A Survey on What Developers Think About Testing

Philipp Straubinger
University of Passau
Passau, Germany

Gordon Fraser
University of Passau
Passau, Germany

Abstract—Software is infamous for its poor quality and frequent occurrence of bugs. While there is no doubt that thorough testing is an appropriate answer to ensure sufficient quality, the poor state of software generally suggests that developers may not always engage as thoroughly with testing as they should. This observation aligns with the prevailing belief that developers simply do not like writing tests. In order to determine the truth of this belief, we conducted a comprehensive survey with 21 questions aimed at (1) assessing developers' current engagement with testing and (2) identifying factors influencing their inclination toward testing; that is, whether they would actually like to test more but are inhibited by their work environment, or whether they would really prefer to test even less if given the choice. Drawing on 284 responses from professional software developers, we uncover reasons that positively and negatively impact developers' motivation to test. Notably, reasons for motivation to write more tests encompass not only a general pursuit of software quality but also personal satisfaction. However, developers nevertheless perceive testing as mundane and tend to prioritize other tasks. One approach emerging from the responses to mitigate these negative factors is by providing better recognition for developers' testing efforts.

Index Terms—Motivation, Survey, Software Testing, Software Engineering, Empirical Study

I. INTRODUCTION

Testing plays a pivotal role in the software development process, serving as the primary technique to ensure or improve software quality. If testing is not given due consideration from the outset, the resulting software applications can suffer from fundamental faults and failures, leading to a bad user experience, crashes, incorrect computations, flawed data, and even project failure. This concern is especially pertinent given the sheer scale of today's large software projects with hundreds of thousands of lines of code in companies and open-source. Regrettably, the existence of software quality problems [1] suggests that testing is inadequately applied in practice.

There has been much research and speculation about the reasons for this. Notably, it has been observed that the role of a "software tester" is not always viewed as desirable, as testers often receive limited recognition within companies, and testing tasks are commonly regarded as challenging, with outcomes that are much less tangible than those achieved when writing code [2]–[4]. Testing should thus not only be the responsibility of dedicated testers, but it would be preferable for it to also be an integral part of the work of software developers, who should not only implement but also test features [5], [6]. Unfortunately, evidence suggests that developers tend to engage much less with testing than one might hope for [7].

The lack of developer testing raises the question: Are developers hindered by their work environment, such as time constraints or inadequate testability, do they deliberately choose not to test because they do not enjoy doing it, or do they maybe not have the skills and training to perform better testing? To shed light on these questions we conducted a survey targeting developers in practice, aiming to understand their testing habits and aspirations. Using data collected from 284 respondents to a comprehensive survey comprising 21 questions, we first aim to understand how developers perceive their engagement with testing, the nature of their testing efforts, and whether they feel these efforts receive appropriate recognition. Consequently, our first research question is as follows:

RQ 1: How do developers engage with testing in practice?
Our findings indicate that developers often feel they have insufficient time for testing, resulting in inadequate testing efforts. Furthermore, testability emerges as a practical challenge, leading to intricate tests that are cumbersome to create. Despite these factors, developers express satisfaction with the adequacy of their testing efforts in over half of the projects.

To gain a deeper understanding of the role of developers' motivation in their testing practices, allowing us to differentiate whether they would actually like to test more but are inhibited by their work environment, or whether they would prefer to test even less, our second research question is:

RQ 2: How would developers like to engage with testing?
The responses reveal that developers are split regarding their desire to test more or less: On one hand, the majority express either a reluctance to increase testing or a preference to invest less time in it, often citing testing as a mundane task or deeming it less important than other responsibilities. On the other hand, a substantial number of developers express a desire to engage in more testing if it were technically easier or, notably, if their efforts received greater recognition from management and peers. Some developers are satisfied with their current testing, either having reached a quality goal or believing that further effort would provide no benefits.

In summary, the contributions of this paper are as follows:

- We present a comprehensive survey aimed at assessing the prevailing and desired testing behaviors of developers.
- We quantitatively and qualitatively evaluate the data, providing insights into developers' thoughts and aspirations.
- With the evaluated data, we can strengthen existing research and provide new insights into current problems and misbehaviors in software testing.

Overall, our results suggest that, while technical and organizational issues pose inhibitive factors, motivation significantly influences developers' testing practices. On the one hand, this reinforces ongoing research on automated testing and test generation to relieve developers of some of the challenges but on the other hand this suggests that future research needs to look into increasing the motivation of developers for testing.

II. Background

A. Developer Testing

Testing is usually done either by external testers (e.g., quality assurance departments or companies), or by developers while coding. Both are established practices in companies [8], but in this paper, we focus on developers.

Depending on the current state of development of an application, developers may apply different approaches for testing, and these may lead to different challenges, perceptions, and motivations. During the early stages of the product or of individual features, unit tests target the smallest components of the software (e.g., methods or classes). Developers may also write automated integration tests, checking the interactions of integrated components and their interfaces. Finally, automated end-to-end tests are a common means to check the functionality and satisfaction of requirements at the system level [9].

Independently of the type of tests, a substantial gap between writing production code and test code has been observed [7], [10], with developers spending about 75% of their time for writing code and only about 25% on testing. Interestingly, developers tend to overestimate the time used for writing tests, often claiming a 50 to 50 ratio, while in truth some developers do not test at all [7], [10], sometimes resulting in entire projects without automatically executed tests. In this paper, we aim to understand whether this is caused by technical or organizational challenges, or rather just a lack of motivation.

B. Motivation and Engagement in Software Engineering

We aim to investigate factors of developer motivation while testing, but the term "motivation" is overloaded, and we thus need to consider several different dimensions in our analysis. In everyday language, the terms motivation and satisfaction are used synonymously, but according to the new theory of work motivation and job satisfaction of software engineers [11], [12] they are not the same: Motivation needs to be awakened before the work starts, while satisfaction is caused by results. Motivation and satisfaction are connected because being satisfied by the previous task can motivate the developer for the next task. Motivation can be intrinsic, referring to the inner willingness to do an activity for personal satisfaction, or extrinsic, referring to a separable outcome that comes with or after completing a task, such as recognition for a person's work [13]. The term engagement in the context of software engineering is defined as commitment, hard-working, and interest in the person's current work, which might go beyond the simple motivation to satisfy a task [14] and instead invest extra effort exceeding what is required [15], [16].

In our study, we focus on motivation as the initial factor for developers, as both satisfaction and engagement can only be achieved once they are motivated to test. While factors influencing motivation have been observed for software engineering in general [11], [12], it is not known yet whether these also apply to software testing.

