An instrument to measure factors that constitute the socio-technical context of testing experience

Mark Swillus*, Carolin Brandt† and Andy Zaidman‡
Delft University of Technology
The Netherlands
Email: *m.swillus@tudelft.nl, †c.e.brandt@tudelft.nl, ‡a.e.zaidman@tudelft.nl

Abstract—We consider testing a cooperative and social practice that is shaped by the tools developers use, the tests they write, and their mindsets and human needs. This work is one part of a project that explores the human- and socio-technical context of testing through the lens of those interwoven elements: test suite and tools as technical infrastructure and collaborative factors and motivation as mindset. Drawing on empirical observations of previous work, this survey examines how these factors relate to each other. We want to understand which combination of factors can help developers strive and make the most of their ambitions to leverage the potential that software testing practices have. In this report, we construct a survey instrument to measure the factors that constitute the socio-technical context of testing experience. In addition, we state our hypotheses about how these factors impact testing experience and explain the considerations and process that led to the construction of the survey questions.

I. INTRODUCTION

Software testing practices, which we define as the systematic usage of software development tools to automate the verification process of software, is an effective way to prevent unexpected failures of software systems which can have a detrimental effect on people and society. For example, in 2024, a software bug that co-occurred with a software testing system bug [1], grounded flights at airports around the world and even disrupted hospital operation [2]. Many researchers and practitioners also recognize benefits that go beyond failure prevention [3]. Testing can facilitate software development processes and is considered as a core component in software development methodologies like extreme programming [4]. Given its impact on both software quality and the development process, academics have been encouraging research on software testing practices for more than 40 years. And research in this area continues to be exciting. Recent advancements in the field of AI, especially with the usage of large language models for code generation have provoked new approaches to testing.

Today, there is a vast ecosystem of concepts, tools, and approaches for software testing. Given this diversity, one would think every developer is able to integrate testing practices into their projects in a way that suits their needs. However, research into testing experience (TX) paints a different picture. Developers demonstrate emotional responses to testing [5], seem discouraged from meeting their own testing ambitions [6], and even perceive it as a daunting task [7]. Despite the widely recognized benefits of testing and the availability of tools and techniques for almost every use case, it is still seen as optional. Especially when time is short and deadlines are near [8]. As researchers based in the Netherlands, we find the comparison to helmet use among Dutch cyclists fitting: Studies have demonstrated that helmets save lives time and again [9]. The advantages and importance of helmet use are clear, and yet most Dutch cyclists, confident in their habits and environment, choose to ride without. Further, in bordering countries like Germany, where cultural and infrastructural factors differ, helmet usage is more common. The reasons for not wearing helmets in the Netherlands, just like the choice of developers not to test, we argue, are not of a purely objective or technical nature.

Both phenomena can only be understood by considering the broader context in which they occur. In the case of cycling, this broader context concerns practical aspects like infrastructure or road safety, and personal concerns like the leisureliness, confidence, and joy with which people use their bicycles. Our prior work explores the broad context that influences testing decisions, and we find that in the case of testing, practical aspects and personal concerns play a role as well [10]. A combination of various technical and non-technical factors, such as the availability of testing infrastructure (tools and test automation) and the presence of a *testing culture* seem to play an important role. Adoption and adaptation of testing strategies does not seem to be a linear process. Rather, it can better be described as a collective pushing of ambitions, ideas, and technical possibilities through a filter of available materials and techniques. In this report, we continue our investigation of this entanglement. We first propose hypotheses about the conditions under which testing is enabled or inhibited, and then present a survey instrument with which we are going to test those hypotheses. By publishing our research design and survey instrument ahead of data collection, we demonstrate transparency and methodological rigor. Publishing preconceptions and intentions in advance precludes practices like HARKing (hypothesizing after results are known), which can compromise the integrity of research projects and the validity of findings.

A. Pre-registration of hypotheses and research design

Studies like the recent work of Cologna and Mede [11], who investigate and discuss public trust in scientists, echo what can be observed in contemporary public discourse and policymaking: if a fraction of society loses its trust in scientific work, the consequences, like in the case of COVID-19, can be disastrous [12]. According to Cologna and Mede [11], the majority of people still consider science an important foundation for change. However, rapid developments in artificial intelligence have raised new concerns about the integrity of research and the reliability of research findings [13]. In the context of these developments, we are motivated to demonstrate transparency, rigor, and integrity in our work. Engaging in practices promoted by open science initiatives is one way to act on that motivation. By making the research process and results more accessible and transparent, open science initiatives improve reproducibility and foster greater trust in scientific work. In the discipline of software engineering, this is often done by publishing replication packages, which include datasets and source code when the results of a project are published. However, transparency can begin earlier with the publication of research design and instruments, before data is collected and analyzed. For example, survey instruments can be shared prior to data collection to allow other researchers to assess or reuse them in different contexts, independent of the results of the research project they were constructed for (see [14]). In line with this practice, we are publishing the present report, which includes the survey instrument we intend to use before collecting and analyzing data.

