Deconstructing Sentimental Stack Overflow Posts Through Interviews: Exploring the Case of Software Testing*

Mark Swillus
Delft University of Technology
The Netherlands
m.swillus@tudelft.nl

Andy Zaidman
Delft University of Technology
The Netherlands
a.e.zaidman@tudelft.nl

***Abstract*—The analysis of sentimental posts about software testing on Stack Overflow reveals that motivation and commitment of developers to use software testing methods is not only influenced by tools and technology. Rather, attitudes are also influenced by socio-technical factors. No prior studies have attempted to talk with Stack Overflow users about the sentimental posts that they write, yet, this is crucial to understand their experiences of which their post is only one fragment. As such, this study explores the precursors that make developers write sentimental posts about software testing on Stack Overflow. Through semi-structured interviews, we reconstruct the individual experiences of Stack Overflow users leading to sentimental posts about testing. We use the post as an anchor point to explore the events that lead to it and how users moved on in the meantime. Using strategies from socio-technical grounded theory (STGT), we derive hypotheses about the socio-technical factors that cause sentiment towards software testing.**

I. Introduction

We already know for over 40 years that software testing is one of the most pragmatic mechanisms by which we can ensure the quality of the software artefacts that we engineer [1]–[4]. In the light of the unquestionable growing impact that software and software supported devices are having on our daily lives, the role of software testing becomes ever more important. However, to this day there is a schism between widespread recommendations for software engineering practice and our knowledge of how software testing *actually* happens. The urgency to solve this conflict was also signalled by others with a call to arms to better understand the testing process [5], [6]. We have recently seen studies emerge that have observed how software developers test. Beller et al. [7] have investigated when and how developers write test cases in their Integrated Development Environment. They observed that around 50% of the studied projects do not employ automated testing methods at all. But they also found out that for almost all cases testing happens far less frequently than developers estimate. If testing is truly considered a last line of defense against software defects, we need to understand why developers *do* or *do not* engineer and execute test cases. We have already seen glimpses of this in literature. Studies have shown that company culture or time pressure leads to cognitive biases during testing [8]–[10], estimations of the time it takes to write test are often inaccurate [7], [11], availability of documentation shapes the development of tests [12], and that the cost/benefit of testing is often unclear [13]. Additionally, Kasurinen et al. [11], Runeson [14], and Daka and Fraser [15] highlight issues with motivating developers to test software: only half of them have positive feelings about testing, and approachability of tools is a major factor. Their research demonstrates that technical and social aspects which affect practitioners' motivation and commitment to test are interwoven. We argue that one needs to consider the interplay of technical and social aspects to truly understand why practitioners do (not) test. As we set out to investigate what influences software engineers when practicing software testing, we therefore go beyond an analysis of technical aspects to reveal socio-technical factors that influence practitioners in their decision-making. In line with the definition of socio-technical systems by Whitworth et al. [16], we understand as socio-technical factors the intertwined technical and social factors that contextualize the creation of software artifacts. *"Social"* includes for example the people, their interaction, company policies and norms. *"Technical"* in this context includes components of the technical infrastructure that enable and support the creation of artifacts and the facilitation of social needs (e.g., tools for communication).

Understanding socio-technical factors of software testing is important as they directly influence the lived experience of practitioners and have an influence on the quality of their produced artifacts. Emotional attitudes and unhappiness for example, which are very likely be connected to these factors can have a detrimental effects as they can for example lead to process divergence [17]. In the scope of an exploratory study we have already identified factors that are linked to attitudes about testing [18]. In [18], we have qualitatively analyzed 200 posts and find that practitioners who ask their questions on Stack Overflow in a sentimental way, show aspiration or report discouraging circumstances. As *sentimental* we consider expressions that indicate emotional arousal of a person or statements that reflect opinionated views and