III. Survey Design

To answer RQ1 on the current and RQ2 on the desired engagement with testing (see Section I), we designed a survey based on the guidelines by Linåker et al. [17].

A. Questions

The survey consists of a total of 21 mandatory questions and one optional one, divided into four categories. The questions are either single or multiple-choice questions, with or without an 'others' free-text option, Likert scale choices, and stand-alone free-text questions. Designing a survey involves a trade-off between asking many questions and the resulting difficulty in acquiring survey responses and their costs. We chose and refined questions through an iterative process, where we based them on our research questions (RQs) and refined them through multiple steps within our research group and a pilot study involving different researchers. We tried to keep textual explanations brief and clear to ensure valid responses. This resulted in the revised questions shown in Table I and their answer options included in the artifacts (Section VI).

The survey starts with demographic questions. Since we use Prolific (cf. Section III-B1), an established provider of survey respondents, we do not require questions about information already provided by Prolific, in particular age, country of residence, employment status, sex, and student status. Beyond this, we ask the questions with the ID "UD" listed in Table I regarding the participant's degree (UD1), experience (UD2), number of employees (UD3), and role in the company (UD4).

The second category of the questionnaire consists of questions about the software projects the respondents work in (Table I, IDs with PD). In particular, we query context information such as the project size (PD2, PD3), the working domain (PD1), and quality metrics used in the participant's projects (PD4–6). Together, the demographic and project questions provide context for the testing-related questions.

The third category asks about the current state and efforts for testing in the respondents' projects to answer RQ1. In particular, we ask developers about their daily test behavior (CS1–3), recognition (CS4), and struggles while testing (CS5–7 in Table I). These questions are of special importance for RQ2 likewise, since they serve as a baseline to see the deviation between desires and current testing practices in companies.

In order to learn how the participants would like to test (RQ2) and what they want to change (DS1–4), questions with ID "DS" in Table I enable direct comparison to the previous set of questions (IDs with UD). An additional optional question at the end asks whether participants have any other information to share about testing in their company that has not been sufficiently covered by the previous questions (AE1).

ID	Question	Type
Questions in the pre-study		
PS1	Do you professionally develop/test software?	SC
PS2	Do you currently work on a software project?	SC
PS3	Do you write code or tests?	SC
Questions in the category user demographics		
UD1	What is your highest type of graduation?	SC
UD2	How many years of professional experience do you have?	SC
UD3	How many full-time employees does your company have?	SC
UD4	What is your current role in the company?	SC + free-text
Questions in the category project demographics		
PD1	In which domain is your current project?	SC + free-text
PD2	How many employees are working in your current project?	SC
PD3	What is the size of your current team?	SC
PD4	What are the metrics used for measuring quality in your project?	MC + free-text
PD5	What is the overall code coverage in your project?	SC
PD6	Is there a requirement for quality in the project?	SC + free-text
Questions in the category current situation		
CS1	How much time of your daily work do you use for testing?	SC
CS2	What kind of tests do you write/perform?	MC + free-text
CS3	Do you prefer to test or write code?	Likert 5 point
CS4	What kind of recognition do you get for writing tests?	MC + free-text
CS5	In your opinion, has your current project been tested well enough?	SC + free-text
CS6	How much effort does it take to write tests for the project?	Likert 5 points
CS7	How complex are your tests?	Likert 5 points
Questions in the category desired situation		
DS1	Would you like to use more or less of your daily work for testing?	SC
DS2	Why do you want to spend more or less time for testing?	Free-text
DS3	What kind of tests do you want to write/perform the most?	MC + free-text
DS4	Would you spend more time on testing if it would be recognized?	Likert 7 point

B. Survey Tools

We implemented the survey using *Prolific* to recruit participants and *SoSci Survey* to host the survey itself.

1) Prolific: Prolific[1] is an online platform to recruit participants for different kinds of studies like interviews and surveys. Together with Mechanical Turk (MTurk)[2], Prolific is one of the biggest recruitment platforms for participants. Prolific has some clear advantages to MTurk, since there are clear rules for both researchers and participants. All involved parties know about payments, obligations, and rights, and researchers also have better insights into the pool of possible participants [18]. MTurk provides more participants (over 250,000), but most of them are located in the US [19] while the more than 150,000 participants of Prolific are better distributed globally (Section IV-A). There is also empirical evidence that Prolific provides data of higher quality with less cheating and higher attention rates than MTurk [20].

Recruiting participants with Prolific is not free, since both the participants and Prolific itself require payment. Our survey is set for an estimated completion time of ten minutes with an hourly rate of 10.50 £, which means every respondent received 1.75 £ for completing the survey. The advantages of participants acquired by Prolific are that only preselected participants are permitted to take part in the survey and that they are motivated by their payment and approval score, which influences their future commissioning. Participants receive payment only after approval of their answers by the client.

[1]https://www.prolific.co/
[2]https://www.mturk.com/

2) SoSci Survey: SoSci Survey[3] is a powerful online platform to compose questionnaires with flexibility and individual design. SoSci Survey was designed for university research in 2003, has been under constant development ever since, and is free to use for researchers. The platform provides an easy, yet powerful editor for different kinds of questions and the collected data can be exported in various ways.

C. Participants

The target population of our survey consists of software developers since we want to understand their current and desired testing behavior. Prolific allows to pre-screen users based on demographic information as well as their self-declared expertise. We excluded users who do not work full- or part-time, and selected 17 terms related to software engineering (e.g., debugging, version control) out of the hundreds of possible terms provided by Prolific to filter by relevant expertise. To increase trust in the participants' answers, we only accepted participants with an approval rate of 100 %. This rate is maintained by Prolific to keep track of how satisfied study conductors are with their respondents.

After pre-screening, a pool of 9,156 eligible participants who had been active on Prolific in the last 90 days remained. Since we are only interested in (1) professional software developers who (2) currently work on a software project and (3) write code or tests, we used a pre-study (Table I, IDs with PS) to filter the possible participants further. As users also receive payment for completing the pre-study, we requested 600 responses to the pre-study from Prolific, of which 284 answered all three pre-study questions in the affirmative. The final data is based on the responses of these 284 participants to the main survey, all of whom answered it completely.

D. Analysis of Responses

To analyze open-ended questions (PD6, CS5, DS2, AE1), we used *qualitative content analysis* [21]. For each free-text question, we defined an empty set of codes/categories. While going through the answers manually, we added new categories whenever we encountered a new idea or perspective. If more than one participant mentioned the same idea, we used the same code for both (Table II). Each code means that the participants mention this category in their answers. The coding was independently done by two researchers, who discussed and resolved all points of disagreement. A summary of the analysis is given in Sections IV-A2 (Requirement for quality), IV-B (Tested well enough), and IV-C (More or less time). The questions with ID "CS" (Table I) are used to answer RQ1, while the one with ID "DS" is for RQ2.