In empirical software engineering research, the publication of research designs is becoming more common. Some conferences and journals now offer registered reports tracks [15], [16], where the study design undergoes a peer-review process before data is gathered. Early feedback and transparency help improve methodological rigor, and in many cases, this feedback also comes with a commitment to publish the study regardless of its results. On the one hand, this discourages practices like HARKing (hypothesizing after results are known), which can compromise the validity of findings. On the other, it encourages the publication of negative results, which are often undervalued but critical to advancing knowledge in the field. Not being pressured to produce positive outcomes to secure publication, researchers are more likely to uphold the scientific imperative of disinterestedness.

By publishing our research design ahead of data collection, we act on our motivation to strive for a more open and trustworthy research culture in empirical software engineering. This report can therefore also be understood as a commitment to publish outcomes independent of their implications for our prior work.

II. METHOD

Our prior exploratory work investigated testing experiences using qualitative research strategies. In [6], we analyze documents to uncover the roots of sentimental attitudes of developers towards testing. Following that, in [10], we analyzed semi-structured interviews to explore the topic further. Data triangulation enables researchers to understand a phenomenon from multiple perspectives. Method triangulation reduces the bias that using a single method to compare different perspectives can introduce [17]. We therefore choose a survey-based study design to deepen our work on socio-technical aspects of software testing. In the following sections, we describe how we construct the survey instrument. We follow Kitchenham and Pfleeger [18], which provides a guideline to create personal opinion surveys. The rest of this section is organized according to the six activities Kitchenham et al. identify in their guideline. As this report is published prior to data collection, we only consider the first four activities, which lead to the construction of a survey instrument and a preliminary plan for its evaluation.

- Setting the objective
- Survey Design
- Developing the survey instrument
- Evaluating the survey instrument

III. SETTING THE OBJECTIVE

Declaring the research objective, including the hoped-for outcomes, is necessary to constrain the scope of questions in a survey [19]. To identify the research objective, we choose a top-down approach [20, p.10], breaking down the broader research question we want to tackle into concrete hypotheses based on prior work.

A. Research Questions

We argue that the many factors influencing developers' engagement with testing often lead to unanticipated outcomes, which can understandably provoke emotions. We align with Rooksby, Rouncefield, and Sommerville [21] who describe testing as a stochastic process: What makes a test plan effective is often the ability to handle various unpredictable contingencies as they arise. In our prior work [10], we identify factors that can give rise to these contingencies, and others that enable developers to manage them. The objective of this study is to further investigate the interplay of those factors:

> **RQ1** How does the interplay of testing conditions shape a software developer's motivation to use testing?
>
> **RQ2** How does the interplay of testing conditions shape a software developer's perception of the value of testing?

Through our preliminary work, we observed that concrete conditions influence how testing is perceived and used, as well as which factors encourage developers to reflect on their testing practices. While we have identified conditions related to developers' motivation to test, we have yet to determine whether there are common patterns or combinations that stimulate or inhibit testing. A key goal of our work is to identify these relationships and examine how they correlate with testing usage and motivation.

> **RQ3** Which sets of conditions are common among developers who are motivated to engage in testing?

To investigate this question, we aim to measure developers' motivation regarding software testing, their self-reported perception of effort spent on testing, and their evaluation of how conditions such as *testing infrastructure*, *complexity*, or a developer's *autonomy* impact their efforts. We then test a set of hypotheses that emerged from our previous qualitative exploratory work.

IV. SURVEY DESIGN

We choose to construct a descriptive cross-sectional survey which aims to discover factors that affect software testing (predispositions) and their relationship with each other. Participants are asked for information at one fixed point in time. The survey therefore provides a snapshot of what developers experience. We implement the survey using a web-based self-administered questionnaire, as this format serves both our need for method triangulation and facilitates simple distribution and coverage of the population we want to study.

Before we approach the implementation of the survey instrument, we conduct a lean literature review to determine how others have investigated the factors we want to measure with our instrument. Reusing tested and peer-reviewed instruments increases the credibility and reproducibility of results.

V. LEAN LITERATURE REVIEW

By reviewing available literature, we establish to what extent the construction of a new instrument is required. Re-using tested and peer-reviewed instruments increases the credibility and reproducibility of results. We also familiarize ourselves with approaches others have chosen to investigate similar phenomena, in order to learn how the many aspects of it can be observed, conceptualized, and measured. Identifying measurable variables through this process is the first step in constructing survey questions. The aim of a lean literature review is not to provide a comprehensive overview of all relevant topics, but rather to scan and scope, optimized to serve its key aims quickly and as easily as possible [22, p.119].

1) Motivation: Motivation of software developers has already been researched for over 40 years, as it has been identified as an important factor impacting software developer productivity. Questionnaires are often used to measure various aspects that relate to motivation [23]. Over the years, many different motivators that affect developers, including recognition, trust, autonomy, and variety of work, have been identified [23], [24]. Different models of motivation of software developers have been proposed [25], and though many different perspectives on motivation have been researched, it remains

