CHALMERS
UNIVERSITY OF TECHNOLOGY

How Do Different Types of Testing Goals Affect Test Case Design?

How Do Different Types of Testing Goals Affect Test Case Design? *

Dia Istanbuly, Max Zimmer, and Gregory Gay[0000−0001−6794−9585]

Chalmers | University of Gothenburg, Gothenburg, Sweden
(gusistdi, gusmaxzi)@student.gu.se, greg@greggay.com

Abstract. Test cases are designed in service of goals, e.g., functional correctness or performance. Unfortunately, we lack a clear understanding of how specific goal types influence test design. In this study, we explore this relationship through interviews and a survey with software developers, with a focus on identification and importance of goal types, quantitative relations between goals and tests, and personal, organizational, methodological, and technological factors.
We identify nine goal types and their importance, and perform further analysis of three—correctness, reliability, and quality. We observe that test design for correctness forms a "default" design process that is modified when pursuing other goals. For the examined goal types, test cases tend to be simple, with many tests targeting a single goal and each test focusing on 1–2 goals at a time. We observe differences in testing practices, tools, and targeted system types between goal types. In addition, we observe that test design can be influenced by organization, process, and team makeup. This study provides a foundation for future research on test design and testing goals.

Keywords: Software Testing, Test Design, Testing Goals, Functional Testing, Non-Functional Testing

1 Introduction

Software testing is a process where input is applied to a system-under-test (SUT) and observations of the reaction to the input are used to verify that the SUT operates correctly [19]. It is the most common verification technique, and can be conducted in many forms and at different levels of granularity within the code of the SUT [4]. Testers may write test cases before or after writing the code-under-test [20], and according to different personal problem-solving models [9, 12].

Unifying all testing approaches, practices, and technologies is that tests are designed in service of *goals*. These goals can vary in nature and type—for example, a test might be written to verify functional requirements, to show that known security risks are mitigated, or to ensure that performance thresholds are met [17]. The nature of problem-solving implies that developers must have goals

* Support provided by Software Center Project 30: "Aspects of Automated Testing".

that they wish to achieve through the act of designing tests, even if those goals are not explicitly enumerated in requirements or other documentation [9, 18].

Despite the prominence of testing as a development practice, we lack a clear understanding of how specific types of testing goals influence the practice of test design. For example, Enoiu et al. proposed a model of test case design as a problem solving process [9]. This model makes it clear that tests are designed to show the attainment of specific goals, but does not discuss what common types of goals are, or how different goal types could influence this process.

The purpose of this research is to explore the types of goals that testers pursue and the influence of these goal types on the process that developers follow to design tests that assess attainment of those goals. Understanding the relationship between test case design and different types of testing goals could provide benefits to both researchers and practitioners. For example, such understanding enables characterization of test design practices and the ability to offer clear guidance to developers creating tests for specific types of goals—potentially leading to improved effectiveness or efficiency of the testing process. In addition, characterization of design practices can benefit automated test generation, potentially leading to the development of more human-like generation tools [8, 11].

In particular, we are interested in the exploring the types of goals pursued, the relative importance of different goal types, quantitative relations between goal types and test cases—tests-per-goal and goals-per-test—and personal, organizational, methodological, and technological factors that may influence the relationship between goal types and the test design process. To address these topics, we have conducted a series of interviews with software developers in various domains and of varying experience. Thematic analysis of the interviews was then used to develop a survey for wider distribution.

Based on analyses of the interview and survey responses, we have identified nine goal types, including correctness, reliability, performance, quality, security, customer satisfaction, risk management, improving maintenance cost, and process improvement. Correctness was ranked most important, followed by reliability and security. Customer satisfaction and maintenance cost were seen as least important, but were still valued.

We focused on correctness, reliability, and quality for analysis. Test design for correctness forms a "default" design process, which is followed in a modified form for other goals. For all three goal types, several tests are needed to assess goal attainment, and tests focus on 1–2 goals at a time. Testers often start the design process following pre-existing patterns (e.g., using past tests as templates). We also make observations regarding differences in testing practices and tools employed during test design for these three goal types. We further observe that test design can be influenced by process, organization, and team structure.

This study provides a foundation for future research on test case design and testing goals, and enables deeper modeling of test design as a cognitive problem-solving process. To help enable future research, we also make our thematic interview coding and survey responses available[1].

[1] Available at https://doi.org/10.5281/zenodo.8106998.

2 Background

During software testing, input is applied to the SUT and the resulting output and other observations are captured [19]. These observations are compared to embedded expectations, called test oracles, to determine whether the SUT operated within expectations. Oracles often directly reflect the goals of test creation [2].

Testing can take place at multiple levels of granularity [19]. At the lowest level, unit tests examine small code elements in isolation with dependencies substituted for "mock" (static) results [21]. During integration testing, dependent units are combined. Then, during system testing, high-level functionality is invoked through interfaces. Tests can be written at all three levels as executable code, using frameworks such as JUnit, PyTest, or Postman [21].