Closed-ended questions are analyzed by visualizing and bringing them into context with the research questions. We also measured the Spearman rank correlation matrix [22] as well as the multiple linear regression [23] and ordinal logistic regression [24] matrices for all our variables to find dependent and significant variables.

[3]https://www.soscisurvey.de

TABLE II
CODES FOUND DURING THE QUALITATIVE CONTENT ANALYSIS

Requirement for quality (PD6)	Tested well enough (CS5)	More or less time (DS1)	Anything else (AE1)
Customer Satisfaction	Not valued	Skill	Boring
Mutation Score	No structure	Learn	Business
Review	Client	Like testing	Data
Bugs/Defects	Bugs	More important tasks	Project
Time	Data	Dislike testing	Recognition
Requirements	Resources	Process	Skill
Approval	Not needed	Satisfaction	Testing
Manual	More testing	Not enough testing	Time
Test Plan	- Missing parts	Quality	
Code Coverage	- Too big	- Customer satisfaction	
	- Too early	- Solving issues	
	- Forgetting	- Avoid issues	
	- Edge cases	Right amount of testing	
	Skill		
	Time		

E. Threats to Validity

Threats to *internal validity* arise since the participants are distributed across several countries, the questions asked may be misunderstood or misinterpreted because of local differences in the language or because English is not their first language. This risk is reduced by our pilot study through which ambiguities in the questions were removed. The participants received the remuneration regardless of the time needed to finish the questionnaire, which may impact the quality of responses; however, respondents receive payment only if their answers are approved. Possible inconsistencies in the categorization of free-text responses were addressed by two researchers independently coding and resolving disagreements.

There may also be threats to *external validity* as the participants may not be distributed globally well enough to generalize the results, and the sampling algorithm of Prolific may be biased. Answers may not be from the perspective of a developer but from, e.g., a manager or consultant, since they may take different roles within their teams. However, since all participants stated they are currently coding or testing in a project, we assume they have insights of a developer.

Threats to *construct validity* arise from the design of the survey. The questions may not be specific enough to measure relationships between them and answer options may be missing. Despite the evaluation of the questionnaire in a pilot study, some questions and possible answers can nevertheless be misinterpreted by the participants because of missing explanations about tools and metrics asked. There may also be topics related to the research questions (e.g., whether the participants formal training or use test automation tools) that were not asked in the survey but could have given more insights into the subject, which is why an optional answer in the end was added (AE1).

IV. SURVEY RESULTS

A. Demographics

The survey questions provide demographic information at the level of individual participants as well as their projects.

1) Participants: The participants in our study exhibit a diverse range of ages, spanning from 18 to 52 years old. The majority of participants fall within the age range of 22 to 31, with a decreasing number of participants as age increases. A plausible explanation for this trend could be that as individuals progress in their software development

careers, they may transition into managerial roles with reduced involvement in coding activities. This trend aligns with the Developer Survey conducted by Stack Overflow [25], where the majority of developers were between 18 and 34 years old, with few participants exceeding the age of 44.

Approximately 31 % of our participants are students, which is expected considering a significant portion of our participants are under 30 years old. Interestingly, only 58 % of the students reported working part-time in a company, while the remaining 42 % work full-time while simultaneously pursuing their studies. The majority of our participants have already graduated with a university degree, with only 42 individuals reporting employment in a company with a High School diploma or equivalent (Table III). With 80 % of the participants possessing a Bachelor's or Master's degree, the respondents slightly exceed the average qualification reported by the Developer Survey, where about 75 % reported a similar degree [25].

The diverse range of experience levels from less than one year (9.5 %) to more than ten years (21.1 %) allows for a comprehensive exploration of different perspectives on testing. Additionally, representatives from various companies and company sizes (Table III) providing insights into a wide array of development and testing processes.

Current research [26]–[29] and practice [30], [31] suggest that testing already needs to be done during the development phase, which implies that developers should at least perform unit testing to ensure the software's quality [32]. Consequently, we targeted developers specifically and not testers in this work, and indeed only 15 of our participants stated in their answers that there is a dedicated QA department rather than developer testing [33] in their company. While the majority of participants (62 %) identify their current role as developers, our dataset also includes consultants, testers, and managers. While these are not strictly software developers, we accepted them nevertheless with our pre-study criteria because of their active involvement in software development.

The participants come from diverse geographical locations worldwide, with a particular emphasis on Europe and North America. The distribution of participants is detailed in Table III, which also includes additional demographic information. Interestingly, over 25 % of the participants are from Portugal, but the study encompasses participants from countries outside of Europe, such as South Africa, New Zealand, and Chile, contributing to a globally representative sample.

Our study includes data from 20 % female participants. While this figure falls below the average of women in computer science with 25 % reported previously [34], the same study found a general decline in female computer science graduates by 8 % to 17 %. This suggests that our study achieves an average or potentially higher representation of female participants following the reported decline [34].

2) Projects: The global distribution of participants results in a wide variety of software projects spanning different domains. Table IV provides an overview of project demographics and highlights that the majority of projects are concentrated in the communication and finance domains. However, numer-

TABLE III
DEMOGRAPHICS OF THE PARTICIPANTS

Variable	Number of participants	Number of participants in percentage	Variable	Number of participants	Number of participants in percentage
Country of Residence			**Highest type of graduation**		
Germany	6	2.1 %	Ph.D.	10	3.5 %
Hungary	9	3.2 %	Master	81	28.5 %
Greece	9	3.2 %	Bachelor	152	53.5 %
Netherlands	10	3.5 %	High School	42	14.8 %
Canada	10	3.5 %	**Years of professional experience**		
South Africa	12	4.2 %	>10	60	21.1 %
Mexico	12	4.2 %	6 - 10	45	15.8 %
Spain	13	4.6 %	3 - 5	89	31.3 %
Others	17	6.0 %	1 - 2	64	22.5 %
Italy	21	7.4 %	<1	27	9.5 %
United Kingdom	21	7.4 %	**Employees in the company**		
Poland	27	9.5 %	>500	84	29.6 %
United States	36	12.7 %	251 - 500	34	12.0 %
Portugal	74	26.1 %	101 - 250	35	12.3 %
			51 - 100	24	8.5 %
			26 - 50	24	8.5 %
			11 - 25	30	10.6 %
			<10	54	19.0 %