a complex topic. As the world is changing, past work on motivation needs re-validation to remain insightful. Unfortunately this is done very little [25]. A more recent study by Verner, Babar, Cerpa, *et al.* [26] about motivation using survey instruments finds that social factors and human needs are important for developer motivation. Developers seem to be motivated by a project manager with good communication skills, in projects where risk is controlled, when the work environment is supportive, In their survey, Verner, Babar, Cerpa, *et al.* [26] measure motivation by asking developers in a self-administered online questionnaire about the motivation of their team members. Similar to Daka and Fraser who asking participants directly: *"What motivates you to write unit tests?"* [27]. Straubinger et al., who investigate opinions of software developers regarding testing, take a different approach because they identify that *motivation* is an overloaded term [28]. Motivation, they argue, is a multi-dimensional concept which needs to be considered from many angles in order to be analysed. They do not directly ask developers about their motivation but instead ask about circumstances of their work that transcend motivation. Aligning with this view, social psychologists consider motivation to be a construct that cannot be recorded or observed directly. Accordingly, measuring motivation in experiments or through questionnaires is considered a non-trivial task. In a guide-line to measure motivation, Touré-Tillery et al. suggest differentiating between outcome-focused motivation to complete a goal, and process-focused motivation (more commonly known as intrinsic motivation), which has less emphasis on the outcome and includes elements such as appropriate means and enjoyment during goal pursuit [29]. The first resembles an attitude of *getting it done*, the latter of *doing it right*. In experiments, indicators for those two aspects can be measured using different means. For example, outcome-focused motivation can be revealed through a subject's positive evaluation of goal-congruent constructs such as means, objects, or persons, and a subject's negative evaluation of goal-unrelated constructs such as distractions. Process-focused motivation is, in this context, revealed by positive evaluation of the process [29].

2) Human factors and software development processes: Several recent studies investigate how human and social factors influence software development practices and the organization of software development projects. Investigating success factors of agile method adoption using a survey instrument, Misra et al. find that technical competency, team size, and planning are not strong factors, but that corporate culture and training initiatives have a significant effect [30]. Instead of measuring success factors like *Corporate culture* using with questions, each success factor was measured using a set of questions that collectively represent the factor. Dybå uses a similar approach in a questionnaire-based study investigating the impact of company size on software process improvement [31]. According to their study, the size of companies in which software is developed influences how well projects can leverage different kinds of knowledge and expertise. To investigate influences on software process improvement, Dybå identifies six key success factors and breaks each factor down into several indicators, each measured using one question. The approach is motivated by the argument that complex concepts like software process improvement success factors can not be reliably measured through simple one-dimensional questions. Instead of measuring a factor with one question, multiple questions are therefore combined to measure each concept [31]. A similar approach is taken by Machuca-Villegas et al., who investigate the perception of human factors that influence the productivity of software development teams [32]. In their survey instrument, they first consider key human factors and then scrutinize each using several questions [33]. Using the perspective of human factors, they find a positive influence of empathy, social interaction, communication, and

autonomy on productivity.

3) Software testing: Opinions and needs of developers: In an attempt to close a gap between academic and practitioner views on software testing, Rafi et al. use a combination of a systematic literature review and a practitioner survey to investigate benefits and limitations of software testing practices [3]. Using their literature review, they derive a set of hypotheses which they test by asking practitioners whether they agree or disagree. As the studies considered in their review mostly focus on technical aspects, the results of their study also primarily concern technical aspects of software testing. For example, their survey demonstrates that the benefits of test automation are related to test reusability, repeatability, test coverage, and effort saved in test execution [3]. Daka and Fraser [27] also investigate testing practices with a focus on established practices and problems. They identify that there is a need to improve the technical capabilities of unit testing, especially in terms of automation. However, they also find that the need for testing is often motivated organizationally, and that a developer's own conviction is a strong factor. Like Rafi et al. [3], Daka and Fraser [27] measure aspects with distinct questions, but to increase the validity of those questions, they use verification questions—questions that measure the same variable using alternate wording. They then disregard responses where answers to the same question diverge. All questions were derived from their research questions, but how exactly those questions were developed is not clearly explained [27].

VI. HYPOTHESES AND VARIABLES

In our prior work [10], we identify eleven conditions that affect developers' testing practices:

- Complexity
- Software Development Process
- Safety & Responsibility
- Business and Application Domain
- Vision
- Resource Usage
- Mandates
- Testing Infrastructure
- Testing Culture
- Community Perspective
- Personal Leanings

We also proposed a theory that conceptualizes software testing as a dualism, composed of ephemeral and material elements that influence each other [see 10, pp.17-22]. To approach the research questions we above, we focus our investigation on the interplay of a subset of above-mentioned conditions. First, we investigate how organizational conditions relate to developers' motivation to use testing practices and the extent to which testing is used in projects. Second, we explore the relationship between complexity and testing infrastructure in relation to testing motivation. Third, we examine how material and ephemeral aspects of testing influence one another.

1) Software Process and Testing Motivation: The introduction or adaptation of software testing methods within an organization can be seen as part of an effort to improve the software development process. Dybå [31] identified that the success of process improvement efforts depends on the correlation of particular organizational factors, including company size. Large companies benefit from structured processes that leverage past knowledge, while smaller organizations benefit more from experimentation and exploration [31]. Our prior research suggests that business context and software development organization influence testing practices as well [10], [21]. Autonomy (reflected in the capacity to experiment) and the exploitation of existing knowledge appear to play major roles. Based on these

findings, we propose the following hypotheses about the relationship between autonomy, knowledge sharing, and company size in the context of testing:

[H1] Organizational factors affect testing practices

    H1.1 In large companies, testing is used more extensively when past knowledge is extensively leveraged

    H1.2 In small companies, testing is used more extensively when new ideas and exploratory approaches are embraced

[H2] Motivation to test (both goal-focused and process-focused) stems mainly from individual conviction and not from organizational factors. However, organizational factors do impact the extent to which testing is used.

