspective regarding the quality of test cases and test suites, they might overestimate or underestimate the importance of quality attributes. Senior practitioners, on the other hand, might want to focus more on a suitable selection of quality attributes for a better positive return on investment.

Our interest in the association between the working experience and the selection of important quality attributes was inspired by studies (Chatzipetrou et al., 2020; Lo et al., 2015) showing that practitioners' working experience could influence their perception of certain software-engineering-related aspects. For example, Chatzipetrou et al. (2020) have shown that there exists a correlation between the working experience and the consideration of what software component attributes are important when selecting new components.

Figure 7 presents how the number of important quality attributes (of test case and test suite) changes with respect to the number of years of experience. The dot scale illustrates how many respondents have the same number of years of experience while marking the same number of attributes as important. For example, for test cases, the figure shows that one respondent with 40 years of experience marked five out of seven quality attributes as important, while twelve other respondents with two years and a half of experience marked five out of seven attributes as important.

Figure 7 shows no clear pattern in how the number of important quality attributes varies in relation to the number of years of experience. In other words, it is hard to tell if novice practitioners would consider more quality attributes as important than seniors or the other way around. Hence, we ran a Pearson's correlation (Sheskin, 2020) for each data set (the test suite data set and test case data set) to assess the relationship between working experience and the percentage of quality attributes classified as important.

For each data set, the analyses showed the relationship to be linear with both variables normally distributed, as assessed by Shapiro-Wilk's test ($p > 0.05$), and there were no outliers. The Pearson test indicated that there was no statistically significant correlation between working experience and the percentage of *test-suite* quality attributes classified as important, $r(136) = -0.034$, $p = 0.698$. Likewise, the Pearson test also showed that there was no statistically significant correlation between working experience and the percentage of *test-case* quality attributes classified as important, $r(218) = -0.024$, $p = 0.723$.

Finding 1: With respect to test-case quality attributes, practitioners perceived Fault Detection, Maintainability, Reliability, and Usability as significantly more important than Resource Efficiency, Reusability, and Simplicity. In the case of test-suite quality, Fault Detection, Usability, Coverage, Maintainability, and Reliability were significantly more important than Resource Efficiency and Simplicity. Additionally, Fault Detection and Usability of test suites were significantly more important than Diversity and Reusability. There was no statistically significant correlation between the working experience and the number of quality attributes classified as *important*.

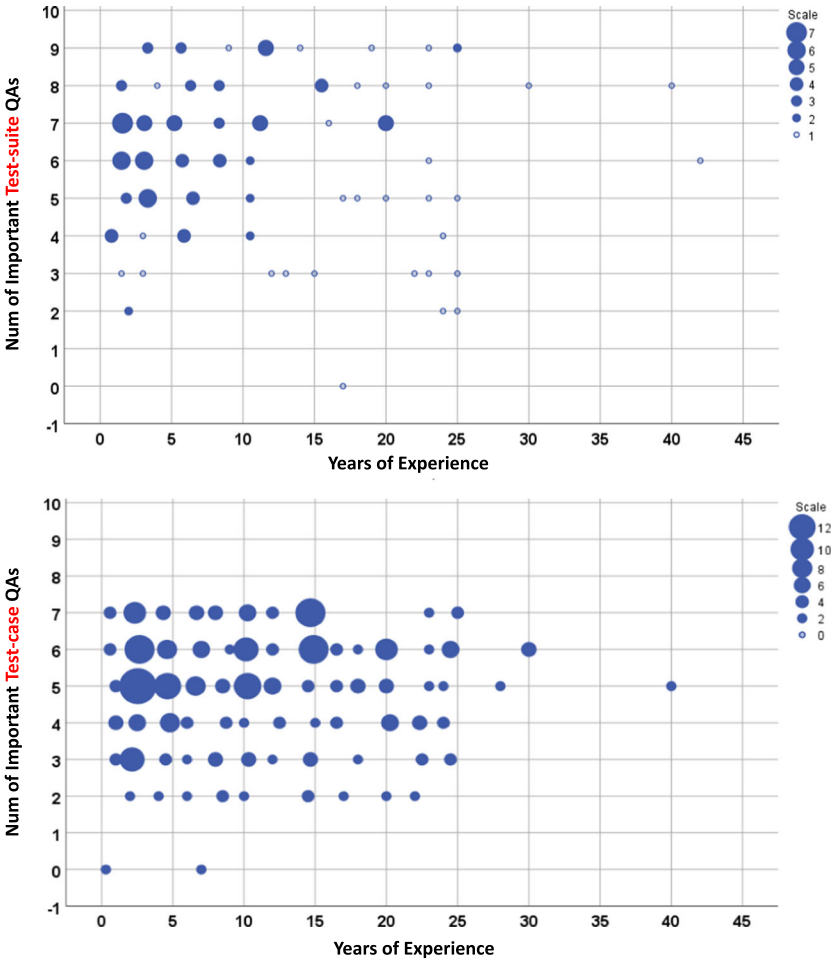

Fig. 7 Number of important quality attributes (QA) of test suites and test cases across years of experience

4.3 RQ2 - Correlation between software-testing contexts and the perceived importance of quality attributes of test cases and test suites

With this research question, we focused on collecting and analyzing the relationship between the software-testing contexts of the respondents and the quality attribute classification outcomes.

First, for each data set, the *TwoStep Cluster* analysis (provided by SPSS Statistics 28) returned two groups of respondents: the group of respondents who classified more quality attributes as important versus the other group of respondents. The quality of the clustering result was considered fairly good (average silhouette = 0.2).

For the test-suite quality, to ease the discussion, we call the two groups returned by the TwoStep Cluster analysis Group TS_1 (71 respondents) and Group TS_2 (41 respondents). While most of the respondents in Group TS_1 agreed that all quality attributes of test suites should be important, most respondents in Group TS_2 agreed that Diversity, Simplicity, Resource Efficiency, and Reusability should be optional or not relevant at all.