* Accepted at the CHASE 2023 Registered Reports Track

subjective, sometimes judgemental perspectives. Both positive and negative posts on Stack Overflow show that sentimental questions are often asked to discover new approaches or to reflect on practices. Through our analysis we were able to pinpoint reoccurring issues that are connected with negative and positive sentiment. For example, like Pham et al. we find that project complexity seems to be a factor that influences decision making and ignites attitudes about software testing [19]. We investigated *how* software engineers express sentiment about testing. With the knowledge we gained we take the next step, asking *why* software engineers express sentiment by illuminating the larger context of posts that we analysed. In this study we explore this context by interviewing authors, reconstructing the individual experience leading to sentimental posts. We argue that deconstructing posts in this way gives us insights into the lived experience of software developers and their perception of software testing. Analysing these experiences and perceptions helps us to understand why developers decide (not) to test. We approach the reconstruction of experiences by first investigating how and why sentimental posts – in which authors ask their questions in a subjective or emotional way – come about. Together with the authors, we aim to explore events and circumstances that are related to their sentimental post.

RQ1: Why are Practitioners Sentimental?

RQ1.1 Which events and circumstances lead software engineers to post about testing on Stack Overflow?
RQ1.2 What influences them to become sentimental?

By analysing what practitioners tell us about their experience, we want to identify factors that influence motivation, adoption and commitment to testing practices.

RQ2: Motivation and Commitment to Test

Which factors affect software developers who ask sentimental questions on Stack Overflow (not) to use systematic software testing?

In our former work that investigated how practitioners express themselves sentimentally on Stack Overflow [18] we developed hypotheses that approach **RQ2**. To guide our investigation into **RQ2**, we consider these hypotheses while remaining open to other theoretical directions. In section II we present further details regarding the hypotheses. A more detailed elaboration can be found in [18].

Hypotheses to Guide Interviews for RQ2

$\mathbf{H}_1$ The motivation to test depends on:
- $\mathbf{H}_{1.1}$ The style of project management (socio-technical).
- $\mathbf{H}_{1.2}$ The approachability of testing with best practices (technical).
- $\mathbf{H}_{1.3}$ How peers communicate its use and value (social).

$\mathbf{H}_2$ If developers have never experienced the value of testing in larger, more complex projects, they will not be inclined to test.

$\mathbf{H}_3$ Adoption or learning of testing in complex projects provokes negative attitude towards it

$\mathbf{H}_4$ Adoption of or change in software testing is inspired and determined by shared, social experiences, and not so much by tools and technology.

Finally, we want to understand how the attitude towards testing and the way in which it is carried out changes over time as participants do (not) gain knowledge and confidence about software testing approaches through experience. By analysing and comparing interviews we want to identify the possible different trajectories that software developer can take as they change roles and responsibilities regarding the development of software and how it influences they way in which they perceive testing.

RQ3: Evolving Attitudes About Testing

How does the subjective perception and experience of software testing change over time?

In semi-structured interviews we use the Stack Overflow post of the interviewee as a chronologic anchor point. We explore the events that lead to it, and reflect together with the participants how they moved on in the meantime. By setting our analysis of posts into contrast with the perspectives that participants explain to us in interviews we derive hypotheses about socio-technical factors that cause sentiment towards software testing.

II. Hypotheses

Whether practitioners of software engineering have a positive or negative attitude towards testing depends on many individual factors. Efficiency in testing, and the willingness to systematically test software is not only a matter of tools and technology. In our study of posts on Stack Overflow [18] we find that a confrontation with challenging testing scenarios can under some circumstances cause negative feelings. We know that developers distance themselves from tasks to which their unhappiness relates [17]. The confrontation with challenging scenarios can therefore lead to withdrawal from testing, potentially resulting in process deviation and reduced code quality. However, we also see that challenges can increase the motivation or ambition of practitioners in the case of software testing. Creativity and being able to make a difference can make the engineering of test suites worthwhile, even when

challenges arise. Whether engineering of sophisticated test cases leads to increased motivation or withdrawal we therefore argue, highly depends on context.

Additionally Meyer et al. [20] found out that on good workdays, developers make progress and create value for projects they consider meaningful. On good days, they spend their time efficiently, with little administrative work, and without facing infrastructure issues; what makes a workday typical and therefore good is primarily assessed by the match between developers' expectations and reality. We argue that hindrances to engineers who are working on test cases, created for example by infrastructure issues, overly complicated development environments, or **constraints introduced through the project management style affect practitioners' attitude towards testing ($\mathbf{H}_{1.1}$)**. We hypothesise that **practitioners are motivated** to invest their time into testing, **when best-practices** or examples from documentations **can easily be applied ($\mathbf{H}_{1.2}$)**. Even more so if they and their **peers communicate the value that systematic testing can bring to projects ($\mathbf{H}_{1.3}$)**.