    H2.1 Motivation to test is not significantly influenced by organizational factors

    H2.2 The extent of testing is influenced by organizational factors such as business sector, mandates and company size

---

Testing these hypotheses requires measuring the following variables, for some of which measurement methods are described in the literature we reviewed in Section V:

- Company size (`H1`)
- Business sector (`H2`)
- Employee participation [14] and Mandates (`H2`)
- Extent of testing (`H1`)
- Exploitation of existing knowledge [14] (`H2`)
- Exploration of new knowledge [14] (`H2`)
- Motivation (process-focused and goal-focused) [29] (`H2`)

2) Complexity and Testing Infrastructure: The complexity of a software system can influence testing motivation in several ways. When software consists of multiple interacting components, systematic testing can help developers maintain an overview of the project. Testing becomes a way to reduce perceived complexity. However, complexity can also discourage testing if simple approaches are inadequate and the effort required for effective testing becomes overwhelming [6]. We argue that the availability of testing infrastructure and team culture significantly mediate this relationship:

[H3] The presence of usable testing infrastructure increases the process-oriented motivation to test

[H4] The (perceived) complexity of a project impacts motivation to test

    H4.1 Complexity decreases process-oriented testing motivation

    H4.2 Complexity increases process-oriented motivation and goal-oriented motivation if testing infrastructure is present

    H4.3 Complexity decreases process-focused motivation if testing infrastructure is absent

---

Testing these hypotheses requires measuring the following variables, for some of which measurement methods are described in the literature we reviewed in Section V:

- Testing infrastructure (`H3`, `H4`)
- Motivation (process-focused and goal-focused) [29] (`H3`, `H4`)
- Complexity (`H4`)

3) Material and Social Construction of Testing: In prior work[10], we conceptualized software testing as an interplay between material elements (e.g., test frameworks, source code, documentation) and ephemeral elements (e.g., discussions, culture). Developers leave *signatures* in code that reflect their testing values, while discussions and team culture produce *echoes* that foster testing culture. Testing-signatures can be understood as traces in artifacts (e.g., testing infrastructure) that represent developers' knowledge and experience and can spark reflection when encountered. We argue that the presence of this record of experience and knowledge can, when properly exploited by an organization stimulate discussions and reflections on testing.

Testing-echoes are collaborative reflections, of ideas and knowledge in the context of testing. We argue that this collaborative exploration of new ideas and a positive testing culture leads to the extension of testing efforts. In this context, we argue that it is the process of reflection that motivates developers to extend their testing efforts.

We further hypothesize that ambitions to extend testing practices benefit from autonomy, meaning the developer has influence on how the development process is structured.

[H5] Knowledge exploration positively influences the extent of testing efforts

    H5.1 Active discussion and interaction influences the extent to which knowledge exploration impacts testing effort extension

[H6] Knowledge exploitation positively influences the extent to which testing practices are reflected

    H6.1 The presence of testing infrastructure influences the extent of reflection on testing

[H7] When testing practices are discussed interactively, motivation (goal-focused) is higher and adoption of testing practices is greater

[H8] Whether reflection through testing echoes leads to code changes depends on the participant's level of autonomy

---

Testing these hypotheses requires measuring the following variables, for some of which measurement methods are described in the literature we reviewed in Section V:

- Exploration of new knowledge [14] (`H5`)
- Extent of testing (`H5`, `H7`)
- Discussion and interaction (`H5`, `H6`, `H8`)
- Exploitation of existing knowledge [14] (`H6`)
- Testing infrastructure (`H6`)
- Motivation (goal-focused) [29] (`H7`)
- Employee participation [14] and Mandates (`H8`)

A. Summary

We aim to illuminate relationships between organizational, technical, social, and cultural factors that shape software testing practices. The survey instrument we construct to investigate these relationships is designed to deepen our understanding of how and why developers are motivated to adopt testing, while also providing actionable insights to enhance testing processes across diverse development environments. In order to reach our research objective we measure 10 variables:

1) Company size (`H1`)
2) Business sector (`H2`)
3) Extent of testing (`H1`, `H5`, `H7`)
4) Complexity (`H4`)
5) Testing infrastructure (`H3`, `H4`, `H6`)
6) Discussion and interaction (`H5`, `H6`, `H8`)
7) Employee participation [14] and Mandates (`H2`, `H8`)

8) Exploitation of existing knowledge [14] (H2, H6)
9) Exploration of new knowledge [14] (H2, H5)
10) Motivation (process-focused and goal-focused) [29] (H2, H3, H4, H7)

Using a questionnaire to measure those variables we test eight hypotheses and aim to generate insights that enable more effective integration of testing strategies, ensuring that tools and techniques align with developers' needs, goals, and the broader organizational contexts in which they are employed.

VII. IMPLEMENTATION

For the 10 variables we want to measure with a questionnaire, we re-use three key factors of software process improvement in organizations from Dybå [14].