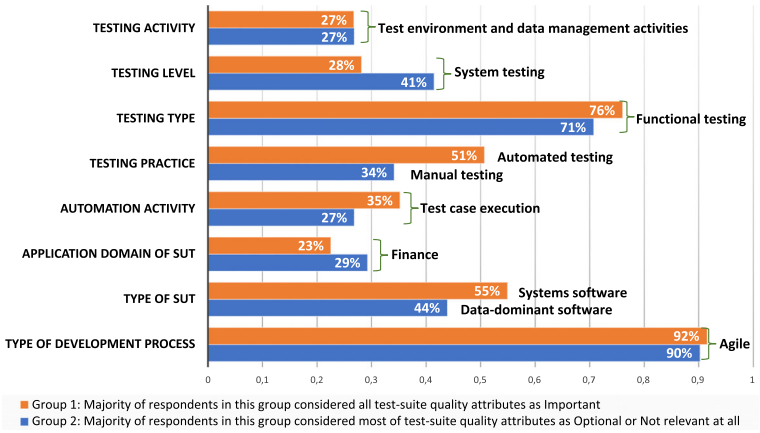

Fig. 8 Main differences in the software-testing contexts between respondents considering **test-suite** quality attributes (QAs) more important than those considering the QAs less important

Figure 8 illustrates the context differences between two groups TS_1 and TS_2. We can see that the contexts between the two groups are quite similar and consistent with the common context reported earlier (detailed in Section 4.1). The main differences between these two groups are that the respondents from Group TS_1 applied Automated testing as their testing practice while the respondents of Group TS_2 focused more on Manual testing. Moreover, Group TS_1 mostly worked with Systems software and Group TS_2 mainly worked with Data-dominant software.

For the test-case quality, also to ease the discussion, we call the two groups returned by the TwoStep Cluster analysis Group TC_1 (71 respondents) and Group TC_2 (110 respondents). While most of the respondents in Group TC_1 considered all quality attributes of test cases as important, respondents in Group TC_2 agreed that Simplicity and Resource Efficiency should be optional or not relevant at all.

Figure 9 illustrates the context differences between two groups TC_1 and TC_2. Even though the contexts between these two groups are rather similar, there are three context dimensions that distinguish the groups. The three dimensions include Testing activity, Testing level, and Testing practice. Most respondents from Group TC_1 were involved in Test design and implementation activities, whereas respondents from Group TC_2 mainly performed Test strategy and planning activities. With the Testing level, the majority of Group TC_1 focused on Integration testing, while most respondents from Group TC_2 focused on System testing. Furthermore, Automated testing was the main Testing practice of Group TC_1, and Group TC_2 mostly worked with Manual testing.

Overall, based on the outcome of the TwoStep Cluster analysis, despite the similarities, we can still see the differences in the software-testing contexts between the groups of respondents, which are distinguished by their diverse views on the importance of the quality attributes (of test cases and test suites). To verify if such differences in the contexts could actually influence how important quality attributes were perceived, we conducted the Ordinal regression models and Multinominal Regression models (Sheskin, 2020). The models were constructed for the test case data set and the test suite data set separately.

Table 3 presents the results of the statistical tests. For test-suite quality, the tests revealed that there were context dimensions that influenced the ranking of the importance of three quality attributes, namely Resource Efficiency, Reusability, and Usability. More specifically,

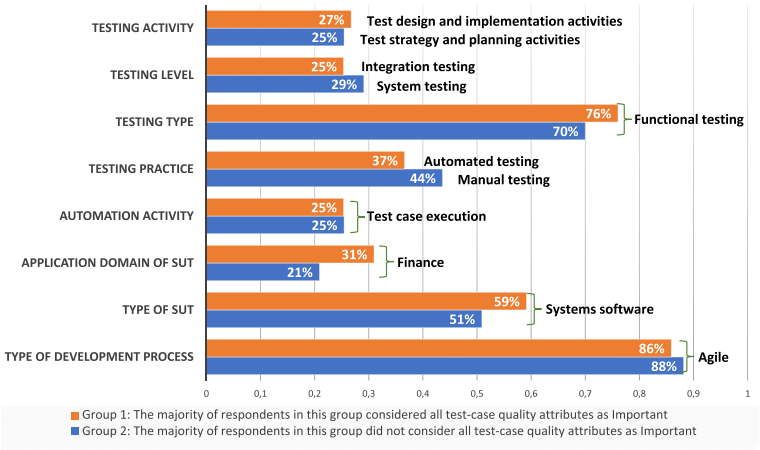

Fig. 9 Main differences in the software-testing contexts between respondents considering **test-case** quality attributes (QAs) more important than those considering the QAs less important

which Automation activity and Type of development process the respondents worked with had statistically significant effects on whether Resource Efficiency was perceived as important, optional, or not relevant at all. Likewise, which Testing activity the respondents were involved with had a statistically significant effect on whether Reusability was perceived as important, optional, or not relevant at all. In the case of test-suite Usability, even though the model met the assumption of proportional odds and the model fitted the data well, the convergence criteria were not satisfied. Hence, no effect could be reported in this case. Section 5 discusses in detail how these context dimensions influence the perception of the quality attributes.

Table 3 Results of Ordinal regression model and Multinomial regression model to answer RQ2

		Ordinal Regression Model			Multinominal Regression Model	
		Step 1: Test of Parallel Lines	Step 2a: Model Fitting Information	Step 2b: Tests of Model Effects	Step 3a: Model Fitting Information	Step 3b: Likelihood Ratio Tests
Test-Suite Quality Attributes	Coverage	1.000	0.175			
	Diversity	0.548	0.010	> 0.05		
	Fault detection	1.00	0.410			
	Maintainability	0.988	0.310			
	Reliability	< 0.001			1.000	
	Resource efficiency	0.381	0.014	Automation activity (p = 0.020) Type of development process (p = 0.001)		
	Reusability	0.882	0.013	Testing activity (p = 0.018)		
	Simplicity	< 0.001			< 0.001	Testing activity (p < 0.001) Application domain of SUT (p = 0.010) Type of development process (p = 0.027)
	Usability	1.000	0.001	Some convergence criteria are not satisfied.	0.391	
Test-Case Quality Attributes	Fault detection	0.001			0.577	
	Maintainability	0.994	0.097			
	Reliability	0.999	0.096			
	Resource efficiency	0.003			0.019	Testing level (p = 0.003) Testing practice (p = 0.008) Automation activity (p < 0.001) Type of SUT (p = 0.026)
	Reusability	0.649	0.071			
	Simplicity	0.103	0.014	> 0.05		
	Usability	0.996	0.090			

Regarding test-case quality, the four context dimensions, namely Testing level, Testing practice, Automation activity, and Type of SUT, had statistically significant effects on whether Resource Efficiency was perceived as important, optional, or not relevant at all. No statistically significant result was found for the other quality attributes of test cases.

Finding 2: Software-testing contexts that practitioners were involved in did have an influence on their perception of the importance of certain quality attributes. More specifically, the importance of Resource Efficiency, Reusability, Simplicity, and Usability of test suites depended on four context dimensions: Automation activity, Type of development process of SUT, Testing activity, and Application domain of SUT. The importance of Resource Efficiency of test cases was affected by four context dimensions, namely Testing level, Testing practice, Automation activity, and Type of SUT.