Testing can also be performed manually. This is common during exploratory testing—where humans perform ad-hoc testing based on "tours" [24]—and acceptance testing—where customers offer feedback [22]. Tests are often written after code has been developed. However, test-driven development advocates for test creation before code development [14].

3 Related Work

Enoiu et al. hypothesized that our understanding of test design practices could be improved by formulating test design as a cognitive problem solving model [9]. Their exploratory work proposes that testers follow a seven stage process, including identification and definition of testing goals, knowledge analysis, strategy planning, information and resource organization, progress monitoring, and evaluation. Their research provides inspiration. In particular, we focus on the goal aspect of this model—filling in gaps in our understanding of the goals that testers focus on and how those goals influence test design. Our findings complement, and could lead to elaborations, in this model—as well as in other research that examines testing as a cognitive process, e.g., Aniche et al.'s framework for how developers approach test design [1] or Hale et al.'s cognitive model of how developers choose, implement, and evaluate debugging rules and strategies [12, 13].

There has been limited research on how developers approach test case design in practice [9]. For example, Garousi et al. surveyed software developers to examine techniques, tools, methods, and metrics used in test design [10]. They observed that the usage of JUnit and IBM Rational testing tools have been overtaken by NUnit and web application tools. They also observed that organizations were still slow to adopt test-driven development. Eldh et al. asked experts to review 500 test cases written by novice testers [7]. They examined whether the tests matched the IEEE test case template, and also compared the mistakes made by the novices to those made by expert testers. Some mistakes were made by both groups, but others—such as not cleaning up after test execution—were made far more often by novices. Beer and Ramler also investigate how testers' experience impacts the effectiveness of testing methods and tools [3]. They briefly examine test design, noting that experience plays a large role in determining the effectiveness of the designed tests. Although all three studies examine aspects of test design, they—and other past studies—do not closely examine the influence of different goals on test design. Our study contributes to filling this gap.

Table 1: Demographic information on interview participants.

ID	Role	Development Experience
P1	Software Developer	7.0 Years
P2	Senior Consultant, DevOps Architect	13.0 Years
P3	Software Developer	0.5 Years
P4	Software Developer	9.0 Years
P5	Software Developer	4.0 Years
P6	Software Developer	3.0 Years
P7	Test Manager, Scrum Master	10.5 Years
P8	Quality Manager	6.0 Years
P9	Software Developer	2.0 Years

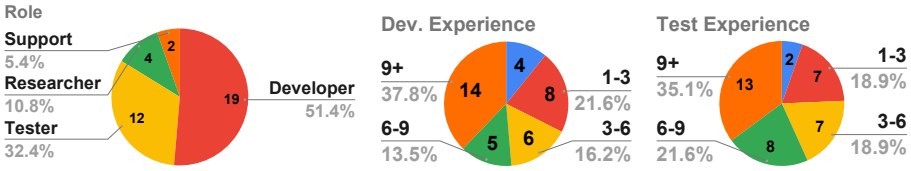


Fig. 1: Demographic information on survey participants.

4 Methodology

We hypothesize that developers follow distinct patterns when designing tests in relation to specific types of testing goals. To examine this hypothesis, we address the following research questions:

- **RQ1**: What types of goals do developers attempt to achieve when designing test cases, and how do they perceive the importance of these goal types?
- **RQ2**: What approaches are taken to design tests for these goal types?
  - **RQ2.1**: How many test cases are generally required to achieve a goal of a specific type?
  - **RQ2.2**: How many goals of each type are generally targeted by a test?
  - **RQ2.3**: Are there patterns that explain the relationship between the selected goal type and the resulting test design process?

To address these questions, we performed semi-structured interviews to gain insight into the test design process. We then performed a thematic analysis of the interview findings, and designed a follow-up survey for wide distribution to provide further evidence for the findings from the interview study, as well as additional quantitative and qualitative data to help answer our research questions.

4.1 Population and Sampling

We performed a series of nine interviews—selected through convenience sampling—with software developers in Gothenburg, Sweden. 37 unique respondents, from around the world, filled out the survey[2]. Table 1 outlines demographic information on the interview participants, including their role and development experience. Figure 1 summarizes the same information for the survey participants, with

[2] One survey response was discarded, as a respondent answered twice. We retained the first response from this participant.

Table 2: Interview questions, linked to research questions

RQ	Interview Question
N/A	1. What is your professional role? 2. How much experience do you have in software development? 3. Do you write test cases for code? 4. How much experience do you have in software testing?
RQ1	5. Can you describe the goals that you target through the creation of test cases? 6. What are you trying to achieve, avoid, or discover when creating test cases?
RQ2	7. What methods do you use to test your code? 8. Describe the process you follow to design test cases. 9. What tools do you use to create test cases? 10. Do you design test cases by yourself or in a group? 11. How do you perform test design in order to achieve your goals? 12. How have your testing methods changed over time?
RQ2.1	13. How many test cases do you tend to design for each goal you want to achieve?
RQ2.2	14. Do you design test cases to achieve one or multiple goals at once? 15. How many goals do you target with each test case?
RQ2.3	16. Do you have any recurring test case design habits? 17. Do you tend to design test cases in the same way as long as it works, or do you regularly search for new tools and approaches when designing test cases? 18. How do you assess success or failure at achieving a goal? 19. In case of failure to achieve a goal, what do you do to address problems? 20. If you have achieved a goal, how do you proceed?