Accordingly, practitioners who already consider testing a valuable building block of software engineering are ambitious and aspirational about it. We further hypothesize that **the value of testing is not evident to practitioners that never experienced its benefits in larger, complex projects ($\mathbf{H}_2$)**. Observations of Pham et al. [19] seem to support this hypothesis. They identified that novice developers adjust their testing effort according to the perceived complexity of code. A project has to be complex to warrant testing to be beneficial. We suggest that there is a huge potential for negative experiences here. Practitioners only start to test when they perceive a project as complex enough, but in those cases testing in not easily approachable anymore. Complexity we argue causes unexpected behaviors and makes best-practices hard to apply. If not supported by more experienced peers within a project, **the adoption of testing in complex projects can provoke negative feelings and potentially a consequential withdrawal from testing ($\mathbf{H}_3$).** Pham et al. [19] even report that some students develop an anxious attitude towards testing while they learn it.

Positive feelings or ambitions mentioned in posts on Stack Overflow are often self-aroused for example through engagement with *inspiring* resources like books or blogs. Daka and Fraser also identified that peer pressure is only rarely mentioned as a motivating factor to write unit tests; the driving force for a developer to use unit testing is supposedly their own conviction [15]. We contend however that project specific non-technical factors like the knowledge of testing within a team and the way in which developers contribute to the project do play an important role. Changing of attitude about testing, so we hypothesize, may be stimulated by the adoption of new testing tools, but more crucially, **change is inspired by human and social interaction and determined by *shared* experiences ($\mathbf{H}_4$)**.

III. Participants and Dataset

In a prior study we qualitatively analyzed 200 Stack Overflow posts about software testing [18]. Stack Overflow posts are living documents which are edited by their authors and moderators and extended with comments sometimes even a decade after they were created. We analyzed if and how practitioners express sentiment about software testing in posts and what the circumstances are that practitioners describe when they express sentiment. We considered posts to be sentimental when questions are asked in an emotional, subjective way, demonstrating that their author is in some way aroused, for example when authors indicate an aversion or attraction to testing. The replication package which contains all posts and the output of our work is publicly accessible [21]. Of the 200 posts that we manually analysed, 108 turned out to be sentimental, reflecting negative, positive or both attitudes. Based on our analysis we constructed 22 focused codes and four analytical categories. Subsequently, we assigned codes and categories to all posts. This has enabled us to work with the dataset systematically by comparing data and establishing connections between underlying themes in the dataset. Coding and categorization illuminate what practitioners write about when they are sentimental, how they express their sentiment. Our exploratory work also already explores the reasons for sentimentality in posts.

For our interviews we recruit subjects from the list of authors who wrote the posts we analyzed. During interviews posts serve as chronological anchor points to explore the events leading to them and how attitudes, experience and other factors have changed in the meantime. Beyond serving as a chronological anchor point, posts provide us thematic entry points to explore details of individual circumstances. By deepening our understanding of practitioners' lived experiences through interviews we will refine the preliminary findings of our former work. We for example argue that the practitioner writing the following post has an aspirational attitude but expresses both a positive and negative sentiment about testing. *"I'm refactoring one big complicated piece of code [...]. So, I need to write a unit test [...]. After googling I came up with 2 ideas [...]. Am I missing some silver bullet? Possibly, DBUnit is the tool for this?"* We hypothesise that even when aspirational, practitioners can develop negative attitudes because they face their ambiguities about testing too late. They get motivated to use testing only when the complexity of their project has reached a threshold that is very hard to overcome without experience in testing. In an interview we could for example explore the post mentioned above further. Focusing the interview on the circumstances that required refactoring of the code and why the necessity of testing arose in this context will allow us to put our analysis into a bigger context. In addition to the refinement of our preliminary work, we thereby explore new vantage points for theory construction to answer **RQ1–3**.