TABLE IV
DEMOGRAPHICS OF THE PROJECTS

Variable	Number of participants	Number of participants in percentage	Variable	Number of participants	Number of participants in percentage
Domain			**Employees in the project**		
Military	2	0.7 %	<5	106	37.3 %
Insurance	5	1.8 %	5 - 9	89	31.3 %
Automotive	13	4.6 %	10 - 19	48	16.9 %
Food	14	4.9 %	20 - 50	25	8.8 %
Government	15	5.3 %	>50	17	6.0 %
Entertainment	16	5.6 %	**Team sizes**		
Health	19	6.7 %	1	11	3.9 %
Logistics	21	7.4 %	2 - 3	76	26.8 %
Games	22	7.7 %	4 - 6	109	38.4 %
Public Services	24	8.5 %	7 - 10	68	23.9 %
Finance	38	13.4 %	>10	21	7.4 %
Communication	40	14.1 %	**Code Coverage**		
Other	56	19.7 %	Not measured/available	90	31.7 %
Quality measurements			0% - 20%	19	6.7 %
Code Coverage	130	45.8 %	21% - 40%	23	8.1 %
Mutation Score	23	8.1 %	41% - 60%	49	17.3 %
Defects opened/closed	156	54.9 %	61% - 80%	71	25.0 %
None	60	21.1 %	81% - 100%	33	11.6 %
Other	14	4.9 %			

ous other domains, such as education, robotics, fashion, and real estate, are also represented in the dataset. A significant proportion of projects (70 %) are realized by teams with fewer than ten members, while only 6 % of the projects involve more than 50 staff members. This suggests that many projects might have adopted agile methodologies using small teams [35].

B. RQ 1: How do developers engage with testing in practice?

1) How much time do developers invest in testing? The time invested in testing activities among participants (CS1) reveals that a majority (more than 80 %) dedicate relatively limited time to testing (less than 40 %) during their work hours (Fig. 1). Additionally, a great portion (more than 44 %) allocates even less time to testing (less than 20 %), which will be further analyzed and investigated in Section IV-C.

TABLE V
KIND OF TESTS THE PARTICIPANTS ARE PERFORMING VERSUS WHAT THEY WANT TO PERFORM

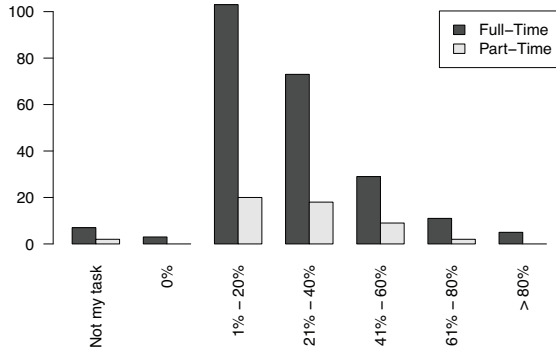| Kind of tests	Tests currently performed		Tests wanted to perform	
	Number	Percentage	Number	Percentage
Unit tests	147	51.8 %	139	48.9 %
Integration tests	121	42.6 %	81	28.5 %
System tests	98	34.5 %	71	25.0 %
API tests	91	32.0 %	64	22.5 %
UI tests	102	35.9 %	66	23.2 %
Manual tests	158	55.6 %	68	23.9 %
Smoke tests	27	9.5 %	17	6.0 %
Sanity tests	28	9.9 %	21	7.4 %
Regression tests	67	23.6 %	46	16.2 %
Other	7	2.5 %	12	4.2 %

Fig. 1. Proportion of the working time that is used for testing with full- and part-time jobs by the participants (CS1)
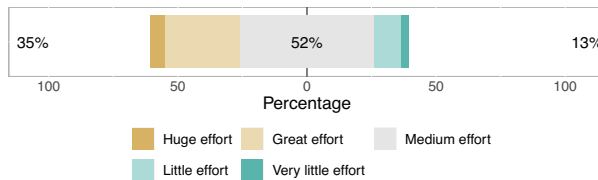

Fig. 2. Effort needed for testing (CS6)

Fig. 3. Complexity of the tests written by participants (CS7)

Fig. 4. Complexity and effort combined from questions CS6 and CS7

2) What kind of tests do developers write? The respondents apply a variety of testing approaches (CS2). Table V illustrates that both unit and manual testing are carried out by over half of the participants (51.8 %), making them the primary test types followed by integration tests with 42.6 %. Other types of tests, such as smoke or sanity tests, are less commonly applied, each accounting for less than 10 %. The reasons for this disparity could be multifaceted, such as a lack of training or the perception that functional testing is more important.

3) How much effort does it take to write tests? The level of effort required for writing tests in participants' projects is illustrated in Fig. 2 (CS6). The data reveals that only 13 % of participants consider testing to be easy while more than half find testing to be challenging, indicating moderate effort involved. On the other hand, 35 % face significant issues in testing, as it demands great effort to test their project.

The complexity of tests is influenced by the effort required (CS7). Among the participants, 44 % primarily write simpler tests, maybe aiming to achieve quality goals with minimal effort (Fig. 3). Roughly one-third write both complex and simple tests, while only 21 % develop complex tests to ensure quality and address edge cases (based on answers of CS5).

Figure 4 highlights the relationship between the complexity of tests and the effort expended to write them, indicating that as the complexity of tests increases, the effort to write them increases, too. An intuitive explanation would be that poor design amplifies the effort required for writing tests, resulting in more intricate tests and discouraged developers.

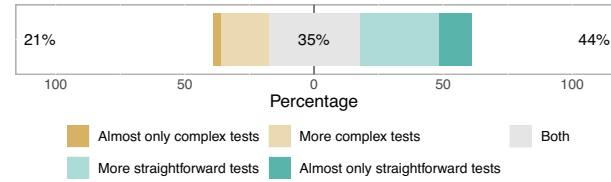*4) Do developers follow certain quality metrics and requirements?* Code coverage is a widely utilized metric for quality [36] and a first indicator of a project's testing state. Nevertheless, only 45 % of the participants' projects use code coverage (PD4). In certain cases, answers indicate that measuring coverage may be infeasible, such as with embedded software, or too computationally expensive [37]. Overall, developers may lack awareness of the true quality of their projects due to insufficient or inadequate metrics, which we believe can influence their motivation to test.

Furthermore, only 27 % of participants' projects have specific quality requirements or goals, as indicated by question PD6 (Table I): 31 participants mention code coverage as their quality goal, while others emphasize the need for minimal defects or the basic functionality of the software, which may not be easily measurable. Some participants rely on management approval or feedback from quality assurance departments, while others prioritize customer satisfaction or peer reviews. Two projects even use mutation scores as their testing goal. Different forms of reviews, evaluations by superiors, or input from customers, are used, too. In certain projects, timely completion plays a crucial role to meet deadlines, even at the expense of quality. Projects that undergo requirements engineering in the first place rely on the fulfillment of those or adherence to a test plan as their indicators of quality. Moreover, the lack of specific quality goals may result in misguided assumptions about code quality and the overall necessity of testing since their perception of quality is based on their personal feeling rather than a quality metric.