Table 3: Survey questions, linked to research questions. MC = multiple choice.

RQ	Survey Question	Format
N/A	1. What is your profession?	Free Text
	2. How many years of experience do you have with development?	MC
	3. How many years of experience do you have with software testing?	MC
RQ1	4. Do you have specific goals or types of goals when designing tests?	MC
	5. Please rank the following types of goals by ascending importance.	MC (Grid)
(Q6–12 repeated for two goal types indicated as most important)		
RQ2	6. What testing techniques do you use for this goal?	MC, Free Text
	7. Please outline your typical design process when designing a test case to achieve the selected goal type.	Free Text
	8. What tools do you use to design tests for the selected goal type?	MC, Free Text
	9. What type of system do you design tests of this goal type for?	MC, Free Text
RQ2.1	10. How many tests do you typically design for a goal of this type?	MC
RQ2.2	11. How many goals of this type do you try to cover with a single test?	MC
RQ2.3	12. When designing test cases for this goal type, how often do you tend to re-use a particular test case design pattern?	MC, Free Text
	13. Do you design test cases by yourself or in a team?	MC
	14.Are there standards or pre-defined methods used within your organization when designing test cases or goals?	MC, Free Text
	15. How do development and process methodologies (e.g. Scrum/DevOps) influence the way you write tests to achieve different goal types?	Free Text

the addition of testing experience[3]. The interview and survey participants work in many domains—e.g., automotive, data analytics, and telecommunications.

[3] Job titles have been merged when similar, e.g., "software tester" and "test engineer". The survey asked for both development and testing experience, while the interview only asked about years of development experience.

4.2 Data Collection

Interviews: Interviews were conducted electronically between January–February 2022. At the start of an interview, we gave a brief overview of the research topic. Interview responses were recorded, with permission, for analysis and observer triangulation. The interviews were semi-structured, following the interview guide in Table 2. However, follow-up questions were asked if further discussion was needed. The interviews were transcribed using speech-to-text software, then manually corrected through consultation with the original audio.
Survey: The survey consisted of mostly quantitative questions, with a small number of open-ended questions, and was designed according to the guidelines of Linåker et al. [15]. The questions are listed in Table 3. We focused on quantitative questions to decrease the time burden [16]. To complete the survey, the participants were required to answer all multiple-choice questions, but open-ended questions were optional. The survey was pre-tested with two participants, and the feedback was used to clarify wording and question order. The survey was conducted using Google Forms[4]. Links for the survey were then distributed via email, as well as on social media platforms.

4.3 Data Analysis

Thematic Analysis: The interview transcripts were analyzed using thematic analysis, following the guidelines of Cruzes et al. [6] and Braun et al. [5]. To conduct this process, we independently examined transcripts, highlighting aspects relevant to the research questions ("codes"). The codes were subsequently organized and aggregated into sub-themes, which were clustered into themes.

After individual coding was completed for the first interview, we compared the codes. After the first iteration, our codes did not achieve an 80% similarity threshold. Therefore, we came to an agreement on sub-theme identification and code classification. After a second iteration, a similarity of over 80% was achieved. This process was conducted iteratively, and was paused for discussion.

An overview of the themes and sub-themes is shown in Table 4. These themes and sub-themes, as well as the underlying codes, were used both to directly assist in addressing the research questions as well as to design the survey instrument.
Survey Analysis: The survey consisted of both quantitative and open-ended qualitative questions. To analyze quantitative data, we used summary statistics (e.g., mean and variance of responses, separated by goal type) [23]. Responses to open-ended questions were assessed using thematic coding. As the survey was designed using the interview codes and themes, no additional themes or sub-themes were identified. Instead, survey responses enriched the existing codes.

5 Results and Discussion

5.1 Goals and Goal Importance

Test design can not take place without *some* reason to design tests in the first place. During the interviews, we identified nine specific types of goals that testers

[4] https://forms.gle/bhzpUCX9PdXbebiH8

Table 4: Themes (bold) and sub-themes from interviews.