We adapt the questions were needed to fit our focus on software testing. To measure motivation, we use indicators for process-focused and goal-focused motivation as suggested by [29]. To measure goal-focused motivation we construct 5 questions which address the recall and evaluation of goal-related constructs that relate to testing. To measure process-focused motivation we construct 3 questions which address the evaluation of the impact of testing on the process of software development.

Of the other six questions, we measure two variables (company size and business sector) with single-item questions as they are similar to demographical questions and therefore unambiguous. For company size, a range from a set of fixed values can be selected (e.g. 2-10, 11-50, 51-200, 201-500, 501+). Similarly, we use the list from the Global Industry Classification Standard (GICS[1]) from which participants can select the Industry sector that best fits their organization.

Both our prior work and the literature review presented above indicate that the remaining variables we want to measure are inherently complex and multi-faceted. To reliably measure complex variables, we therefore follow an approach similar to [14]. Instead of using a single question to measure the four remaining, complex variables we use multi-item scales which measure a single variable with multiple representative indicators. The scores (in this context called *scale scores*) of each variable are then calculated through the sum of all respective indicators. For each factor we want to measure, we therefore construct three or more indicators. For each indicator we then construct one question.

We derive indicators for each variable with our prior work on software testing experience [10]. We then construct questions by turning each indicator into one unambiguous statement. For example, to measure *complexity*, one indicator is a developers' *perception of technical complexity of the project*. From this indicator, we derive the following statement, which is answered using a 5-point Likert scale ranging from strongly agree to strongly disagree: *I consider the software project as complex: it is large and consists of many components which interact with each other*. We choose 5-point Likert scales with the same options for all questions in our survey instrument, except for demographic questions. Table I contains all factors and corresponding questions.

---

[1] 🔗 msci.com/our-solutions/indexes/gics

TABLE I: Multi-scale variables and corresponding questions

Variable	Indicator	Question
Extent of Testing	Extent to which testing is used as an individual activity to develop software	I am making extensive use of software testing techniques in my daily software development activities
	Extent to which testing is used by the project to develop software	The project for which I develop software is making extensive use of software testing
	The organization supports and embraces software testing	The organization in which the project is embedded promotes the usage of software testing practices
	Degree to which testing is mandated	Testing is strictly required when contributing to the software I develop
Complexity	Perception of technical complexity of project	The software project I contribute to is large and consists of many components which interact with each other
	Usage of standard tools	We mainly use common, standardized technologies when we develop, test, build and distribute the software we develop
	Degree to which best practices can be used	Large parts of the code base use common patters that can be learned in books or online.
Infrastructure	Extent of tool support for testing goals	Our project is built upon strong testing frameworks which include elements like test suites and development tools
	Ability to use testing with respect to resources available	I (would) have enough time to expand and improve our project's testing framework
	Importance of existing source code for the extensions of test suites	I often reuse existing source code when developing new tests
	Extent of explicit guidelines for testing in the project	Guidelines and expectations for testing are clearly defined and explicitly documented
Discussion and Interaction	Degree to which discussions translate to contributions	Conversations about testing usually lead to actual improvements in our testing efforts
	Degree to which testing is learned through mentoring	I learned a lot about testing through colleagues who taught me how to approach it
	Extent of implicit guidelines for testing in the project	I mainly know what is expected of me in terms of testing by interacting and talking with other developers
	Extent to which testing culture can establish something as untestable	There is a shared understanding in our project that parts of the software we develop are untestable
	Extent to which testing culture influences the perceived difficulty of testing in project	In our team testing tasks are considered to be straight forward and easily done
Employee Participation	Extent of employee involvement in decisions that should best be done at their own level	Software developers are involved to a great extent in decisions about the implementation of their work
	Extent to which employees contribute with improvement proposals	Software developers are actively contributing with proposals to adapt testing strategies
	Extent of developer participation in the formalization of routines	Software developers are actively involved in creating routines and procedures for testing
	Extent of ongoing dialog and discussion about software development	We have an on-going dialogue and discussion about software development
	Extent of ongoing dialog and discussion about software testing	We have an on-going dialogue and discussion about software testing

Table continued from previous page

Variable	Indicator	Question
Exploitation of Existing Knowledge	Extent to which existing knowledge is exploited	We exploit the existing organizational knowledge to the utmost extent
	Extent of learning from past experiences	We are systematically learning from the experience with prior projects
	Degree to which formal routines are based on past experience	Our routines for software testing are based on experience from prior projects
	Degree of systematization of past experience	We collect and classify experience from prior projects
	Degree of internal experience transfer	We put great emphasis on internal transfer of positive and negative experience
Exploration of New Knowledge	Degree of adaptability to rapid change, increasing complexity and environmental uncertainty	We are very capable at managing uncertainty in the organization's environment
	Extent to which innovation/change is encouraged	In our organization, we encourage innovation and creativity
	Degree of experimentation with new ideas, strategies and technologies	We often carry out trials with new software engineering methods and tools
	Degree of experimentation with new ideas, strategies and technologies	We often conduct experiments with new ways of working with software development methods.
	Ability to question underlying values	We have the ability to questions *established* truths
	Degree of flexibility in task execution	We are very flexible in the way we carry our our work
	Degree of detail in task specification	We do not specify work processes more than what is absolutely necessary
	Importance of matching the variety and complexity of the organization's environment	We make the most of the diversity and interests to manage the variety and complexity of the organizations environment
Motivation (goal-focused)	Extent to which testing is perceived to contribute to software quality goals	Software testing practices enable developers to write better code
	Degree of accessibility and memory for goal-congruent constructs (means, objects, person)	I can name several testing tools and methods that (would) help me to achieve the project's goals
	Degree of accessibility and memory for goal-incongruent constructs (temptations)	I can recall many situations in which testing related work distracts me from getting the job done
	Degree of positive evaluation of goal-congruent constructs (means, objects, persons)	Testing tools, methods and testing contributions of my colleagues are crucial for the success of our project
	Degree of negative evaluation of goal-incongruent constructions (temptations, distractions)	Unnecessary testing tasks often distract me and hinder my progress towards my actual goals
Motivation (process-focused)	Extent to which software testing contributes to pr.	Software testing increases my productivity
	Positive experience from the process	I enjoy using software development more when I use software testing tools and practices
	Positive evaluation of the process	The use of software testing methods and tools positively impacts my workflow