5) Do developers believe their projects are sufficiently tested? Approximately 60 % of the developers expressed confidence in the level of testing conducted in their projects (CS5). This percentage is notably high considering the limited amount of time dedicated to testing by the participants.

However, many developers also highlighted various factors that contribute to perceived inadequacies in testing. One common concern raised by developers in the free-text field
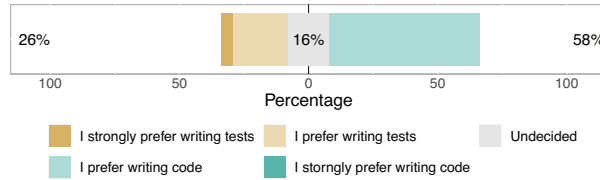
85

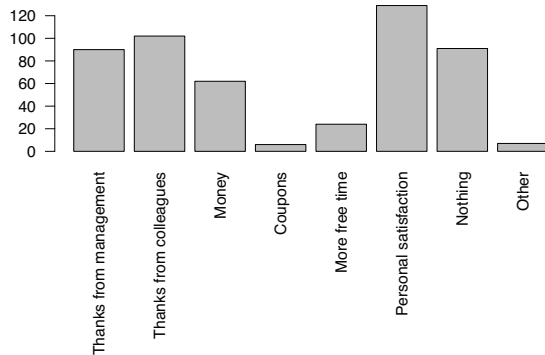Fig. 5. The preference to write code or tests (CS3)

Fig. 6. Recognition for testing (CS4)

of CS5 is the undervaluation of testing within their companies. Developers often assign higher priority to other tasks and allocate their time accordingly, neglecting comprehensive testing. Another factor mentioned to influence the perceived lack of testing is the absence of well-defined test processes, which is thought to lead to issues such as poor documentation, insufficient communication between developers and testers, and overall inadequate testing practices.

Nominal client involvement was cited as a reason for minimal engagement with testing or a focus on new features because there is no requirement to write tests. Additionally, some developers believe their software lacks proper testing because evidence shows that the software still contains bugs, unresolved defects, or uses artificial data for testing. Projects with untested components or pending evaluation of critical edge cases are also cited as requiring more testing.

The lack of time and resources emerges from CS5 as a recurring constraint that hampers proper testing. Many participants express a desire to engage in testing activities but are constrained by factors beyond their control. Overall, developers' perceptions of sufficient testing vary and are influenced by factors such as prioritization, development process maturity, client engagement, and resource limitations.

6) How is testing recognized? Approximately one-third of developers receive an acknowledgment from both management and colleagues (Fig. 6), demonstrating that testing is valued in their respective companies and projects while notable 25 % of developers receive monetary rewards for their testing efforts in addition (CS4). Furthermore, different types of external recognition, or the lack thereof, are stated by the participants in

question AE1. One participant stated that more code coverage in the project proved to lead to fewer bugs in the remaining project and that a "reputation of writing 'bug-free' software (or as close as possible)" (P65) was given to the participant for it. Enough testing does not only improve quality but also shows that the software is working as intended efficiently and the risk of failing the project is minimized (P193). Some describe that their company understands the importance of software testing (e.g., P33), but there are many more who do not (e.g., P95).

Nearly half of the participants (around 50 %) claim to be intrinsically motivated and test for personal satisfaction, even though only a minority of the developers see testing as a crucial part of the development process and are engaged in testing and enjoy it. For example, one participant compared testing with "a fun puzzle to figure out" (P65) and another is satisfied with the little effort required for testing because their software is designed for it (P11). Discovering hidden bugs and issues that would have gone into production without sufficient testing are also mentioned as motivating (P66), as well as the resulting time savings (P48). Thoroughly testing the participant's software gives the developers peace of mind when they are sure "they did not mess things up" (P61). A good test suite also improves the ability to refactor the code base when the application slowly evolves over the years, because bugs introduced during maintenance can be found by the existing test suite (P65). It also gives the developers personal satisfaction when their code works smoothly and users can work without frustration (P65). We believe that this intrinsic motivation serves as a driving force for writing tests and investing effort beyond what is formally required.

On the other hand, it is concerning that approximately one-third of developers do not receive any form of external recognition or have no intrinsic motivation for testing (e.g., P95). This lack of acknowledgment and motivation can potentially result in a reduced inclination to write tests, ultimately impacting the overall software quality. In addition, the majority of developers (58 %, Fig. 5) prefer writing code over tests (CS3).

We applied both correlation and regression analysis to the demographic variables about the participants (Table I, IDs with UD) and the projects (IDs with PD), and the variables of the current state of testing in their project (IDs with CS). Unfortunately, no significant or dependent variables could be found to explain our findings. Consequently, we cannot report whether the current state of testing is dependent on any of the participant or project demographics we considered.

Summary (*RQ 1*): Testing is not the favorite task of developers. The time invested in testing is limited and sometimes takes great effort. Even though many believe their projects are tested well enough, common demotivating factors are higher prioritized tasks, bad communication, and missing recognition.

C. RQ 2: How would developers actually like to engage with testing?

The participants are split (DS1) between those who want to test more, those who do not, and those who are satisfied
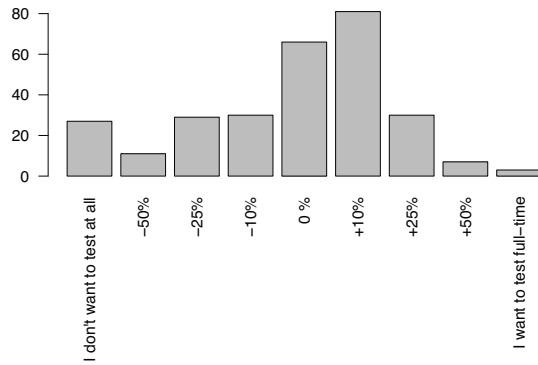
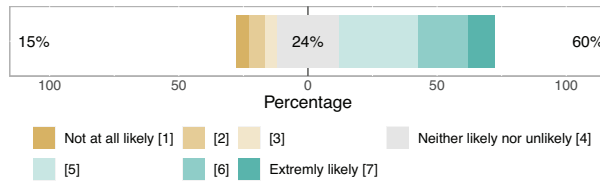Fig. 7. Additional testing time the participants want to use (DS1)

Fig. 8. More or less testing if better recognized (DS4)

with their current testing efforts (Fig. 7). While 42 % of the participants want to test more, there are 58 % who do not, either because they want to test less (24 %) or because they are satisfied with their current testing effort (34 %).