Theme	**Explanation**
Experience	Interviewees' experiences in development and testing.
Profession	Interviewees' job titles and responsibilities.
Test Case Design	Encompasses sub-themes related to test design planning and execution.
Design Process	Steps that interviewees take when designing tests to achieve their goals.
Alone/In Group	Whether interviewees work individually or in a group during test design.
Design Plan	Specific practices interviewees apply while test design.
Design-Goal Relation	Factors that relate goals to test design (e.g., the number of tests to achieve a specific type of goal).
Recurring Habits	Common practices performed while designing test cases, (e.g., basing new tests on earlier tests).
Testing Goals	Specific goal types that interviewees design tests to achieve (e.g., functional correctness or performance).
Measuring Success	How interviewees determine whether goals are achieved (e.g., code coverage, customer satisfaction).
In Case of Failure	Steps taken when goals are not met (e.g., fault analysis).
In Case of Success	Steps taken when tests show goals are achieved (e.g. performing a demo to the client).
Testing Tools	Tools and technologies used to plan, design and execute tests (e.g., JUnit).
Testing Methods	Testing methods or practices (e.g., Test Driven Development) used by the interviewees to achieve their goals.
Change Over Time	How interviewees evolved their testing practices following experiences, mistakes, and new technologies or practices.
System Type	The type of system tested (e.g., API end-points, embedded).

Table 5: Goal types identified in interviews.

Goal Type	Definition
Correctness	Tests assess SUT behavioral consistency with specifications.
Reliability	Tests assess the ability of the SUT to remain available and failure-free in a specified environment over a period of time.
Performance	Tests assess the ability of the SUT to meet performance goals (e.g., response time).
Quality	Tests assess whether the SUT meets specified quality goals (e.g., usability).
Security	Tests assess whether the SUT can protect data and services from unauthorized access.
Customer Satisfaction	Tests assess whether the SUT meets the needs of a customer.
Risk Management	Tests are used to forecast and evaluate threats and their possible impact on the SUT.
Maintenance Cost	Tests are used to make SUT maintenance more efficient.
Process Improvement	Tests are created as part of an attempt to improve the testing process (e.g., writing automated test code to replace manual testing).

pursue. These types are defined in Table 5. 76% of survey respondents confirmed that they have specified, pre-determined goals when designing test cases.

RQ1 (Goals): The goal types identified include correctness, reliability, performance, quality, security, customer satisfaction, risk management, improving maintenance cost, and process improvement.

Survey participants were asked to rank these goal types in importance. The results of this ranking are shown in Figure 2, along with their average ranking. As might be expected, correctness was ranked as the most important goal (59%).

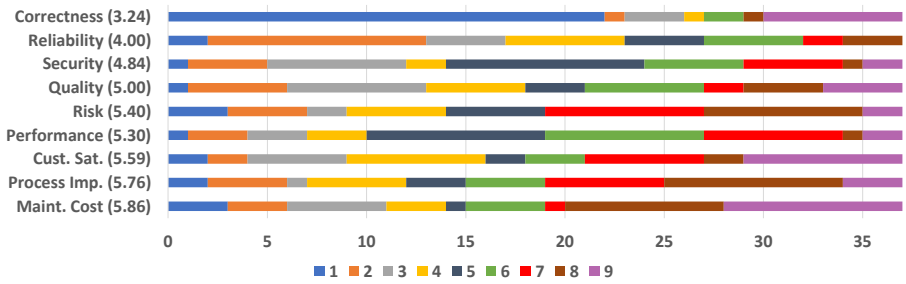


Fig. 2: Ranking of goal importance (1 = most important, 9 = least important). Average ranking indicated in parentheses.

Interestingly, seven participants indicated correctness as their *least* important goal. At least three of these work with machine learning, where non-determinism often makes assessing correctness difficult. This could be a reason for the low ranking. Among the seven, risk management, performance, customer satisfaction, and maintenance cost reduction ranked highly.

Correctness was followed by reliability. These are followed by assessment of non-functional qualities—security, quality, and performance—and risk management. These goals are important, but are often secondary considerations. In some cases, the participants were highly split in their rankings—e.g., for security, quality, and risk—with a near-even split between high and low importance.

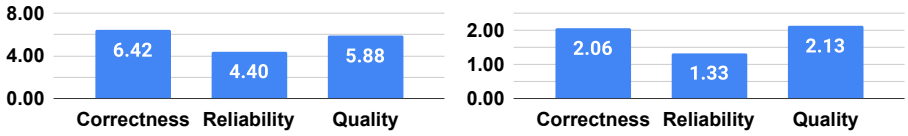
Customer satisfaction and improving process or maintenance are seen, generally, as the least important goals. However, all goal types were highly important among a subset of participants. Some differences could be explained by the nature of the organization that the tester works for. If products are created for clients, then pleasing those clients is—naturally—a high priority. If the organization sells the product widely, then individuals do not have to give approval.

RQ1 (Goals): Correctness was ranked most important, followed by reliability. Customer satisfaction and maintenance cost were seen as the least important. However, all goal types are important for some.

In the survey, we asked participants to answer a set of questions for two chosen goal types. To avoid drawing biased conclusions, we primarily focus in the following subsections on goal types that we received at least five responses for—**correctness** (33 responses), **reliability** (12 responses), and **quality** (8 responses)—or on observations not dependent on specific goal types.

5.2 Quantitative Relationship Between Goal Types and Tests

Test Cases Per Goal: Figure 3(a) shows the average number of tests needed to achieve a goal. Correctness requires the most tests per goal, on average (6.42). Advice on test design often advocates for creating multiple tests for functions that, individually, are simple and focused on a single outcome or facet of the



(a) Avg. num. tests to achieve a goal. (b) Avg. num. goals targeted in a test.