As we approach data collection and analysis within the framework of socio-technical grounded theory (STGT) [22],

incorporating strategies from constructivist grounded theory, we follow Charmaz' recommendations not to pre-determine the sample size that is required for us to reach theoretical saturation [23]. Instead, using theoretic sampling we alternate between data collection and data analysis to deepen, refine and test the construction of analytical categories, codes or interpretive theory. We will therefore continue to conduct interviews until our analysis reaches a point at which additional data does not provide new insights in the form of codes or categories anymore. As we approach data gathering and analysis in this iterative way, we continuously recruit new subjects and contact subjects again for follow up interviews if we realize that more details about concrete aspects are needed.

Summarized, the dataset that we will construct and make available in form of a replication package will contain the following artifacts:

Artifacts to be Published

Anonymized interview transcripts
- Initial interviews
- Follow-up interviews

REFI-QDA[a]. file containing analysed dataset
- Coded transcripts
- Categorization
- Contextual information (analytical memos)

Codebook
- Description of codes
- Inclusion and exclusion criteria
- Examples for application of codes

[a]REFI-QDA is an open standard that enables interoperability between qualitative data analysis software: QDASoftware.org

IV. Execution Plan

At the core of our study we carry out and analyse semi-structured interviews with software developers that we recruit from sentimental Stack Overflow posts. The analysis of these interviews gives us insights into the lived experience of software developers with a particular focus on how they perceive and experience software testing. Steps (1) to (12) in Figure 1 illustrate how we prepare, conduct and analyse interviews to systematically infer knowledge from insights that developers give us. The circular pattern in Figure 1 which starts with *theoretic sampling* (9) reflects the research methodology of Socio-technical Grounded Theory (STGT) [22] which we use in our study. Iteratively it leads us back to the analysis of additional, focused interviews (7) and their incorporation into emerging codes, categories and hypothesis until we hopefully reach a point of saturation at which we report our findings and possibly an interpretive theory that explains observed phenomena.

A. Recruiting

All contributions to Stack Overflow are openly accessible under the creative commons license, which makes Stack Overflow very accessible for data-mining. The platform does however not provide a way to contact users directly. We therefore first need to identify users which we are able to recruit via email (2). Considering users' right to privacy and self-determination, we only recruit users who publish their contact information in a way that implies that they are expecting that this information can be used by researchers. For example, many users reference their personal website which often contains a *Contact* page that indicates how the user can be reached in a way that is not unsolicited. If users do not provide any such references in their Stack Overflow profile, we refrain from using extant data to de-anonymise users as we and others [24] consider practices to be unethical, as they threaten the contextual integrity of participants [25].

Collecting a substantial amount of data compensates the negative effects that misleading or fabricated accounts of participants cause. However, going into details too far by gathering and analysing too much data can lead to descriptivism where abstraction of observed phenomena is no longer possible [23][p. 89]. Accurately pre-determining the number of interviews that we need to conduct is therefore difficult. For our first round of interviews, we aim to recruit 15 participants, selecting 5 negative posts, 5 positive posts, and 5 post with both negative and positive sentiment about testing. During this first round of interviews we will evaluate if those numbers are appropriate.

In order to find recruitable participants we first consider the 200 posts that we have analysed in our former study (1). In case the dataset of 200 posts does not provide us enough potential candidates for recruitment or because of ethical considerations on which we elaborate in Section V, we will consider additional testing related posts from the whole stack overflow data dump, categorizing them by sentiment using sentiment analysis tools (0). Once we find participants that confirm their participation, we invite them to select a date for their interview and provide them additional information about our intentions and the conditions of their participation to ask them for their informed consent (3).