4.4 RQ3 - Challenges that apply to the important quality attributes of test cases and test suites

With this research question, we focused on collecting and analyzing the challenges that practitioners face in defining, measuring, and maintaining the important quality attributes.

Apart from the three provided challenges (Challenge C_{def} - Inadequate definition, Challenge C_{metric} - Lack of useful metrics, and Challenge C_{review} - Lack of an established review process), we found that the other challenges given by the respondents (in free text) could be grouped together to form another generic challenge, that is $C_{support}$ - *lack of external support* (with respect to a particular quality attribute). This fourth generic challenge includes specific challenges given by respondents such as "not enough time to achieve high maintainability", "lack of team members interested in achieving high quality", "lack of competence and experience of testing", "lack of product knowledge", "lack of support from team management", "Inadequate requirements definitions for new products", etc. Note that this fourth challenge does not cover the existing Challenge C_{review} "Lack of an established review process". Our three initially stated Challenges, together with the emerging one, represent four fundamental issues concerning a quality attribute of a test case or test suite, which are: cannot define the quality of test case or test suite quality (Challenge C_{def}), cannot measure the quality (Challenge C_{metric}), cannot review/maintain high quality (Challenge C_{review}), and cannot acquire support to achieve high quality (Challenge $C_{support}$).

Note that the respondents selected (or provided) challenges for the sub-quality attributes of each main quality attribute they marked as important. Hence, to obtain an overview of how the four challenges are selected, for each sub-quality attribute, we calculated the percentage of respondents selecting each challenge. The maximum number of votes that a sub-quality attribute received for a particular challenge was the number of respondents who classified the main quality attribute as important. For example, Fault Detection (of test cases) was classified as important by 89 respondents, 35 out of which selected Challenge C_{def} for Fault Detection Capability/Effectiveness. Hence, we report that for Fault Detection Capability/Effectiveness, among all the four challenges, Challenge C_{def} was voted by 39% (**35** $\div$ **89**) of the corresponding respondents. Figure 10 illustrates the calculation results.

As shown in Fig. 10, for both test cases and test suites, among Challenges C_{def}, C_{metric}, and C_{review}, none of the challenges received a remarkably higher percentage of voters than the others. Compared to Challenges C_{def}, C_{metric}, and C_{review}, Challenge $C_{support}$, which emerged from the data collection, had a significantly lower percentage of voters. Here, we were interested in verifying if this observation just happened by chance or if some chal-

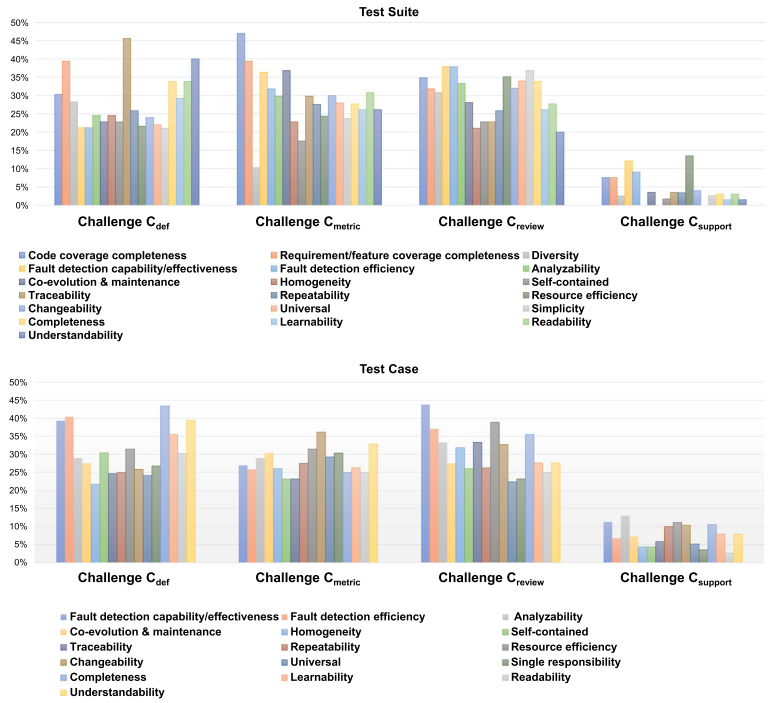

Fig. 10 Distribution of challenges that apply to the important quality attributes of test suites and test cases

lenge(s) were actually more common than others. This question stemmed from the finding in the study of Juhnke et al. (2021) where Challenges C_{def}, C_{metric}, and C_{review} were originally reported. Juhnke et al. found that Challenge C_{review} appeared to be the most popular challenge, followed by Challenge C_{metric}, then Challenge C_{def} in the context of test case specifications in automotive software testing.

Before running the statistical tests (ANOVA and Friedman tests) to verify our observation, we checked whether each data set (test cases and test suites) had outliers and was normally distributed. As a result, we found that the test suite data set had outliers and was not normally distributed, according to the assessed boxplot and Shapiro-Wilk's test ($p > 0.05$) respectively, while the test case data set met both assumptions, i.e., there were no outliers and the data was normally distributed. Hence, we used the Friedman test, which is commonly used instead of ANOVA if the normality assumption is not met (Sheskin, 2020), for the test suite data set and the one-way repeated measures ANOVA for the test case data set.

The results of the Friedman and ANOVA tests and their post-hoc tests are reported in Table 4. For the test case data set, Mauchly's test of sphericity indicated that the assumption of sphericity is violated ($p < 0.05$). Therefore, we interpreted the results using the Greenhouse-Geisser correction. Generally, we can see that there was a statistically significant difference in the challenge votes amongst the quality attributes. In particular, the post-hoc analysis revealed statistically significant differences in the percentage of votes between Challenge $C_{support}$ and any other challenges (Challenges C_{def}, C_{metric}, and C_{review}) for both test cases and test suites. The result is consistent with our data observation described previously.