Fig. 3: Quantitative relationship between tests and goals of correctness, reliability, and quality types.

tested function. For example, a common (and highly debated[5]) recommendation is to use a single assertion per test case. Multiple interviewees echoed this advice. Figure 3(a) indicates that this advice is followed, and that multiple tests are needed to demonstrate that a complex function fulfills its specification.

Reliability goals, which typically take different measurements (e.g., failure rate or availability) and compare them to thresholds, require fewer test cases to assess. However, multiple tests are still required to assess a single goal.

RQ2.1 (Tests Per Goal): Several tests are needed to assess whether a single correctness, reliability, or quality goal is met. Correctness requires the most, an average of 6.42 tests per goal.

Goals Per Test Case: Figure 3(b) shows the average number of goals targeted in a single test—on average, approximately two correctness or quality goals, or one reliability goal. This offers further indication that testers tend to focus on creating focused tests over large tests that target many goals at once.

RQ2.2 (Goals Per Test): A single test case tends to focus on 1–2 correctness, reliability, or quality goals.

5.3 Influence of Goals on Test Design

In this subsection, we will explore multiple factors that affect and illustrate the relationship between testing goals and test design practices.

"Typical" Design Process: Survey participants were asked to outline their typical test design process for their selected goal types. Test design for correctness often starts with examination of documentation and discussion with stakeholders. The gathered knowledge is then processed during brainstorming:

> "Identify basic tests to use as foundation..., discussions with others ..., questioning, setting up environments with test data, reviewing tests" - SP10
>
> "I design a simulation of that components usage, and compare with the hand-written solution (done in my head)." - SP3
>
> "I clarify requirements, do mindmaps and discuss with different oracles..." - SP12

[5] E.g., https://softwareengineering.stackexchange.com/questions/394557/should-tests-perform-a-single-assertion-or-are-multiple-related-assertions-acce

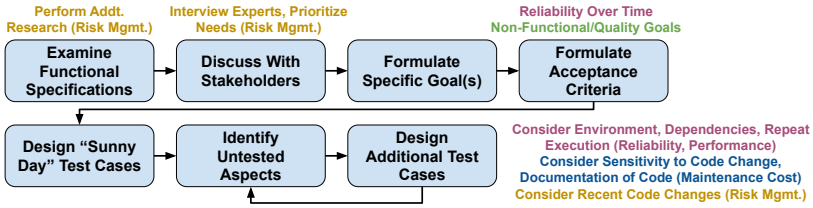


Fig. 4: Typical design process for correctness, with modifications for other goals.

Many respondents start by designing basic "happy path" tests for a function showing that the standard outcomes of the function are met. Then—often iteratively—tests are designed to cover additional scenarios:

> "Write tests, write minimal code to make tests pass, examine if there are features or corner cases that don't have tests yet, go to step one." - SP13

Respondents stressed the importance of simplicity:

> "I try to isolate requirements and design as simple a test case as possible ... I focus on making the test easy to understand, partly by making it small and independent ... I may write multiple test cases for one requirement because there may be multiple modes of failure." - SP4

There was emphasis on considering perspectives, tools, and environments:

> "... cover all aspects of the test, such as—bare minimum—up to maximum range of values, different user types if they exist, positive/negative aspects, etc." - SP30
> "1. try to understand the functionality specification 2. try to understand the test environment needed 3. trying to understand what tools are needed 4. trying to understand acceptance criteria 5. create test steps" - SP19

We asked interviewees to describe their typical test design process. Although specific goal types were not considered at that time, their comments largely echo the process outlined above for correctness and illustrated in Figure 4. Testers collect information, brainstorm, then iteratively create test cases until all functional outcomes are covered—focusing on individually simple and understandable tests.

Enoiu et al. proposed that testers follow a seven stage process—identification and definition of testing goals, knowledge analysis, strategy planning, information and resource organization, progress monitoring, and evaluation [9]. Our observations suggest that this model is largely followed when testers pursue correctness goals. Some aspects of this process may be given more or less weight at times—or even skipped entirely—depending on the form of testing or due to personal experience and preferences. For example, during unit testing, there may not be active discussions with stakeholders. However, this basic process offers a basic outline for discussions on test design.