B. Semi-Structured Interviews

We are still in the process of exploring the lived experiences and attitudes of practitioners which includes their use of language. We still need to learn how they refer to and conceptualize *software testing* which is why we do not exactly know how participants will interpret interview questions (yet). To avoid imposing our own ideas and our language onto participants, especially during the first rounds of interviews, we therefore avoid asking questions that are too leading. This is especially important to avoid a confirmation bias for testing of $\mathbf{H}_1$-$\mathbf{H}_4$. Quite the opposite to imposing our theories, we need to be able to follow unanticipated specifics, hints, and views to gather detailed and genuine accounts of participants'. This is fundamentally different from structured interviews in which the researcher asks the exact same question to all participants. To plan and guide our interviews, we therefore base our strategy for semi-structured interviews on Charmaz'

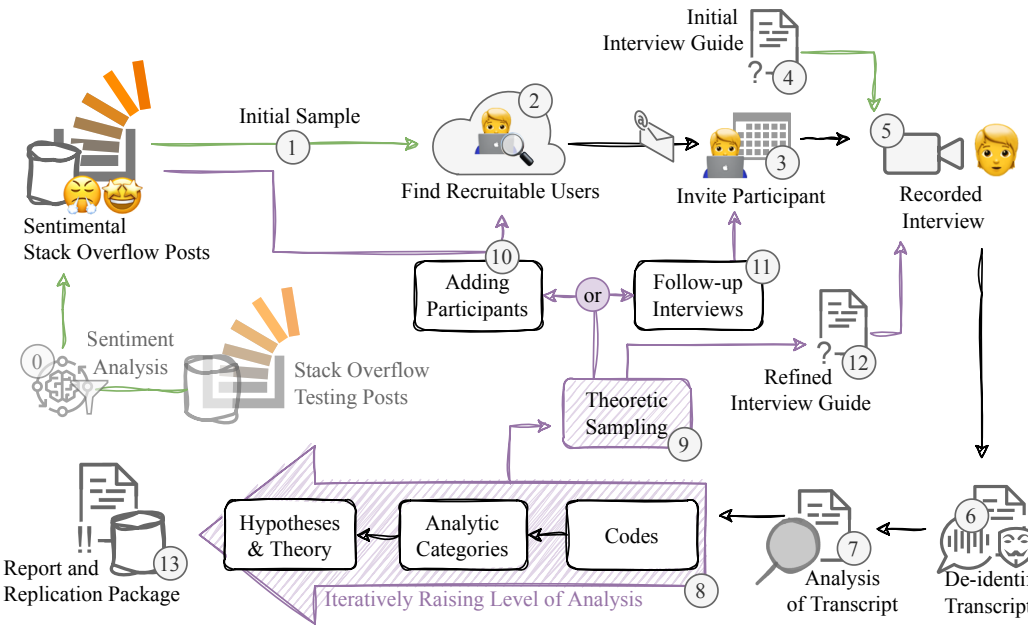

Fig. 1: Execution plan that starts with recruitment of an initial group of participants (3) who are select from a dataset of sentimental Stack Overflow posts (1), and ends with the publication of a report (13) that contains new hypotheses and a theory we construct through systematic, iterative data analysis (8). We consider additional samples from the Stack Overflow data dump (0) to protect participants' integrity and to support our recruiting approach in case of low turnout.

intensive interviewing method which is designed as a tool to explore a person's substantial experience with a research topic. By following Charmaz' guidelines for intensive interviewing we also guard ourselves against confirmation bias regarding the investigation of $\mathbf{H}_1$-$\mathbf{H}_4$. In intensive interviewing the interviewer creates room for the interviewee to tell their story. Instead of dictating the direction of the interview through strictly following a detailed interview guide, the interviewer uses soft control to fill out the details of the story shared by the interviewer [23, p. 69].

According to the method, the initial interview-guide that we create for our first round of interviews will only be used to apply a soft control in interviews which allows us to gently keep participants on topic (4). As our study progresses and our understanding of practitioners' experiences and their use of language grows, we refine the interview-guide, adding pertinent questions that allow us to systematically raise the level of our analysis (12) to answer **RQ1-RQ3**. All interviews are recorded so that the interviewer can focus their attention on the subject, without needing to jot down notes and to avoid mistakes when reconstructing the interviews' content from memory (5). After each interview the recording is transcribed and anonymised, after which the recordings are destroyed to ensure that the privacy of participants is protected (6). Once the transcription process is done we will analyse the transcripts (7).