Table 4 Results of one-way repeated measures ANOVA and Friedman test to answer RQ3

Test Suite			Test Case		
			Mauchly's Test of Sphericity	$\chi^2(5) = 14.90$, $p = .011$	
			Greenhouse-Geisser	$\varepsilon = 28.16$	
Friedman test	$\chi2(3) = 35.708$, , $p < 0.001$		ANOVA test	$F(2.111, 31.667) = 101.822$, $p < 0.001$	
Bonferroni post hoc test	Challenge C_{def} - Challenge C_{metric}	p = 1.000	Bonferroni post hoc test	Challenge C_{def} - Challenge C_{metric}	p = 1.000
	Challenge C_{def} - Challenge C_{review}	p = 1.000		Challenge C_{def} - Challenge C_{review}	p = 1.000
	Challenge C_{metric} - Challenge C_{review}	p = 1.000		Challenge C_{metric} - Challenge C_{review}	p = 1.000
	Challenge $C_{support}$ - Challenge C_{def}	p < 0.0005		Challenge $C_{support}$ - Challenge C_{def}	p < 0.001
	Challenge $C_{support}$ - Challenge C_{metric}	p < 0.0005		Challenge $C_{support}$ - Challenge C_{metric}	p < 0.001
	Challenge $C_{support}$ - Challenge C_{review}	p < 0.0005		Challenge $C_{support}$ - Challenge C_{review}	p < 0.001

Finding 3: The three proposed challenges that apply to the important quality attributes of test cases and test suites (Challenge C_{def} - Inadequate definition, Challenge C_{metric} - Lack of useful metrics, Challenge C_{review} - Lack of an established review process) were regarded as more or less equally by practitioners. In addition, we identified an emerging challenge (Challenge $C_{support}$ - Lack of external support).

5 Discussion

We discuss in this section the two main aspects of our results, covering (i) the perceived importance of test-case quality attributes and test-suite quality attributes and their correlation with software-testing contexts, and (ii) the challenges that apply to the important quality attributes.

5.1 The importance of quality attributes of test cases and test suites

As reported in Section 4.2, there was statistically significant evidence that Fault Detection, Maintainability, Reliability, and Usability were the most important quality attributes for both test cases and test suites. Likewise, Coverage, the quality attribute available for test suites only, was also one of the most important. When investigating the hypothetical correlation between the importance of quality attributes and the context dimensions (Section 4.3), we found that there were no statistically significant results on the influence of context dimensions on the perceived importance of these five important quality attributes (Fault Detection, Usability, Maintainability, Reliability, and Coverage). One possible explanation is that for achieving high-quality test cases and test suites, some quality attributes should always be the main focus, independent of context.

On the other hand, other quality attributes, namely Diversity, Resource Efficiency, Reusability, and Simplicity, were the least important ones. Furthermore, the difference in importance between the most and least important quality attributes was statistically significant. Therefore, our findings suggest that not all quality attributes are regarded as equally important in practice. This could be due to trade-offs between quality attributes. This argument is in line with findings from several studies that have studied the trade-off between

software quality attributes (Sas and Avgeriou, 2020; Wahler et al., 2017; Feitosa et al., 2015; Ali et al., 2012). For example, to achieve high quality with respect to Fault Detection Capability and Efficiency, practitioners might need to utilize computation power and expand the test environment as much as needed, which in turn might reduce Resource Efficiency. In another scenario, it could be that to improve Fault Detection Efficiency ("How quickly can a [test suite | test cases] discover faults"), practitioners might need to combine several test cases with the same setup and teardown into one bigger test case. In this second example, the Simplicity of test cases and test suites might reduce in proportion to the size of the merged test cases.

Besides the potential trade-off between quality attributes, the software-testing contexts can also be the reason why some attributes are considered more important than others (Section 4.3). Indeed, our statistical tests (Section 4.3) showed that certain context dimensions could affect how practitioners perceived the importance of four quality attributes of test suites: Resource Efficiency, Reusability, Simplicity, and Usability as well as Resource Efficiency of test cases. The following discussion touches on each of these quality attributes.

Particularly, regarding test-suite quality, the importance of Resource Efficiency significantly depended on what testing activity is automated and the type of development process of SUT. Resource Efficiency is defined as "The amount of resource (e.g., man-effort, computation, or test environment) needed by a test suite to execute" (Tran et al., 2021, 2019; Adlemo et al., 2018). Based on this definition, we can think of the following example. In the context of automated test case execution, Resource Efficiency might be perceived as less important than in the context where test cases are generated automatically but have to be executed manually by practitioners. Indeed, a practitioner explicitly mentioned that he or she "works only with automatic tests maintained on the local server, and even if a test is not efficient (e.g., takes too long), it is acceptable. I might decide to run it, for example, once a week." Regarding the influence of types of the development process, if the SUT is developed in a traditional way (Waterfall, Structured systems analysis and design method, or no formal process) or in a hybrid way (a mix of traditional approaches and Agile practices), Resource Efficiency might be perceived as more important than in the context where SUT development followed Agile practices. Generally, with the traditional or hybrid development approach, testing typically starts towards the end of the development life cycle after the coding has been done. Hence, it is reasonable that in such a context, higher demand on resources is needed since there is more pressure to have the whole SUT tested adequately, i.e., Resource Efficiency becomes an important factor.

Likewise, the importance of test-suite Simplicity depended on the type of testing activities, the application domains, and the type of development process of the SUT. Simplicity is defined as "How simple a *test suite* is in terms of the number of contained test cases, execution steps" (Tran et al., 2019; Bowes et al., 2017; Adlemo et al., 2018; Kochhar et al., 2019). Hence, from the point of view of practitioners who are responsible for test design and implementation activities or test execution activities, Simplicity might be a more important requirement than for those who are responsible for test environment and data management activities. Regarding application domains, it is known that for some domains, such as finance, as mentioned by one practitioner, it is common to outsource the development of some software components or the entire software system to a third party while the integration and customization would be done internally. Hence, test cases are initially developed to test user acceptance scenarios in order to cover what business users would consider sufficient. In this context, according to the same practitioner, "more complicated tests will usually find more defects, such as pairwise testing mixes systems and devices to find more defects. This is the situation when complicated tests are "better than simple ones." In other words, the simplicity of test suites, according to the

definition, might not be an important factor to consider in this context. Regarding the types of the development process, it is reasonable to believe that the Simplicity of test suites could be perceived as more important by practitioners who followed Agile practices than those who have developed SUT using the traditional approach. Overall, Simplicity in terms of code and design is one of the main principles of Agile methodologies that is seen as one of the main differences between Agile and traditional approaches.

Meanwhile, the importance of the Reusability of test suites depended on what testing activity practitioners were involved with (similar to the case of Simplicity discussed above). For clarity, test-suite Reusability is defined as "the degree to which test cases of a test suite can be used as parts of another test suite" (Tran et al., 2021; Zeiss et al., 2007). Again, based on this definition, it is fair to argue that Reusability is more important to practitioners who have been involved in test design and implementation activities than other activities, such as test execution activities or test incident reporting activities.