1) Why do developers want to test more? Most of the 42 % who want to test more estimate they would use 10 % more of their time for testing. The more time participants already use for testing, the less they want to reduce it. Most of those respondents not testing at all do not want to start with it. In contrast, the more time the participants already invest in testing, the less they want to reduce this amount. This could mean the more they test the more they understand that testing increases the quality and decreases their workload later.

To investigate why developers want to test more in more detail, we analyze the free-text answers to the mandatory question of why they want to test more or less (DS2), where we observe the following reasons:

- Find more issues, blockers, and edge cases (37)
- Ensure and improve quality (35)
- Like to test their own and their colleagues' code (17)
- Decreased maintenance expenses (15)
- Satisfied to show others that their code is working fine and better than other's (9)
- Improve their coding and testing skills (5)
- Try new aspects of development and new technologies (1)
- Improve strategic thinking and solve problems (1)
- Increase variety in their daily work (1)
- Find areas for improvements and refactoring (1)
- Incorporate new team members faster (1)

The observation that lack of recognition inhibits motivation is

confirmed by question (DS4), which shows that nearly 60 % of the participants would test more if it were better recognized (Fig. 8). Overall, the environment seems to have a greater impact on developers than their personal motivation.

> **Suggestion 1.** The recognition of testing within companies should be increased for both developers and at the management level to motivate more engagement with testing.

The answers to question AE1 provide further insights into why developers have a desire to test more: They appreciate the positive outcomes of testing, such as improved quality (P239), as well as the opportunity to solve bugs (P67). In one particular company, the quality assurance team is larger than the development team (P17), which leads to management valuing the time, resources, and effort dedicated to testing (P17). Many participants are motivated to test because they have access to a testing environment that closely resembles the production environment (P11), there is effective communication between developers and testers (P215), and they receive feedback from end-users (P268). Some participants have already completed software testing courses provided by their company and see the benefits of their involvement in the project they are working on (P45). These participants express a need and desire for training in writing software tests and managing test suites and infrastructure, as they recognize the potential benefits (e.g., P45). One participant describes that they have a very good policy for testing, where everyone gets trained to write tests (P33). Some participants see testing as an integral task inside the development cycle (P27) where all code should be accompanied by unit tests (P194), part of a review system (P194), and all tests should run continuously (P218).

2) Why are developers satisfied with their current testing efforts? Almost one-quarter of the participants are satisfied with the amount of time currently invested in testing (Fig. 7). Of these 66 developers, only eight do not write tests and also do not want to. Participants mention a well-set-up and well-handled testing environment or that they are already in maintenance without developing new features. Three participants also point out that there is a dedicated tester in their team or even a separate quality assurance department for testing. There are also projects which currently meet the given quality goal or the developers can choose their used time for testing. Those answers are reflected in the free-text answers to DS2:

- Amount of testing is adequate (33)
- Another person/department responsible for testing (3)
- Quality goal reached (2)
- Well set-up testing process (1)
- Project in maintenance (1)

3) Why do developers want to test less? Out of all respondents, 34 % want to test less than currently, with almost the same number in the categories -10 %, -25 %, and participants who do not want to test at all (DS1). Especially the percentage of participants who do not want to test at all with almost 10 % is high compared to the participants who want to test full-time at about 1 %, which means there is a great contrast between

87

engagement to write tests and code. Many developers do not mind testing in general but do not want to test the code of their colleagues, because it is not well-written or hard to understand.

> **Suggestion 2.** Developers should always write unit tests for their own code to support other developers working with the code, who are unlikely to write these tests.

In comparison with the currently performed tests in the participants' projects (Table V), the number of tests the participants want to perform decreases (DS3). Unit testing is least affected (3 %) which confirms that unit tests are the most popular kind of tests performed by developers, while manual testing is the least popular one decreasing more than 50 %.

In the free-text responses, we identified several reasons why developers want to engage less with testing (DS2):

- Like coding better and dislike testing (70)
- No working testing process and infrastructure (30)
- Do more important things and meet deadlines (21)
- Testing as boring, frustrating, and repetitive activity (19)
- Lack of testing skills (15)
- Missing resources to test well enough (13)
- No proper training in testing (4)
- Project is tested well enough (1)
- Project is too small or old to be tested (1)
- Software will be replaced soon (1)
- There are other developers in charge of testing (1)
- Find manual testing exhausting and unnecessary (1)
- Lack of communication (1)

During the analysis of the answers to question AE1, we discovered more detailed insights into why developers want to test less. For example, manual tests are considered boring and time-wasting by the respondents (e.g., P49), especially when they have to be done over and over again when versions change (P27). Since many participants consider testing as boring, they think "it takes time away from more important issues" (P125) like implementing new features (P137). Because the participants do not want to test these new features either, more bugs may be introduced (P218). The development process in general is depicted as not thought through (P68) which causes the developers to de-prioritize testing (P140) and dislike the current process (P121). Another demotivating factor is a lack of infrastructure like continuous integration (P49), and that tests are badly implemented by colleagues and predecessors (P196), which causes disengagement from testing.

> **Suggestion 3.** Whenever feasible, software tests should be designed to be automatically executed and supported by a continuous integration infrastructure.

Furthermore, managers are mentioned to be interested in testing only when "a major or critical bug that is disruptive to the business is found" (P12), only to get back to old habits when the crisis is averted. Everything the customer does not recognize or value with money will be neglected or de-prioritized (P252). Especially in start-ups or small companies, each employee has more than one task in the company (P86),

which can cause the testing to be forgotten (P86). Others have to use "in-house developed tools" (P104) which have low usability or simply do not work as needed. This prevents developers from engaging with testing as they want to (P90).

> **Suggestion 4.** Awareness at the management level needs to be raised about the importance of testing activities to increase and maintain the quality of software.

Lack of test data and scenarios is mentioned as a source of problems, for example when there is insufficient test data that is as close as possible to production data (P164). Since many of the participants only use fake, sample, or mocked data, their tests are not considered "completely trustworthy" (P4) to them. The testers also need a variance of test data to include edge cases in their test suites, but most of the projects and customers only provide generic test data without covering special cases (P164). Since the developers do not know the interactions of the end-users with the application, either because the product is not released yet (P42) or because user data is not collected (P73), they cannot create test scenarios that are as close as possible to reality. The lack of these scenarios can lead to users breaking the system even if it has been tested well, and developers then receive the blame from management (P42). In addition, the lack of important information and skills (P26) makes it hard to write good and sustainable test suites (P26), and developers get frustrated by all the failures and changes because of missing data and communication (P138).