C. Systematic Data Analysis

Through our qualitative analysis of posts on Stack Overflow we learned that practitioners' sentiments towards software testing are not caused merely by technical aspects [18]. The processes around the topic of software testing is socio-technical in the sense that technical aspects of practice are interwoven with social aspects. One needs to take both into consideration to understand why practitioners become sentimental about testing and why they decide (not) to test the software they develop. To lead our investigation of socio-technical aspects, we employ strategies from Hoda's Socio-Technical Grounded Theory (STGT) [22]. Using codes, categories and the preliminary interpretive theory, that we constructed in our prior study [18], using strategies from STGT's *Basic Stage* for data collection and analysis, we now focus on what Hoda defines as STGT's *Advanced Stage*. Concretely we use the *emergent mode* for data analysis and theory construction. Working in the emergent mode for STGT studies enables the emergence of theory through iterative data collection and analysis. The emergent mode leads to theoretical saturation and results in a mature theory that is grounded in the data that was collected [22, p. 14]. Our epistemological stance with regard to our research questions is that the lived experience of testing practices and the experience of practitioners is highly individual and constructed within a unique complex socio-economic and technical context. We cannot expect that participants fully comprehend all factors that create and form their experience, even less can we expect them to reconstruct all those factors in interviews with us. What they share with us in interviews is a subjective report of what they believe their experience constitutes. With a pragmatist attitude, we do think however that our systematic approach will enable us

to trace factors that affect the lived experience of software engineers when they use software testing. Within the framework of Hoda's STGT we therefore adopt a constructivist epistemology, acknowledging that our work can only be an interpretive translation of the complex lived experience that practitioners describe to us. To reduce the effect that our own ideas have on the study results, we conduct interviews and data analysis with the necessary scientific rigour, by following systematic methods from Charmaz' version of constructivist Grounded Theory [23].

We begin the analysis of transcripts by coding them line by line using *open coding* (8). After the initial round of about 15 interviews is finished, we proceed with additional rounds of coding, summarizing and refactoring codes, also incorporating codes of our former study [18]. Emerging themes, categories and *focused codes* at this point will then guide us in the process of theoretic sampling (9). To reduce the likelihood of a misinterpretation that would pose a threat to the validity of our results, the authors will discuss the interpretation of the data recorded in memos and developed codes, categories and theory and resolve disagreements in a cooperative manner. We will not provide a quantitative analysis of this process of reliability verification as such an analysis would suggest a level of objectivity that we do not want to claim [26]. To deepen our understanding of emerging ideas, we conduct further, more focused follow-up interviews with participants (11) or with new participants (10). Follow-up interviews will enable us to go more into detail with participants while the recruitment of new participants enables us to broaden theory development. We create new interview guides for these interviews to be able to focus on specifics that are relevant to further develop our analysis in order to answer **RQ1-RQ3**. In this way, the iterative process of theoretic sampling enables us to systematically test and extend our hypotheses. To make this process of data analysis and theory construction systematic, we employ methods for *constant comparison* from Charmaz [23] and Saldaña [27]. We compare for example statements and events within one interview, statements from the same interviewee in different interviews, or statements of different interviewees about similar incidents. As we raise the level of analysis, we also compare, on a more abstract level focused codes or analytical categories that were assigned to interview segments. We also incorporate the content of analytical memos that we write during the whole process of data analysis. To facilitate the process of constant comparison for theory construction we use techniques for qualitative research like diagramming [23] and clustering [27].

D. Reporting of Results

The execution plan of our study centers around the iterative process of theoretic sampling and methods for constant comparison which, using the framework of STGT lead to theory development.

We start with a sentimental post of a Stack Overflow user, collect data about the context of that post through interviews and analyse this data. As we increase the level of analysis, we supplement the dataset of anonymised transcripts by assigning codes and categorization and by writing analytical memos. In our report we lead the reader through this process to make transparent how we moved from a low level of analysis at which we construct codes that describe and summarize segments of interviews to a higher level of analysis, that concludes with the construction of a theory. For each level of analysis our report provides concrete examples of application, demonstrating for example how codes were applied to interview segments and how the categories emerged. Apart from making our approach transparent in the report, we present answers to **RQ1-RQ3**, and show how the research questions and the hypotheses $\mathbf{H}_1$-$\mathbf{H}_4$ guided our analysis. We also elaborate on the testing of hypotheses $\mathbf{H}_1$-$\mathbf{H}_4$ and suggest arguments for their rejection, validation or refinement.