Regarding the quality of test cases, we found that the importance of Resource Efficiency varied depending on four context dimensions: Automation Activity, Testing level, Testing practice, and Type of SUT. On the one hand, similar to test suites, Automation Activity was one of the influencing context dimensions. On the other hand, we had no statistically significant evidence regarding the influence of the Type of development process on this quality attribute's perceived importance as found for test suites. For clarity, Resource Efficiency is defined as "the amount of resource (e.g., man-effort, computation, or test environment) needed by a test case to execute." (Tran et al., 2021, 2019; Adlemo et al., 2018), which is similar for test suites. We note here that the definitions of the quality attribute are similar between test cases and test suites, but the correlation findings are different. However, it is worth emphasizing that the differences in findings do not imply conflicts in conclusions. It just means that we did not obtain statistically significant evidence for some correlations.

With regard to Testing level, it is common that the execution time of unit test cases is shorter than that of higher-level test cases (such as system or system integration testing levels). Therefore, Resource Efficiency might not necessarily be the main focus for unit test cases as for test cases of higher testing levels. Regarding the testing practices, we argue that evidence-based testing or exploratory testing generally requires more resources (in terms of human effort) for test execution than other types of testing practices such as model-based testing or scripted testing. Hence, practitioners working with former testing practices might regard Resource Efficiency as more important than those with the latter testing practices. Finally, types of SUT could influence how the testing environments are set up. For example, with parallel and distributed systems, setup and maintenance of a test environment are major cost drivers. Resource Efficiency (in terms of test environments) for this type of SUT could be more important than other types of SUT.

5.2 Challenges that apply to the important quality attributes

In Section 4.4, we report four challenges and how they were voted differently across the quality attributes of test cases and test suites. Based on the collected data (illustrated in Fig. 10, amongst Challenges C_{def} (Inadequate definition), C_{metric} (Lack of useful metrics), and C_{review} (Lack of an established review process), we did not see any challenge which was more popular or common that the others. This finding suggests that practitioners still have difficulties with three fundamental aspects of test-case and test-suite quality, which are how to define, measure, and maintain the quality.

Meanwhile, Fig. 10 shows that Challenge $C_{support}$ (Lack of means to achieve high quality) was significantly less common than Challenges C_{def}, C_{metric}, and C_{review}. The results of the Friedman and one-way repeated measures ANOVA tests also supported this latter observation. Nevertheless, we must remark that Challenge $C_{support}$ was the emerging information collected from the "Other(s)" free text option. It means that the respondents were not informed about Challenge $C_{support}$ like the other challenges when answering the survey. This can explain why Challenge $C_{support}$ appeared to be less common than the proposed challenges. Nevertheless, the emergence of Challenge $C_{support}$ should not be neglected as it might be another common challenge but has not yet received much attention from researchers. In other words, we argue that Challenge $C_{support}$ should be more carefully investigated as it could potentially open new research gaps.

6 Limitations and threats to validity

In this section, we discuss the limitations of our sampling method then three types of validity threats for survey studies based on Kitchenham and Pfleeger's guideline for personal opinion surveys (Kitchenham and Pfleeger, 2008).

6.1 Limitations of the sampling method

There are two potential biases with our sampling approach: the bias due to the source of recruitment and the bias due to non-respondents, i.e., people who do not want to participate in the survey. These two biases might prevent us from acquiring survey results that are representative of the population being studied.

The first bias stems from the fact that we recruited participants mainly from LinkedIn. Even though we also sent survey invitations via personal contacts, the corresponding responses were too few to have any significant impact. There are other forums or platforms whose members may have different characteristics and views on the survey topic that we did not reach out to. Additionally, LinkedIn has a bias in displaying group members. We mitigated the issue from LinkedIn by extracting more members than required so that we could randomly select the needed number of people (as discussed in Section 3.4). Another issue is that we could invite only members from the 20 LinkedIn groups of which the first author was a member. To cope with this issue, we purposely joined groups with the highest number of participants and excluded groups having restrictions on languages, locations, or topics (details in Section 3.4).

The second bias is the self-selection bias, which is caused by us not being able to collect answers from non-respondents. That is normally because they have mild or no interest in the survey topic (Duda and Nobile, 2010), and hence, are not willing to join the survey. People who responded to our survey invitation, by contrast, were likely to be more interested in the survey topic. The potential problem here is that people without a vested interest in the topic might have different (or even opposite) opinions than those having an interest in the topic. Since we could only send different survey links to different LinkedIn groups, we could not identify who did not respond to our survey within each group. Therefore, sending a reminder to the non-respondents was not possible in our case.

6.2 Validity threats

Regarding the survey validity, we refer to Kitchenham and Pfleeger's guideline for personal opinion surveys (Kitchenham and Pfleeger, 2008) to discuss the three types of validity below.

6.2.1 Content validity

Content validity concerns how appropriate the survey instrument is. The assessment is typically done by a group of experts with relevant knowledge. The group ideally includes subject domain experts and members of the target population. In our case, the survey instrument was developed and reviewed through iterations of discussion among the co-authors who have acquired relevant knowledge of the subject. The instrument was also reviewed by practitioners of the target group in the pilot study.

6.2.2 Construct validity

Construct validity concerns how well the survey instrument measures what it is designed to measure. The major threat to our study is whether the respondents shared the same understanding of the context dimensions and context values as we did. For example, Back-to-back testing, and Keyword-driven testing were some of the context values that respondents could select to describe their testing practices. We assumed that the practitioners and we share the same understanding of these context values. Nevertheless, this assumption might not always hold. As pointed out by some respondents, they were not confident that they understood the information the same way we did. We were aware of this potential threat but had to consider the trade-off between the survey length and the clarification of the context information.

To mitigate this threat, we used only peer-reviewed taxonomies for providing the context dimensions and context values. Furthermore, we provided free text fields so that the respondents could express their doubts or different opinions. Nevertheless, some of our respondents mentioned that our survey was quite complex and took them quite a lot of time to complete. Thus, we consider that our mitigation strategy has not fully addressed this threat.

Another threat to the construct validity is that we required each respondent to select only one context value under each context dimension. This restriction was to support the analysis of the potential correlation between the quality attributes and the context dimensions (RQ2). In practice, it is likely that respondents work in different contexts in parallel. Hence, the restriction requires the respondents to be mindful and consistent when answering the questions relating to the importance of the quality attributes and their challenges so that their answers correspond to the selected context only. To assist the respondents in focusing on one single context, we constructed a single matrix question covering all context dimensions in a table format (a matrix question) instead of having a question for each context dimension. With this question design, the respondents could always review all their context answers in one view.