The responses written about other goal types suggest that the process in Figure 4 is followed—in a modified form—when pursuing the other goals. For example, the core differences for reliability-demonstrating tests are that (a) reliability must be measured over a period of time, and (b), the SUT should be tested in a realistic environment—which may contain unreliable dependencies. For both reliability and performance, it was also suggested that tests must execute operations multiple times:

> "... stress test both that failing dependencies are correctly handled and that failures don't happen too often over a longer time of nominal operation." - SP4

For quality goals, tests examine both functional correctness and attainment of non-functional properties:

> "... to improve the quality, there could be some overlap of functional and non functional requirements testing here." - SP7

To reduce maintenance costs, testers should design tests that are sensitive to inadvertent code change and use tests to document the code:

> "... ensure that the changes to the interface are documented through tests, with the goal of making tests that would be sensitive to inadvertent changes ..." - SP21

Risk management often requires interviews and research, prioritization, and understanding of recent code changes:

> "I ask subject matter experts about the software under test, do some research, then try to write tests that will help mitigate risk." - SP33
> "I take input from all stakeholders to assess what is most important ... Also talk to developers to assess complexity of code that might have been mostly impacted by the change ..." - SP12

RQ2.3 (Relationship Factors): Test design for correctness goals starts with knowledge gathering and brainstorming, then design of "happy path" tests, then design of tests covering alternative outcomes. When assessing reliability or performance, tests should utilize a realistic environment, assess behavior over time, and be executed multiple times. Tests for quality blend functional and non-functional aspects. Tests can reduce maintenance costs by documenting and detecting changes. Risk management requires research, discussion, and awareness of change.

Use of Recurring Design Patterns: Survey participants were asked if they follow recurring patterns when designing tests for different types of goals. This could include, for example, using past test cases as templates or following specific structures for writing test cases for that goal. Figure 5 indicates that the majority of respondents use such patterns as a starting point:

> "... I overwhelmingly look for similar tests that I can adapt ... initially, then use it to get to the happy path. Following that I'll typically duplicate the test and ensure that some critical conditionals are covered. If it feels like there isn't a test case I can steal, or that setup is too onerous, then I will manually repeat the happy path process until it works before considering adding a test case or two." - SP21

RQ2.3 (Relationship Factors): Testers often start test design following patterns. For quality goals, testers often deviate from these patterns.

System Type: The type of system may influence the goal types pursued and their importance. In Figure 6, we indicate the percentage of respondents who

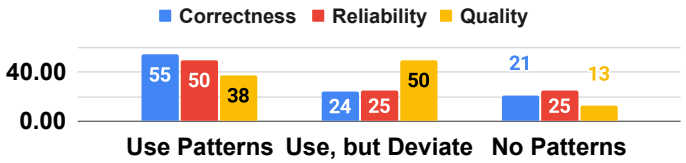


Fig. 5: % of participants re-using patterns when designing tests for correctness, reliability, and quality goals.

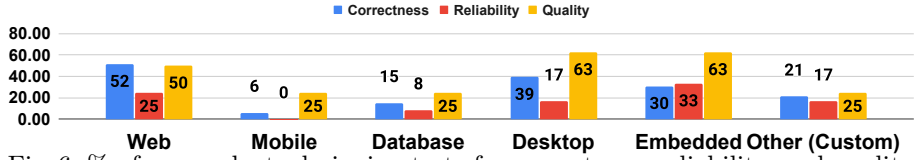


Fig. 6: % of respondents designing tests for correctness, reliability, and quality goals for different system types.

target various system types. We observe a potential relationship between reliability's importance and the level of required trust in a SUT.

Embedded systems make up the largest proportion of reliability responses. Such systems have high safety demands, and testers may need to show evidence of correctness, reliability, and quality. No respondents indicated that they develop reliability tests for mobile applications. This may be due to the *lack* of criticality in such programs. This observation should be further explored in future work.

RQ2.3 (Relationship Factors): Demonstrating reliability is a focus for embedded systems. Reliability may not be important for mobile apps.

Practices and Tools: The testing practices employed—as well as the tools—may differ between goal types. Figure 7 indicates the percentage of respondents who employed different approaches, including levels of granularity (e.g., unit testing), focuses (e.g., functional versus non-functional testing), and other practices (e.g., mocking, test-driven development). Figure 8 does the same for different types of tools. For both, the initial set of options were derived from interviews. However, some respondents suggested additional options. "Automated Testing Framework" includes frameworks where tests are written as code. The most common responses included JUnit, PyTest, and Google Test.

All three goal types are pursued at all major levels of granularity. However, reliability is relatively uncommon when using human-driven practices—i.e., acceptance, exploratory, and manual testing. Reliability is often demonstrated by executing tests repeatedly or over a period of time [19]. This typically requires automation. In addition, reliability is often attained before presenting the SUT to a client, reducing the importance of acceptance testing. Test design for reliability also typically does not seem to use mocking or test-driven development, perhaps because reliability is most meaningful for a near-final product.

In contrast, quality is often a focus of GUI and human-driven practices. Some typical quality types, such as usability, rely on replicating the typical user experience. This may lead to prominent use of human-driven practices. Correctness,

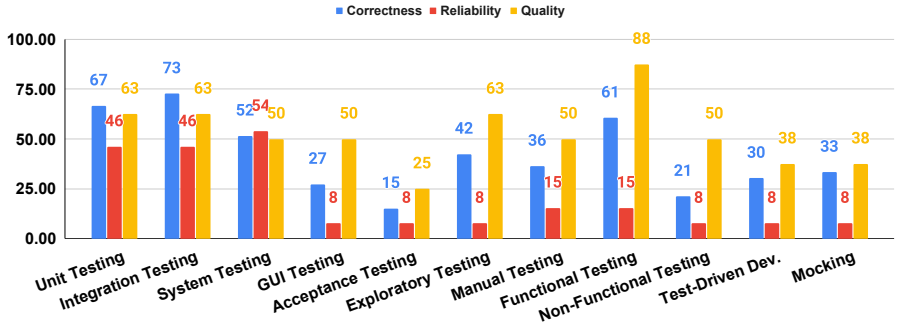


Fig. 7: % of respondents performing testing practices when designing tests for correctness, reliability, and quality goals.