> **Suggestion 5.** Requirements, common scenarios and problems should be communicated among all stakeholders, including developers, to decrease the time for troubleshooting.

Software in some domains seems particularly difficult to test, like games (P74), simulators (P74), embedded devices (P200), or software with hardware requirements (in particular when the required hardware is not available, P128). There are also old projects that contain many bugs and lack a testing infrastructure (P78), which are mostly replaced with new ones including an automated test suite and documentation (P78).

> **Suggestion 6.** Developers require sufficient test data and robust infrastructure for automated test executions to write tests and avoid demotivational factors.

In addition to identifying various blockers, we also observed different attitudes among the developers. For instance, one participant believes that others who work fewer hours should be responsible for writing tests ("There are people working less than me so they should be doing the testing", P69). Some participants consider their projects too small to warrant testing at all(P3). On the other hand, one of the participants stated to prioritize simplicity and reliability in their code rather than creating a complex and robust product (P121).

> **Suggestion 7.** Developers should be made aware of the significance of testing, ensuring they understand the advantages of testing as well as the drawbacks of neglecting it.

We also noticed that many developers are frustrated by the lack of time (P140) and training (P38) they are given. Not only are lack of testing skills in general mentioned, but some think it is especially hard to write tests for edge cases (P185). Some participants would like to have more or better training in writing good tests (e.g., P38) which may result in better test suites and an increase in the quality of their products. Another important issue is the lack of time for writing tests given by the management or the customer (P65). Some participants consider this as the main problem when the quality of the software does not meet the excepted one (P44). Others have enough time for testing and a test suite, but it takes too long to execute all of them (P187).

> **Suggestion 8.** Developers would benefit from software testing training programs to gain the necessary skills, knowledge and mindset to execute effective testing practices.

We applied both correlation and regression analysis to the demographic variables about the participants (Table I, IDs with UD) and the projects (IDs with PD), and the variables of the current (IDs with CS) and desired state of testing in their project (IDs with DS). Unfortunately, also in this case no significant or dependent variables could be found to explain our findings. This means that we cannot make any statements about how the desired state of testing depends on the current state of testing or any of the participant or project demographics we asked for.

> **Summary (*RQ 2*):** Developers claim they would write more tests to ensure quality and increase personal satisfaction, but would like to receive better recognition for this. On the other hand, many developers want to test less than they currently do since they perceive testing as boring compared to other tasks.

V. RELATED WORK

Most of the time developers spend in their IDEs, they are reading, writing, and modifying their code. Only about 9 % of their time is used for writing and executing tests, which was found in a study with 40 students [26]. Since the study contained students, it may not generalize to companies, which is why almost 2,500 developers were monitored in companies for 2.5 years in a follow-up study, showing that developers spend a quarter of their work for testing—while believing they test half of their time [7]. Our respondents appear to be slightly more realistic, estimating their testing at 40 % or less.

Developers would like to execute their tests more frequently but are handicapped by difficult testing frameworks as well as too little time given by management. This insight and that there is only a weak correlation between writing code and executing tests were found during a study with subsequent interviews [38]. While this study focused on executing rather than writing tests, their conclusions about testing frameworks, conditions, and lack of time are confirmed by our survey.

A recent survey on unit testing practices [27] found that the developers are primarily driven by their conviction to test

and management requirements, which matches our results. In addition, these survey results show that developers focus on writing, refactoring, and fixing code instead of writing tests because they do not enjoy testing, which can also be seen in the answers of our participants. In contrast to this work, we do not focus on unit tests, but on testing in general.

According to a survey about thoughts on the career of a software tester [3], both students and professional testers think that testing is an important part of the software development cycle, but also tedious, frustrating and that they are missing developing software. Another study [4] also came to the same conclusion that missing recognition is one of the main reasons why testers are not satisfied with their job. In this work, we set our focus on the aspect of developer testing, but our findings are in agreement with these findings on software testers.

Developers and testers in Brazilian companies have been reported to lack both training and knowledge in fundamental testing concepts [39]. Moreover, testing itself apparently is generally not viewed as an important and prioritized activity in the Brazilian companies involved in this study, which aligns with our global findings.

VI. CONCLUSIONS

Insufficient testing is known to affect software quality. In order to better understand whether developers do not engage with testing because of technical, organizational, or motivational challenges, we conducted a survey with 284 participants. We find evidence that all three factors inhibit testing, and the details of how these factors affect testing can inform future research on how to improve testing practices.

Of the many reasons that inhibit effective testing, we found evidence of a lack of intrinsic and extrinsic motivation in developers. Consequently, one potential opportunity to motivate developers to write more tests would be the application of external recognition systems, such as gamification, i.e., the inclusion of game elements to non-game related tools and contexts [40]. Such tools and approaches have been shown to provide benefits towards the motivation of developers in practice and educational scenarios to better engage with software testing [41].

To increase the generalization of the finding of this paper, it would be useful to replicate the survey with different audiences. To support replication, we provide a replication package containing all data and information: https://doi.org/10.6084/m9.figshare.23212562

REFERENCES

[1] "The cost of poor software quality in the us: A 2020 report." https://www.it-cisq.org/the-cost-of-poor-software-quality-in-the-us-a-2020-report.htm. Accessed: 30.08.2022.

[2] E. J. Weyuker, T. J. Ostrand, J. Brophy, and R. Prasad, "Clearing a career path for software testers," *IEEE Software*, vol. 17, no. 2, pp. 76–82, 2000.

[3] P. K. Waychal and L. F. Capretz, "Why a testing career is not the first choice of engineers," *CoRR*, vol. abs/1612.00734, 2016.

[4] A. Deak, T. Stålhane, and G. Sindre, "Challenges and strategies for motivating software testing personnel," *Inf. Softw. Technol.*, vol. 73, pp. 1–15, 2016.

[5] A. Spillner and T. Linz, *Software Testing Foundations: A Study Guide for the Certified Tester Exam-Foundation Level-ISTQB® Compliant*. dpunkt. verlag, 2021.

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

[7] M. Beller, G. Gousios, A. Panichella, S. Proksch, S. Amann, and A. Zaidman, "Developer testing in the IDE: patterns, beliefs, and behavior," *IEEE Trans. Software Eng.*, vol. 45, no. 3, pp. 261–284, 2019.

[8] G. J. Myers, C. Sandler, and T. Badgett, *The art of software testing*. John Wiley & Sons, 3 ed., 2012.

[9] A. Spillner and T. Linz, *Basiswissen Softwaretest: Aus- und Weiterbildung zum Certified Tester – Foundation Level nach ISTQB-Standard*. dpunkt, 5 ed., 2012.