End of table

Score	Answer Criteria
1	• Not relevant so it should be removed • This item is not clear • This item has no logical relationship with the factor. • This item can be removed without affecting the factor measurement
2	• It must be rewritten • The item requires several modifications or a very large modification in terms of wording or structure • A very specific modification is required for some wording • The item has a moderate relationship with the factor it is measuring • The item is relatively important
3	• This item is relevant • This item is clear with proper semantics and syntax • This item is completely related to the factor being measured • This item is very relevant and must be included

TABLE II: Item assessment criteria for survey instrument evaluation

VIII. EVALUATING THE SURVEY INSTRUMENT

Before turning the instrument into a web-based questionnaire that can be self-administered by participants, all questions and answers should be evaluated. Evaluation servers multiple goals[18]

- Ensure that questions are understandable
- Assess the likely response rate
- Determine the reliability and validity of the instrument
- Test the data analysis techniques on expected outcomes

To evaluate the survey instrument we first conduct an unstructured interview with a software developer who we consider to be an expert on the topic. After the interview in which we discuss the research objective we incorporate the interviewee's feedback into the first draft of our questionnaire and then give our questionnaire the interviewee for a more structured evaluation. We ask them to rate each question on a scale from one to three, ranging from *Not relevant - should be removed* to *relevant - can remain as is* and provide explanations for each answer in free text fields. We provide an extensive description of each point on the evaluation scale, which we take from the work of Machuca-Villegas et al. [33] can be seen in Table II.

After we incorporate this in-depth feedback, we conduct a pilot study for which we recruit a small group of software developers to complete the questionnaire and encourage them to provide feedback about questions using free-text fields. Through feedback from the pilot study we further improve wording and clarity where possible. We also use the data gathered in the pilot study to run preliminary tests for reliability and validity.

REFERENCES

[1] B. Pariseau, *CrowdStrike outage underscores software testing dilemmas*, Jul. 2024. [Online]. Available: https://web.archive.org/web/20250226164043/https://www.techtarget.com/searchsoftwarequality/news/366599/CrowdStrike-outage-underscores-software-testing-dilemmas (visited on 04/09/2025).

[2] E. Pilkington and L. Aratani, *US transportation, police and hospital systems stricken by global CrowdStrike IT outage*, Jul. 2024. [Online]. Available: https://web.archive.org/web/20250306211836/https://www.theguardian.com/technology/article/2024/jul/19/crowdstrike-outage (visited on 04/09/2025).

[3] Dudekula Mohammad Rafi, Katam Reddy Kiran Moses, K. Petersen, and M. V. Mantyla, "Benefits and limitations of automated software testing: Systematic literature review and practitioner survey," in *2012 7th International Workshop on Automation of Software Test (AST)*, IEEE, Jun. 2012, pp. 36–42, ISBN: 978-1-4673-1822-8 978-1-4673-1821-1. DOI: 10.1109/IWAST.2012.6228988. [Online]. Available: http://ieeexplore.ieee.org/document/6228988/ (visited on 12/01/2021).

[4] D. Wells, *ExtremeProgramming.org*, 2009. [Online]. Available: https://web.archive.org/web/20250309052557/http://www.extremeprogramming.org/ (visited on 04/22/2025).

[5] I. Evans, C. Porter, and M. Micallef, "Scared, frustrated and quietly proud: Testers' lived experience of tools and automation," en, in *European Conference on Cognitive Ergonomics 2021*, ACM, Apr. 2021, pp. 1–7, ISBN: 978-1-4503-8757-6. DOI: 10.1145/3452853.3452872. [Online]. Available: https://dl.acm.org/doi/10.1145/3452853.3452872 (visited on 01/08/2024).

[6] M. Swillus and A. Zaidman, "Sentiment overflow in the testing stack: Analyzing software testing posts on Stack Overflow," en, *Journal of Systems and Software*, vol. 205, p. 111804, Nov. 2023, ISSN: 0164-1212. DOI: 10.1016/j.jss.2023.111804. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121223001991 (visited on 07/31/2023).