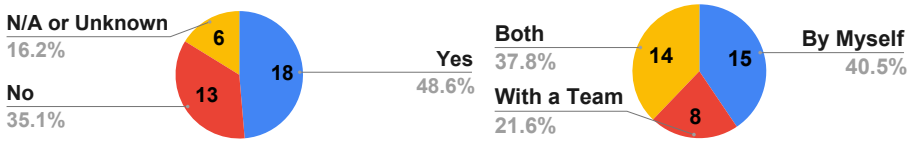
Fig. 8: % of respondents using different tools when designing tests for correctness, reliability, and quality goals.

as the "default" goal of testing, necessitates use of almost all practices, with the exception of acceptance testing. Acceptance testing is generally conducted at a late stage of development—when developers have a product to demonstrate [22]. At this stage, correctness testing may have largely concluded.

Automated testing frameworks, CI/CD pipelines, API frameworks, and command-line scripting are used for all three goal types. Property-based testing tools, which generate random input to violate properties, are used to assess correctness or reliability, but not quality goals. Test management tools, such as Jira and its Xray plug-in, are used for correctness and quality. However, they are less useful for reliability, which depends primarily on executing tests on demand.

RQ2.3 (Relationship Factors): Reliability is often pursued using code-focused, automated practices on a near-final product. Quality is often pursued using human and GUI-focused practices. Acceptance testing is rare for correctness and reliability, but more common for quality. Automated and API testing frameworks, CI/CD pipelines, and command-line scripting are common for all goal types.

Organizational Factors: Organizational policies, process, and team composition could also influence test design, regardless of the goal type. We asked



(a) Whether participants must follow company or legal policies during test design.

(b) Whether participants design tests alone or in teams.

Fig. 9: Organizational factors that may affect test design.

survey participants whether there are test design constraints enforced by their organization or legal regulations. Figure 9(a) indicates that constraints affect the majority (48.60%) percent of respondents.

We also asked respondents whether they design tests alone, in a team, or both, as collaborative design may lead to different tests than design by a single tester. Figure 9(b) indicates that a slight plurality (40.50%) work along, but that many work in team settings at all (21.60%) or some (37.80%) times. Interviewees suggested that the need for teamwork increases with project complexity:

> "If design strategies are needed, teamwork is mandatory." - P4

In a group setting, test design is also often led by test leads with assistance from others working under them:

> "Test leads can do a pretty good job there. So, trust your test leads." - P2

Survey participants were also asked about the influence of development processes on test design. Many felt that there was no influence—other than the positive increase in the use of CI tools and DevOps—or even that testers are the ones that influence development practices:

> "Most often we influence them... We are also the gateway that demands documentation." - SP19

Others discussed positive and negative aspects of short development cycles:

> "Short cycle times encouraged by an agile-like methodology do make it difficult sometimes to add test cases, but some of our PRs tend to be quite self contained (i.e. the change is proposed with several test cases)." - SP21

Multiple respondents indicated that agile processes can be beneficial for getting feedback and offering structure:

> "It helps me to be structured. Keep track of things, like if all the functionalities are covered by the test suits are not." - SP7

Respondents also warned that rigid enforcement of practices can waste testing resources that could otherwise be devoted to more productive goals:

> "large suite that must pass ... makes the teams 'waste' time when making sure that as many as possible checks will pass when pushing to master." - SP23

RQ2.3 (Relationship Factors): Test design can be influenced by process, organization, and team structure. Many testers are constrained by organization or regulatory policies. Testers perform design individually somewhat more often, but also often work in teams. Testers feel they can influence development methodologies, and that agile processes offer feedback and structure. However, there are positive and negative aspects of short release cycles, and rigid practice enforcement can waste time.

6 Threats to Validity

Conclusion Validity: The number of responses may affect conclusion reliability. However, our thematic findings reached saturation within nine interviews, and the qualitative survey results fell within the same themes and sub-themes. Further interviews or survey results could enrich our findings, but may not produce significant additions.
Construct Validity: The interviews or survey could have missing or confusing questions, and there was opportunity for misinterpretation. There is also a risk that participants may not be familiar with particular terminology. However, as all participants had prior experience in testing, this risk is minimized. We also provided a brief introduction before the interviews and the survey to further reduce this risk. The use of semi-structured interviews allowed us to ask follow-up questions. We also conducted pre-testing of the survey.
External Validity: The generalizability of our findings is influenced by the number and background of participants. Our participants represent a variety of development roles, experience levels, and product domains. Therefore, we believe that our results are relatively applicable to the software development industry.
Internal Validity: We applied thematic coding, a qualitative practice that suffers from known bias threats. We mitigated these threats by performing independent coding and comparing results, finding sufficient agreement. We make our results available for further analysis, increasing transparency.