Finally, the reporting of the theory which we developed is central to the publication of the results of this study. There is however a difficulty here: it is impossible to foresee the depth and maturity of such a theory, before we start our analysis. We might for example realize that the research population or the data gathering method used in this study is not sufficient to reach theoretic saturation that is required to report a mature theory. A conclusion could be that more triangulation of method or data is needed in order to establish maturity [28]. However, as Hoda emphasizes, reporting of preliminary theories and emerging hypotheses – like we did in our former study [18] – is important to assess and improve the relevance and rigour [22]. The reporting of our results will thus, even in the case that it centers around the presentation of a *preliminary theory* motivate new venture points to investigate socio-technical aspects of software engineering which will help us to better understand and ultimately enrich the experience of software developers.

To stimulate future research and to subscribe to the values of open science, we publish all anonymised data and our research output – as outlined in Section III – openly in the form of a replication package.

V. Ethical considerations

This study investigates and reports an analysis of perspectives of human subjects. We want to ensure that participants are not harmed through our study. Neither directly through the recruitment- or data collection process, nor indirectly through repercussions caused by the publication of our work and its artifacts. We also want to respect the policies of online platforms when we extract data to identify subjects for recruitment. For example, we are carefully considering case by case whether to use email addresses published by users on GitHub profile pages as it is stated in GitHub's accepted-use policy that extracting email addresses to send unsolicited emails is not permitted [29].

To protect subjects from harm we consider their right to privacy and self-determination and follow Nissenbaum's principles to protect contextual integrity [25]. For recruitment, we only consider information that has been made public

by subjects in a way that implies an expectation that the information can be used by researchers to contact them. This includes for example email addresses referenced on the *Contact* page of a personal website which has been linked by the user on their public Stack Overflow profile. We also consider Marwicks's concerns about context collapse [30]. Users on social media websites like Stack Overflow participate and contribute within a specific social context that is largely defined by their expectations of how their contributions are perceived and used. Linking these contexts, for example by establishing a link between activities on Stack Overflow and profiles on Instagram or Twitter through de-anonymization leads to a collapse of this context. Connecting content that individuals contributed in different contexts can cause unexpected and damaging consequences. We therefore de-identify all information that participants provide us before publication. To reduce the lieklihood of the reversal of this de-identification, we won't make transparent which samples of the dataset were considered for recruitment of participants. Additionally, after reviewing the dataset from which we select potential participants, we will evaluate how likely a reversal of de-identification is considering the group- and dataset size. In case we cannot be confident in the de-identification approach at this point, we will take additional measures and adjust our recruiting strategy to reach that confidence.

To seek balance between transparency and protection of subjects' integrity when publishing our results, we will ask participants to review the de-identified transcripts and allow them to remove the parts that they are not comfortable to share before publication.

Our study design was submitted and approved by the privacy team and ethics council of TU Delft.

Acknowledgements

This research was partially funded by the Dutch science foundation NWO through the Vici "TestShift" grant (No. VI.C.182.032).

References

[1] P. H. Carstensen and C. Sørensen, "Let's Talk About Bugs!" vol. 7, 1995.

[2] W. C. Hetzel, *The complete guide to software testing*, 2nd ed. QED Information Sciences, 1988.

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

[4] E. Yourdon, *Managing the system life cycle*, 2nd ed., ser. Yourdon Press computing series. Yourdon Press, 1988.

[5] A. Bertolino, "Software Testing Research: Achievements, Challenges, Dreams," in *Future of Software Engineering (FOSE '07)*, May 2007, pp. 85–103.