[7] K. Stobie, "Too Darned Big to Test: Testing large systems is a daunting task, but there are steps we can take to ease the pain.," en, *Queue*, vol. 3, no. 1, pp. 30–37, Feb. 2005, ISSN: 1542-7730, 1542-7749. DOI: 10.1145/1046931.1046944. [Online]. Available: https://dl.acm.org/doi/10.1145/1046931.1046944 (visited on 04/09/2025).

[8] K. Wiklund, S. Eldh, D. Sundmark, and K. Lundqvist, "Impediments for software test automation: A systematic literature review: Impediments for Software Test Automation," en, *Software Testing, Verification and Reliability*, vol. 27, no. 8, e1639, Dec. 2017, ISSN: 09600833. DOI: 10.1002/stvr.1639. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/stvr.1639 (visited on 12/01/2021).

[9] C. M. Büth, N. Barbour, and M. Abdel-Aty, "Effectiveness of bicycle helmets and injury prevention: A systematic review of meta-analyses," en, *Scientific Reports*, vol. 13, no. 1, p. 8540, May 2023, ISSN: 2045-2322. DOI: 10.1038/s41598-023-35728-x. [Online]. Available: https://www.nature.com/articles/s41598-023-35728-x (visited on 02/27/2025).

[10] M. Swillus, R. Hoda, and A. Zaidman, *Who cares about testing?: Co-creations of Socio-technical Software Testing Experiences*, Apr. 2025. DOI: 10.48550/arXiv.2504.07208. [Online]. Available: http://arxiv.org/abs/2504.07208 (visited on 04/22/2025).

[11] V. Cologna and N. G. Mede, "Trust in scientists and their role in society across 68 countries," *Nature Human Behaviour*, Jan. 2025, ISSN: 2397-3374. DOI: 10.1038/s41562-024-02090-5. [Online]. Available: https://doi.org/10.1038/s41562-024-02090-5.

[12] L. C. Hamilton and T. G. Safford, "Elite Cues and the Rapid Decline in Trust in Science Agencies on COVID-19," en, *Sociological Perspectives*, vol. 64, no. 5, pp. 988–1011, Oct. 2021, ISSN: 0731-1214, 1533-8673. DOI: 10.1177/07311214211022391. [Online]. Available: https://journals.sagepub.com/doi/10.1177/07311214211022391 (visited on 04/09/2025).

[13] F. R. Elali and L. N. Rachid, "AI-generated research paper fabrication and plagiarism in the scientific community," en, *Patterns*, vol. 4, no. 3, p. 100706, Mar. 2023, ISSN: 26663899. DOI: 10.1016/j.patter.2023.100706. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666389923000430 (visited on 04/09/2025).

[14] T. Dybå, "An Instrument for Measuring the Key Factors of Success in Software Process Improvement," *Empir. Softw. Eng.*, vol. 5, no. 4, pp. 357–390, 2000.

[15] ICSE, *MSR 2025 - Registered Reports - MSR 2025*, Apr. 2025. [Online]. Available: https://web.archive.org/web/20250409135823/https://2025.msrconf.org/track/msr-2025-registered-reports?#call-for-registered-reports (visited on 04/09/2025).

[16] EMSE, *Registered Reports | Empirical Software Engineering - An International Journal*, 2025-04-09, Apr. 2025. [Online]. Available: https://web.archive.org/web/20250409135914/https://emsejournal.github.io/registered_reports/ (visited on 04/09/2025).

[17] M.-A. Storey, N. A. Ernst, C. Williams, and E. Kalliamvakou, "The who, what, how of software engineering research: A socio-technical framework," en, *Empirical Software Engineering*, vol. 25, no. 5, pp. 4097–4129, Sep. 2020, ISSN: 1573-7616. DOI: 10.1007/s10664-020-09858-z. [Online]. Available: https://doi.org/10.1007/s10664-020-09858-z (visited on 01/30/2023).

[18] B. A. Kitchenham and S. L. Pfleeger, "Personal Opinion Surveys," en, in *Guide to Advanced Empirical Software Engineering*, Springer London, 2008, pp. 63–92, ISBN: 978-1-84800-043-8 978-1-84800-044-5. DOI: 10.1007/978-1-84800-044-5_3. [Online]. Available: http://link.springer.com/10.1007/978-1-84800-044-5_3 (visited on 03/04/2025).

[19] B. A. Kitchenham and S. L. Pfleeger, "Principles of survey research: Part 3: Constructing a survey instrument," en, *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 2, pp. 20–24, Mar. 2002, ISSN: 0163-5948. DOI: 10.1145/511152.511155. [Online]. Available: https://dl.acm.org/doi/10.1145/511152.511155 (visited on 02/11/2025).

[20] J. Linåker, S. Sulaman, M. Host, and R. de Mello, *Guidelines for Conducting Surveys in Software Engineering*. Lund University, 2015. [Online]. Available: https://portal.research.lu.se/files/6062997/5463412.pdf.

[21] J. Rooksby, M. Rouncefield, and I. Sommerville, "Testing in the Wild: The Social and Organisational Dimensions of Real World Practice," en, *Computer Supported Cooperative Work (CSCW)*, vol. 18, no. 5-6, pp. 559–580, Dec. 2009, ISSN: 0925-9724, 1573-7551. DOI: 10.1007/s10606-009-9098-7. [Online]. Available: http://link.springer.com/10.1007/s10606-009-9098-7 (visited on 01/16/2024).