7 Conclusion

Our interviews with testers suggest nine common types of goals pursued when designing test cases, as well as an indication of the relative importance of each goal type. Our findings also shed light on the process of test design for different goals, as well as the factors that can influence this design process.

This research provides a basis for understanding how test design is influenced by particular types of testing goals. Our observations should be confirmed and expanded in future work with further, focused data collection. In particular, we plan to further explore the collective impact of organization factors, team-versus-individual design, and testing goals on the design process. We would also like to explore situations where multiple types of goals are targeted simultaneously during test design. We additionally plan to expand our model of the test design process, with a focus on how knowledge of tester practices can enhance automated test generation.

References

[1] M. Aniche, C. Treude, and A. Zaidman. How developers engineer test cases: An observational study. *IEEE Transactions on Software Engineering*, 48(12):4925–4946, 2022.

[2] E. Barr, M. Harman, P. McMinn, M. Shahbaz, and S. Yoo. The oracle problem in software testing: A survey. *IEEE Transactions on Software Engineering*, 41(5):507–525, May 2015.

[3] A. Beer and R. Ramler. The role of experience in software testing practice. In *2008 34th Euromicro Conference Software Engineering and Advanced Applications*, pages 258–265, 2008.

[4] J. E. Bentley, W. Bank, and N. Charlotte. Software testing fundamentals—concepts, roles, and terminology. In *Proceedings of SAS Conference*, pages 1–12, 2005.

[5] V. Braun and V. Clarke. Using thematic analysis in psychology. *Qualitative Research in Psychology*, 3(2):77–101, 2006.

[6] D. S. Cruzes and T. Dyba. Recommended steps for thematic synthesis in software engineering. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 275–284, 2011.

[7] S. Eldh, H. Hansson, and S. Punnekkat. Analysis of mistakes as a method to improve test case design. In *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*, pages 70–79, 2011.

[8] E. Enoiu and R. Feldt. Towards human-like automated test generation: Perspectives from cognition and problem solving. In *2021 IEEE/ACM 13th International Workshop on Cooperative and Human Aspects of Software Engineering (CHASE)*, pages 123–124, 2021.

[9] E. Enoiu, G. Tukseferi, and R. Feldt. Towards a model of testers' cognitive processes: Software testing as a problem solving approach. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pages 272–279, 2020.

[10] V. Garousi and J. Zhi. A survey of software testing practices in canada. *Journal of Systems and Software*, 86(5):1354–1376, 2013.

[11] G. Gay. One-size-fits-none? improving test generation using context-optimized fitness functions. In *2019 IEEE/ACM 12th International Workshop on Search-Based Software Testing (SBST)*, pages 3–4, 2019.

[12] D. P. Hale and D. A. Haworth. Towards a model of programmers' cognitive processes in software maintenance: A structural learning theory approach for debugging. *Journal of Software Maintenance: Research and Practice*, 3(2):85–106, 1991.

[13] J. E. Hale, S. Sharpe, and D. P. Hale. An evaluation of the cognitive processes of programmers engaged in software debugging. *Journal of Software Maintenance: Research and Practice*, 11(2):73–91, 1999.

[14] I. Karac and B. Turhan. What do we (really) know about test-driven development? *IEEE Software*, 35(4):81–85, 2018.

[15] J. Linaker, S. M. Sulaman, M. Höst, and R. M. de Mello. Guidelines for conducting surveys in software engineering v. 1.1. *Lund University*, 2015.
[16] M. S. Litwin and A. Fink. *How to measure survey reliability and validity*, volume 7. Sage, 1995.
[17] R. McLeod Jr and G. D. Everett. *Software Testing: Testing Across the Entire Software Development Life Cycle*. John Wiley & Sons, 2007.
[18] A. Newell, H. A. Simon, et al. *Human problem solving*, volume 104. Prentice-hall Englewood Cliffs, NJ, 1972.
[19] M. Pezze and M. Young. *Software Test and Analysis: Process, Principles, and Techniques*. John Wiley and Sons, October 2006.
[20] S. Quadri and S. U. Farooq. Software testing–goals, principles, and limitations. *International Journal of Computer Applications*, 6(9):1, 2010.
[21] P. Runeson. A survey of unit testing practices. *IEEE Software*, 23(4):22–29, July 2006.
[22] I. Sommerville. *Software Engineering*. Addison-Wesley Publishing Company, USA, 9th edition, 2010.
[23] G. Upton and I. Cook. *A dictionary of statistics 3e*. Oxford university press, 2014.
[24] J. A. Whittaker. *Exploratory software testing: tips, tricks, tours, and techniques to guide test design*. Pearson Education, 2009.