6.2.3 Criterion validity

Criterion validity is about whether the survey instrument could distinguish respondents from different groups. In our case, we used the demographics questions and the context dimensions to cluster respondents into different groups. The context dimensions were extracted from the results of our tertiary study on the same topic (Tran et al., 2021). Three context dimensions

(testing tools, test suite size, and SUT size) were removed from the analysis, and hence, one could argue that the respondents have not been well distinguished. Additionally, this grouping was done by self-assessments and opinions. The two issues could potentially introduce bias as we did not triangulate these self-assessments in our survey.

7 Conclusion and future work

The benefits of having high-quality test artifacts have been well-studied in the literature such as for improving the productivity of the development teams and for helping developers keep up with the fast development pace. However, the quality of test cases and test suites is composed of a set of roughly 30 quality attributes, according to our recent tertiary study on this topic. It is difficult to focus on all of the quality attributes or, worse, on some random attributes. Hence, to further assist practitioners in achieving high-quality test cases and test suites, it is better to recognize which quality attribute is important to practitioners but challenging for them to define, measure, and maintain. Nevertheless, there have been no studies that have answered this question yet. We, therefore, address this research gap in this industrial survey study.

We synthesized responses from 354 practitioners who got involved in various testing activities with a wide range of working experience. Our novel contribution is presented in the answers to the research questions below.

RQ1: Our findings show that, generally, most practitioners perceived the quality attributes of test cases and test suites as important rather than optional or not relevant at all. In a more fine-grained view, we found that Fault Detection, Usability, Maintainability, Reliability, and Coverage were regarded as the most important quality attributes, while Resource Efficiency and Simplicity were viewed as the least important attributes.

RQ2: The second important finding from this survey study is that the software-testing contexts practitioners were involved in did have an influence on their perception of the importance of quality attributes. More specifically, the importance of Resource Efficiency, Reusability, Simplicity, and Usability of test suites depended on several context dimensions namely Automation activity, Type of development process of SUT, Testing activity, and Application domain of SUT. Likewise, the importance of Resource Efficiency of test cases was affected by four context dimensions, namely Testing level, Testing practice, Automation activity, and Type of SUT.

RQ3: Our survey showed that practitioners still faced difficulties in the three main quality aspects, i.e., quality-attribute definition, measurement, and maintenance. Additionally, they also emphasized the difficult nature of achieving high-quality test cases and test suites.

For researchers, our findings point out where practitioners actually need further support, i.e., the quality attributes that are important but challenging for them to define, measure, and maintain. Our findings also highlight the important role of software-testing contexts in defining and assessing the quality of test cases and test suites. Therefore, future research should (i) focus on these important quality attributes and (ii) further explore contextual differences when studying test-case and test-suite quality as such quality is composed of different quality attributes whose relevance is very likely to vary depending on the contexts. For practitioners, our study encourages self-reflection. Hopefully, our study can motivate organizations to take a step back and consider providing more support to practitioners to achieve high-quality test cases and test suites. Along the same lines, we recommend organizations explicitly consider analyzing their software-testing contexts before searching for tools or different kinds of sup-

ports for assessing, improving, or maintaining the quality of test cases and test suites, which ultimately helps boost their confidence in testing.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s11219-024-09698-w.

Acknowledgements This work has been supported by ELLIIT; the Strategic Research Area within IT and Mobile Communications, funded by the Swedish Government. The work has also been supported by a research grant for the GIST (reference number 20220235) and SERT project from the Knowledge Foundation in Sweden.

Author Contributions Huynh Khanh Vi Tran, Nauman bin Ali, Michael Unterkalmsteiner, and Jürgen Börstler contributed to the study's conception and design. Material preparation and data collection were performed by Huynh Khanh Vi Tran, and the other authors discussed and reviewed all the steps. Huynh Khanh Vi Tran also led the data analysis with support from Panagiota Chatzipetrou. The first draft of the manuscript was written by Huynh Khanh Vi Tran, and all authors commented on intermediate versions of the manuscript. All authors read and approved the final manuscript.

Funding Open access funding provided by Blekinge Institute of Technology. This work has been supported by ELLIIT; the Strategic Research Area within IT and Mobile Communications, funded by the Swedish Government. The work has also been supported by a research grant for the GIST (reference number 20220235) and SERT project from the Knowledge Foundation in Sweden.

Data Availability No datasets were generated or analysed during the current study.

Materials Availability The materials used to conduct this study can be assessed via https://doi.org/10.6084/m9.figshare.23309702.v1.

Code Availability Not applicable

Declarations

Ethics Approval and Consent to Participate Before deciding to answer the survey questionnaire, the participants were informed that the survey was anonymous such that their identities were not retained (stated in the introduction of the survey instrument).

Competing nterests The authors declare no competing interests.

Consent for Publication Before deciding to answer the survey questionnaire, the participants were informed that only aggregated results would be published in a scientific publication (stated in the introduction of the survey instrument).

References

Adlemo, A., Tan, H., & Tarasov, V. (2018). Test case quality as perceived in Sweden. In: *Proceedings of the 5th International Workshop on Requirements Engineering and Testing* (pp. 9–12).

Ali, N. B., Petersen, K., & Mäntylä, M. (2012). Testing highly complex system of systems: An industrial case study. In: *Proceedings of the ACM-IEEE International Symposium on Empirical Software Engineering and Mmeasurement* (pp. 211–220).

Aljedaani, W., Peruma, A., Aljohani, A., et al. (2021). Test smell detection tools: A systematic mapping study. In: *Proceedings of the 25th International Conference on Evaluation and Assessment in Software Engineering* (pp. 170–180).

Ateşoğulları, D., & Mishra, A. (2020). Automation testing tools: A comparative view. *International Journal on Information Technologies & Security, 12*(4).

Athanasiou, D., Nugroho, A., Visser, J., et al. (2014). Test code quality and its relation to issue handling performance. *IEEE Transactions on Software Engineering, 40*(11), 1100–1125.

Baltes, S., & Ralph, P. (2022). Sampling in software engineering research: a critical review and guidelines. *Empirical Software Engineering, 27*(4), 94.

Bertolino, A. (2007). Software testing research: Achievements, challenges, dreams. In: *Proceedings of the Future of Software Engineering* (FOSE'07) (pp. 85–103). IEEE.

Bowes, D., Hall, T., Petrić, J., et al. (2017). How good are my tests? In: *Proceedings of the International Workshop on Emerging Trends in Software Metrics* (pp. 9–14). WETSoM.

Chatzipetrou, P., Papatheocharous, E., Wnuk, K., et al. (2020). Component attributes and their importance in decisions and component selection. *Software Quality Journal, 28*(2), 567–593.