[22] R. Hoda, *Qualitative Research with Socio-Technical Grounded Theory: A Practical Guide to Qualitative Data Analysis and Theory Development in the Digital World*, eng, 1st ed. 2024. Springer International Publishing, 2024, ISBN: 978-3-031-60533-8. DOI: 10.1007/978-3-031-60533-8.

[23] S. Beecham, N. Baddoo, T. Hall, H. Robinson, and H. Sharp, "Motivation in Software Engineering: A systematic literature review," en, *Information and Software Technology*, vol. 50, no. 9-10, pp. 860–878, Aug. 2008, ISSN: 09505849. DOI: 10.1016/j.infsof.2007.09.004. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950584907001097 (visited on 06/24/2021).

[24] A. Franca, T. Gouveia, P. Santos, C. Santana, and F. Da Silva, "Motivation in software engineering: A systematic review update," en, in *15th Annual Conference on Evaluation & Assessment in Software Engineering (EASE 2011)*, IET, 2011, pp. 154–163, ISBN: 978-1-84919-509-6. DOI: 10.1049/ic.2011.0019. [Online]. Available: https://digital-library.theiet.org/content/conferences/10.1049/ic.2011.0019 (visited on 04/10/2025).

[25] H. Sharp, N. Baddoo, S. Beecham, T. Hall, and H. Robinson, "Models of motivation in software engineering," en, *Information and Software Technology*, vol. 51, no. 1, pp. 219–233, Jan. 2009, ISSN: 09505849. DOI: 10.1016/j.infsof.2008.05.009. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0950584908000827 (visited on 04/10/2025).

[26] J. Verner, M. Babar, N. Cerpa, T. Hall, and S. Beecham, "Factors that motivate software engineering teams: A four country empirical study," en, *Journal of Systems and Software*, vol. 92, pp. 115–127, Jun. 2014, ISSN: 01641212. DOI: 10.1016/j.jss.2014.01.008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S016412121400020X (visited on 04/10/2025).

[27] E. Daka and G. Fraser, "A Survey on Unit Testing Practices and Problems," en, in *2014 IEEE 25th International Symposium on Software Reliability Engineering*, IEEE, Nov. 2014, pp. 201–211, ISBN: 978-1-4799-6033-0 978-1-4799-6032-3. DOI: 10.1109/ISSRE.2014.11. [Online]. Available: http://ieeexplore.ieee.org/document/6982627/ (visited on 05/17/2022).

[28] P. Straubinger and G. Fraser, "A Survey on What Developers Think About Testing," in *34th IEEE International Symposium on Software Reliability Engineering, ISSRE 2023, Florence, Italy, October 9-12, 2023*, IEEE, 2023, pp. 80–90. DOI: 10.1109/ISSRE59848.2023.00075. [Online]. Available: https://doi.org/10.1109/ISSRE59848.2023.00075.

[29] M. Touré-Tillery and A. Fishbach, "How to Measure Motivation: A Guide for the Experimental Social Psychologist," en, *Social and Personality Psychology Compass*, vol. 8, no. 7, pp. 328–341, Jul. 2014, ISSN: 1751-9004, 1751-9004. DOI: 10.1111/spc3.12110. [Online]. Available: https://compass.onlinelibrary.wiley.com/doi/10.1111/spc3.12110 (visited on 03/20/2025).

[30] S. C. Misra, V. Kumar, and U. Kumar, "Identifying some important success factors in adopting agile software development practices," en, *Journal of Systems and Software*, vol. 82, no. 11, pp. 1869–1890, Nov. 2009, ISSN: 01641212. DOI: 10.1016/j.jss.2009.05.052. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S016412120900123X (visited on 04/10/2025).

[31] T. Dybå, "Factors of software process improvement success in small and large organizations: An empirical study in the scandinavian context," in *Proceedings of the 11th ACM SIG-SOFT Symposium on Foundations of Software Engineering 2003 held jointly with 9th European Software Engineering Conference, ESEC/FSE 2003, Helsinki, Finland, September 1-5, 2003*, J. Paakki and P. Inverardi, Eds., ACM, 2003, pp. 148–157. DOI: 10.1145/940071.940092. [Online]. Available: https://doi.org/10.1145/940071.940092.

[32] L. Machuca-Villegas, G. P. Gasca-Hurtado, S. M. Puente, and L. M. R. Tamayo, "Perceptions of the human and social factors that influence the productivity of software development teams in Colombia: A statistical analysis," en, *Journal of Systems and Software*, vol. 192, p. 111 408, Oct. 2022, ISSN: 01641212. DOI: 10.1016/j.jss.2022.111408. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0164121222001224 (visited on 04/10/2025).

[33] L. Machuca-Villegas, G. P. Gasca-Hurtado, S. Morillo Puente, and L. M. Restrepo Tamayo, "An Instrument for Measuring Perception about Social and Human Factors that Influence

Software Development Productivity," *JUCS - Journal of Universal Computer Science*, vol. 27, no. 2, pp. 111–134, Feb. 2021, ISSN: 0948-6968, 0948-695X. DOI: 10.3897/jucs.65102. [Online]. Available: https://lib.jucs.org/article/65102/ (visited on 04/10/2025).