[10] M. Beller, G. Gousios, A. Panichella, and A. Zaidman, "When, how, and why developers (do not) test in their ides," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2015, (New York, NY, USA), p. 179–190, Association for Computing Machinery, 2015.

[11] A. C. C. França, "A theory of motivation and satisfaction of software engineers," 2014.

[12] A. C. C. França, H. Sharp, and F. Q. B. da Silva, "Motivated software engineers are engaged and focused, while satisfied ones are happy," in *2014 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM '14, Torino, Italy, September 18-19, 2014* (M. Morisio, T. Dybå, and M. Torchiano, eds.), pp. 32:1–32:8, ACM, 2014.

[13] R. M. Ryan and E. L. Deci, "Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being.," *American psychologist*, vol. 55, no. 1, p. 68, 2000.

[14] C. França, F. Q. B. da Silva, and H. Sharp, "Motivation and satisfaction of software engineers," *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 118–140, 2020.

[15] P. Suff and P. Reilly, "Going the extra mile. the relationship between reward and employee engagement," *Institute for Employment Studies, University of Sussex Campus, UK*, 2008.

[16] V. R. Devi, "Employee engagement is a two-way street," *Human resource management international digest*, 2009.

[17] J. Linåker, S. Sulaman, R. Maiani de Mello, and M. Höst, *Guidelines for Conducting Surveys in Software Engineering*. 2015.

[18] S. Palan and C. Schitter, "Prolific.ac—a subject pool for online experiments," *Journal of Behavioral and Experimental Finance*, vol. 17, pp. 22–27, 2018.

[19] J. Robinson, C. Rosenzweig, A. J. Moss, and L. Litman, "Tapped out or barely tapped? recommendations for how to harness the vast and largely unused potential of the mechanical turk participant pool," *PLOS ONE*, vol. 14, p. e0226394, dec 2019.

[20] E. Peer, D. Rothschild, A. Gordon, Z. Evernden, and E. Damer, "Data quality of platforms and panels for online behavioral research," *Behavior Research Methods*, vol. 54, no. 4, pp. 1643–1662, 2022.

[21] M. D. White and E. E. Marsh, "Content analysis: A flexible methodology," *Libr. Trends*, vol. 55, no. 1, pp. 22–45, 2006.

[22] M. G. Kendall, "Rank correlation methods.," 1948.

[23] D. A. Freedman, *Statistical models: theory and practice*. cambridge university press, 2009.

[24] P. McCullagh, "Regression models for ordinal data," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 42, no. 2, pp. 109–127, 1980.

[25] Stack Overflow, "Developer Survey." https://insights.stackoverflow.com/survey/2021, 2021. Accessed: 2021-08-16.

[26] M. Beller, G. Gousios, and A. Zaidman, "How (much) do developers test?," in *37th IEEE/ACM International Conference on Software Engineering, ICSE 2015, Florence, Italy, May 16-24, 2015, Volume 2* (A. Bertolino, G. Canfora, and S. G. Elbaum, eds.), pp. 559–562, IEEE Computer Society, 2015.

[27] E. Daka and G. Fraser, "A survey on unit testing practices and problems," in *25th IEEE International Symposium on Software Reliability Engineering, ISSRE 2014, Naples, Italy, November 3-6, 2014*, pp. 201–211, IEEE Computer Society, 2014.

[28] E. M. Maximilien and L. A. Williams, "Assessing test-driven development at IBM," in *Proceedings of the 25th International Conference on Software Engineering, May 3-10, 2003, Portland, Oregon, USA* (L. A. Clarke, L. Dillon, and W. F. Tichy, eds.), pp. 564–569, IEEE Computer Society, 2003.

[29] M. Aniche, *Effective Software Testing: A developer's guide*. Simon and Schuster, 2022.

[30] N. Nader-Rezvani, Nader-Rezvani, and McDermott, *An Executive's Guide to Software Quality in an Agile Organization*. Springer, 2019.

[31] K. Bjerke-Gulstuen, E. W. Larsen, T. Stålhane, and T. Dingsøyr, "High level test driven development - shift left," in *Agile Processes, in Software Engineering, and Extreme Programming - 16th International Conference, XP 2015, Helsinki, Finland, May 25-29, 2015, Proceedings* (C. Lassenius, T. Dingsøyr, and M. Paasivaara, eds.), vol. 212 of *Lecture Notes in Business Information Processing*, pp. 239–247, Springer, 2015.

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

[33] A. Tarlinder, *Developer testing: Building quality into software*. Addison-Wesley Professional, 2016.

[34] A. Murphy, B. Kelly, K. Bergmann, K. Khaletskyy, R. V. O'Connor, and P. M. Clarke, "Examining unequal gender distribution in software engineering," in *European Conference on Software Process Improvement*, pp. 659–671, Springer, 2019.

[35] K. Schwaber and M. Beedle, *Agile software development with Scrum*, vol. 1. Prentice Hall Upper Saddle River, 2002.

[36] H. Hemmati, "How effective are code coverage criteria?," in *2015 IEEE International Conference on Software Quality, Reliability and Security, QRS 2015, Vancouver, BC, Canada, August 3-5, 2015*, pp. 151–156, IEEE, 2015.

[37] M. Ivanković, G. Petrović, R. Just, and G. Fraser, "Code coverage at google," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2019, (New York, NY, USA), p. 955–963, Association for Computing Machinery, 2019.

[38] V. Blondeau, A. Etien, N. Anquetil, S. Cresson, P. Croisy, and S. Ducasse, "What are the testing habits of developers? A case study in a large IT company," in *2017 IEEE International Conference on Software Maintenance and Evolution, ICSME 2017, Shanghai, China, September 17-22, 2017*, pp. 58–68, IEEE Computer Society, 2017.

[39] S. M. Melo, Í. Santos, P. S. L. Souza, and S. R. S. Souza, "A survey on the practices of software testing: a look into brazilian companies," *J. Softw. Eng. Res. Dev.*, vol. 10, pp. 11:1–11:15, 2022.

[40] S. Deterding, D. Dixon, R. Khaled, and L. E. Nacke, "From game design elements to gamefulness: defining "gamification"," in *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, MindTrek 2011, Tampere, Finland, September 28-30, 2011* (A. Lugmayr, H. Franssila, C. Safran, and I. Hammouda, eds.), pp. 9–15, ACM, 2011.

[41] T. Fulcini, R. Coppola, L. Ardito, and M. Torchiano, "A review on tools, mechanics, benefits, and challenges of gamified software testing," *ACM Computing Surveys*, 2023.