Chernak, Y. (2001). Validating and improving test-case effectiveness. *IEEE Software, 18*(1), 81–86.

Cruzes, D. S., & Dyba, T. (2011). Recommended steps for thematic synthesis in software engineering. In: *2011 International Symposium on Empirical Software Engineering and Measurement* (pp. 275–284).

Daka, E., Campos, J., Fraser, G., et al. (2015). Modeling readability to improve unit tests. In: *Proceeedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering* (pp. 107–118). ESEC/FSE 2015.

de Mello, R. M., da Silva, P. C., & Travassos, G. H. (2014). Investigating probabilistic sampling approaches for large-scale surveys in software engineering. In: *Proceedings of the 17th Ibero-American Conference Software Engineering, CIBSE* (pp. 364–377).

De Oliveira Neto, F. G., Feldt, R., Erlenhov, L., et al. (2018). Visualizing test diversity to support test optimisation. In: *Proceedings of the Asia-Pacific Software Engineering Conference, APSEC* (pp. 149–158).

Duda, M. D., & Nobile, J. L. (2010). The fallacy of online surveys: No data are better than bad data. *Human Dimensions of Wildlife, 15*(1), 55–64.

Feitosa, D., Ampatzoglou, A., Avgeriou, P., et al. (2015). Investigating quality trade-offs in open source critical embedded systems. In: *Proceedings of the 11th International ACM SIGSOFT Conference on Quality of Software Architectures* (pp. 113–122). Association for Computing Machinery, New York, NY, USA, QoSA '15.

Fitzgerald, B., & Stol, K. J. (2017). Continuous software engineering: A roadmap and agenda. *Journal of Systems and Software, 123*, 176–189.

Forward, A., & Lethbridge, T. C. (2008). A taxonomy of software types to facilitate search and evidence-based software engineering. In: *Proceedings of the Conference of the Center for Advanced Studies, CASCON'08*.

Fulcini, T., Coppola, R., Ardito, L., et al. (2023). A review on tools, mechanics, benefits, and challenges of gamified software testing. *ACM Computing Surveys, 55*(14s).

García-Pérez, M. A. (2013). Statistical criteria for parallel tests: A comparison of accuracy and power. *Behavior Research Methods, 45*, 999–1010.

Garousi, V., Felderer, M., Kuhrmann, M., et al. (2017). What industry wants from academia in software testing? hearing practitioners' opinions. In: *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering* (pp. 65–69).

Garousi, V., Amannejad, Y., & Betin Can, A. (2015). Software test-code engineering: A systematic mapping. *Information and Software Technology, 58*, 123–147.

Garousi, V., Felderer, M., Kuhrmann, M., et al. (2020). Exploring the industry's challenges in software testing: An empirical study. *Journal of Software: Evolution and Process, 32*(8), e2251.

Garousi, V., & Küçük, B. (2018). Smells in software test code: A survey of knowledge in industry and academia. *Journal of Systems and Software, 138*, 52–81.

Garousi, V., Kucuk, B., & Felderer, M. (2019). What we know about smells in software test code. *IEEE Software, 36*(3), 61–73.

Goodenough, J. B., & Gerhart, S. L. (1975). Toward a theory of test data selection. *IEEE Transactions on Software Engineering, SE-1*(2), 156–173.

Grano, G., De Iaco, C., Palomba, F., et al. (2020). Pizza versus pinsa: On the perception and measurability of unit test code quality. In: *Proceedings of the 36th IEEE International Conference on Software Maintenance and Evolution* (pp. 336–347). IEEE.

Grano, G., Scalabrino, S., Gall, H. C., et al. (2018). An empirical investigation on the readability of manual and generated test cases. In: *Proceedings of the 26th Conference on Program Comprehension* (pp. 348–351). ACM, ICPC '18.

ISO/IEC JTC 1/SC 7 Software and systems engineering (2011). ISO/IEC 25010:2011 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – System and software quality models. https://www.iso.org/standard/35733.html

Jabbari, R., bin Ali N, Petersen K, et al. (2018). Towards a benefits dependency network for devops based on a systematic literature review. *Journal of Software: Evolution and Process, 30*(11), e1957.

Juhnke, K., Tichy, M., & Houdek, F. (2021). Challenges concerning test case specifications in automotive software testing: assessment of frequency and criticality. *Software Quality Journal, 29*(1), 39–100.

Kapfhammer, G. M., & Soffa, M. L. (2003). A family of test adequacy criteria for database-driven applications. *ACM SIGSOFT Software Engineering Notes, 28*(5), 98–107.

Kitchenham, B. A., & Pfleeger, S. L. (2008). Personal opinion surveys. In F. Shull, J. Singer, & D. I. K. Sjøberg (Eds.), *Guide to Advanced Empirical Software Engineering* (pp. 63–92). Springer.

Kochhar, P. S., Xia, X., & Lo, D. (2019). Practitioners' views on good software testing practices. In: *Proceeedings of the 41st International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)* (pp. 61–70). IEEE.

Kotonya, G., Sommerville, I., & Hall, S. (2003). Towards a classification model for component-based software engineering research. In: *Proceedings of the 29th EUROMICRO Conference* (pp. 43–52).

Kotrlik, J., & Higgins, C. (2001). Organizational research: Determining appropriate sample size in survey research appropriate sample size in survey research. *Information Technology, Learning, and Performance Journal, 19*(1), 43.

Kovács, A., & Szabados, K. (2014). Test software quality issues and connections to international standards. *Acta Universitatis Sapientiae, Informatica, 5*, 77–102.

Lemos, O. A. L., Vincenzi, A. M. R., Maldonado, J. C., et al. (2007). Control and data flow structural testing criteria for aspect-oriented programs. *Journal of Systems and Software, 80*(6), 862–882.

Lo, D., Nagappan, N., & Zimmermann, T. (2015). How practitioners perceive the relevance of software engineering research. In: *Proceeedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, ESEC/FSE* (pp. 415–425).

Meszaros, G. (2007). *xUnit Test Patterns: Refactoring Test Code* (1st ed.). Addison-Wesley Professional.

Neukirchen, H., Zeiss, B., & Grabowski, J. (2008). An approach to quality engineering of TTCN-3 test specifications. *International Journal on Software Tools for Technology Transfer, 10*(4), 309.

Pei, K., Cao, Y., Yang, J., et al. (2017). Deepxplore: Automated whitebox testing of deep learning systems. In: *Proceedings of the 26th ACM Symposium on Operating Systems Principles - SOSP* (pp. 1–18).

Ray, B., Posnett, D., Filkov, V., et al. (2014). A large scale study of programming languages and code quality in github. In: *Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering* (pp. 155–165).

Santos, R. E., Bener, A., Baldassarre, M. T., et al. (2019). Mind the gap: Are practitioners and researchers in software testing speaking the same language? In: *Proceeedings of the 7th International Workshop on Conducting Empirical Studies in Industry (CESI) and 6th International Workshop on Software Engineering Research and Industrial Practice (SER&IP)* (pp. 10–17).

Sas, D., & Avgeriou, P. (2020). Quality attribute trade-offs in the embedded systems industry: An exploratory case study. *Software Quality Journal, 28*(2), 505–534.

Sheskin, D. J. (2020). *Handbook of parametric and nonparametric statistical procedures*. CRC Press.

Software IJS, systems engineering (2022) *Iso/iec/ieee International Standard - Software and Systems Engineering –Software Testing –Part 1:General Concepts. ISO/IEC/IEEE 29119-1:2022(E)* (pp. 1–60).

Souza, ÉFd., Falbo, Rd. A., & Vijaykumar, N. L. (2017). ROoST: Reference ontology on software testing. *Applied Ontology, 12*(1), 59–90.

Spadini, D., Palomba, F., Zaidman, A., et al. (2018). On the relation of test smells to software code quality. In: *Proceedings of IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 1–12).

Tran, H. K. V., Ali, N. B., Börstler, J., et al. (2019). *Test-Case Quality – Understanding Practitioners' Perspectives. In: Proceedings of the 20th International Conference on Product-Focused Software Process Improvement (PROFES)* (pp. 37–52). Springer.

Tran, H. K. V., Unterkalmsteiner, M., Börstler, J., et al. (2021). Assessing test artifact quality–a tertiary study. *Information and Software Technology, 139*, 106620.

Van Deursen, A., Moonen, L., Van Den Bergh. A,, et al. (2001) Refactoring test code. In: *Proceedings of the 2nd International Conference on Extreme Programming and Flexible Processes in Software Engineering (XP)* (pp. 92–95).

Vijayasarathy, L. R., & Butler, C. W. (2016). Choice of software development methodologies: Do organizational, project, and team characteristics matter? *IEEE Software, 33*(5), 86–94.
Wahler, M., Eidenbenz, R., Monot, A., et al. (2017). Quality attribute trade-offs in industrial software systems. In: *Proceedings of the IEEE International Conference on Software Architecture Workshops (ICSAW)* (pp. 251–254).
Waseem, M., Liang, P., Shahin, M., et al. (2021). Design, monitoring, and testing of microservices systems: The practitioners' perspective. *Journal of Systems and Software, 182*, 111061.
Zaidman, A., Van Rompaey, B., Demeyer, S., et al. (2008). Mining software repositories to study co-evolution of production & test code. In: *Proceedings of the 1st International Conference on Software Testing, Verification and Validation, ICST 2008* (pp. 220–229).
Zeiss, B., Vega, D., Schieferdecker, I., et al. (2007). Applying the ISO 9126 quality model to test specifications–exemplified for TTCN-3 test specifications. In: *Software Engineering 2007–Fachtagung des GI-Fachbereichs Softwaretechnik*
Zhu, H., Hall, P. A., & May, J. H. (1997). Software unit test coverage and adequacy. *ACM Computing Surveys, 29*(4), 366–427.

Huynh Khanh Vi Tran is a Ph.D. candidate at the Blekinge Institute of Technology. Her research focuses on the quality of test artifacts (test case, test suite, test script, test code, test specification) in software engineering, particularly investigating how to improve such quality.

Nauman bin Ali is a senior lecturer at Blekinge Institute of Technology. He is involved in empirical research in the field of software engineering. His research interests include using AI and ML for software engineering, lean software development, software testing, software process simulation, software metrics, and evidence-based software engineering. He received his Ph.D. (2015) in software engineering from Blekinge Institute of Technology, Sweden.

Michael Unterkalmsteiner has been researching Software Engineering since 2009, focusing in particular on the coordination between requirements engineering and software testing. His research work with many industry partners is shaped by empirical problem identification, in-depth analysis of the state-of-art and practice, and collaborative solution development. His current research focuses on designing and implementing automated decision support systems for engineers, using natural language processing and machine learning technologies. Michael is a senior lecturer at the Blekinge Institute of Technology.

Jürgen Börstler is a professor of software engineering at Blekinge Institute of Technology (BTH), Sweden. He received a Ph.D. in computer science from Aachen University of Technology (RWTH), Germany. Prof. Börstler is a member of SERL-Sweden, the Software Engineering Research and Education Lab at BTH and a senior member of the Swedish Requirements Engineering Network. Furthermore, he is a founding member of the Scandinavian Pedagogy of Programming Network. His research interests are in behavioral software engineering, research methods, software process improvement, software quality, program comprehension, and computer science education.

Dr. Panagiota Chatzipetrou is an Associate Professor (Docent) at the Department of Informatics at Örebro University in Örebro, Sweden. As a researcher, she mainly focuses on empirical studies under the different perspectives of software development. Her research interests include applications of statistical methods to quality problems in software engineering and especially to requirements engineering. Currently, she aims to explore how large language models (LLMs) can be used for Requirements analysis processes and works towards the different uses of AI in requirements engineering. Furthermore, she works with behavioral software engineering and the exploitation of the human factor and the different views that ultimately determine the quality of software product development. Moreover, she has been working with decision support systems for the development of software-intensive systems and large-scale agile (and global) software development. She has collaborated with numerous higher education institutions in Sweden, Greece, and Great Britain.

Authors and Affiliations

Huynh Khanh Vi Tran[1] · Nauman bin Ali[1] · Michael Unterkalmsteiner[1] · Jürgen Börstler[1] · Panagiota Chatzipetrou[2]

✉ Huynh Khanh Vi Tran
huynh.khanh.vi.tran@bth.se

Nauman bin Ali
nauman.ali@bth.se

Michael Unterkalmsteiner
michael.unterkalmsteiner@bth.se

Jürgen Börstler
jurgen.borstler@bth.se

Panagiota Chatzipetrou
panagiota.chatzipetrou@oru.se

[1] Department of Software Engineering, Blekinge Institute of Technology, Valhallavägen 1, Karlskrona 371 41, Blekinge, Sweden

[2] Department of Informatics, Centre for Empirical Research on Information Systems (CERIS), School of Business, Örebro University, Fakultetsgatan 1, Örebro 701 82, Örebro, Sweden

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH ("Springer Nature").

Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users ("Users"), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use ("Terms"). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.

These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.

We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.

While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.

These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.

To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.

Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.

If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com