[6] M. V. Mäntylä, J. Itkonen, and J. Iivonen, "Who tested my software? Testing as an organizationally cross-cutting activity," *Software Quality Journal*, vol. 20, no. 1, pp. 145–172, Mar. 2012. [Online]. Available: http://link.springer.com/10.1007/s11219-011-9157-4

[7] M. Beller, G. Gousios, A. Panichella, S. Proksch, S. Amann, and A. Zaidman, "Developer Testing in the IDE: Patterns, Beliefs, and Behavior," *IEEE Transactions on Software Engineering*, vol. 45, no. 3, pp. 261–284, Mar. 2019. [Online]. Available: https://ieeexplore.ieee.org/document/8116886/

[8] G. Çalıklı and A. B. Bener, "Influence of confirmation biases of developers on software quality: an empirical study," *Software Quality Journal*, vol. 21, no. 2, pp. 377–416, Jun. 2013. [Online]. Available: http://link.springer.com/10.1007/s11219-012-9180-0

[9] R. Mohanani, I. Salman, B. Turhan, P. Rodriguez, and P. Ralph, "Cognitive Biases in Software Engineering: A Systematic Mapping Study," *IEEE Transactions on Software Engineering*, vol. 46, no. 12, pp. 1318–1339, Dec. 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8506423/

[10] I. Salman, P. Rodriguez, B. Turhan, A. Tosun, and A. Gureller, "What Leads to a Confirmatory or Disconfirmatory Behaviour of Software Testers?" *IEEE Transactions on Software Engineering*, vol. 48, no. 4, pp. 1351–1368, 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9179007/

[11] J. Kasurinen, O. Taipale, and K. Smolander, "Analysis of Problems in Testing Practices," in *2009 16th Asia-Pacific Software Engineering Conference*. IEEE, Dec. 2009, pp. 309–315. [Online]. Available: http://ieeexplore.ieee.org/document/5358706/

[12] M. Aniche, C. Treude, and A. Zaidman, "How Developers Engineer Test Cases: An Observational Study," *IEEE Transactions on Software Engineering*, vol. 48, no. 12, pp. 4925–4946, Dec. 2022.

[13] A. Begel and T. Zimmermann, "Analyze this! 145 questions for data scientists in software engineering," in *Proceedings of the 36th International Conference on Software Engineering*. ACM, May 2014, pp. 12–23. [Online]. Available: https://dl.acm.org/doi/10.1145/2568225.2568233

[14] P. Runeson, "A survey of unit testing practices," *IEEE Software*, vol. 23, no. 4, pp. 22–29, Jul. 2006. [Online]. Available: http://ieeexplore.ieee.org/document/1657935/

[15] E. Daka and G. Fraser, "A Survey on Unit Testing Practices and Problems," in *2014 IEEE 25th International Symposium on Software Reliability Engineering*. IEEE, Nov. 2014, pp. 201–211. [Online]. Available: http://ieeexplore.ieee.org/document/6982627/

[16] B. Whitworth, "The Social Requirements of Technical Systems:," B. Whitworth and A. de Moor, Eds. IGI Global, 2009, pp. 2–22. [Online]. Available: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/978-1-60566-264-0.ch001

[17] D. Graziotin, F. Fagerholm, X. Wang, and P. Abrahamsson, "What happens when software developers are (un)happy," *Journal of Systems and Software*, vol. 140, pp. 32–47, Jun. 2018. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0164121218300323

[18] M. Swillus and A. Zaidman, "Sentiment Overflow in the Testing Stack: Analysing Software Testing Posts on Stack Overflow," Feb. 2023. [Online]. Available: http://arxiv.org/abs/2302.01037

[19] R. Pham, S. Kiesling, O. Liskin, L. Singer, and K. Schneider, "Enablers, inhibitors, and perceptions of testing in novice software teams," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. ACM, Nov. 2014, pp. 30–40. [Online]. Available: http://doi.org/10.1145/2635868.2635925

[20] A. Meyer, E. T. Barr, C. Bird, and T. Zimmermann, "Today was a Good Day: The Daily Life of Software Developers," *IEEE Transactions on Software Engineering*, vol. 47, no. 5, pp. 863–880, 2021.

[21] M. Swillus and A. Zaidman, "Replication Package for Sentiment Overflow in the Testing Stack," May 2022. [Online]. Available: https://zenodo.org/record/6595110

[22] R. Hoda, "Socio-Technical Grounded Theory for Software Engineering," *IEEE Transactions on Software Engineering*, vol. 48, no. 10, pp. 3808–3832, Oct. 2022.

[23] K. Charmaz, *Constructing grounded theory*, 2nd ed., ser. Introducing qualitative methods. Sage, 2014.

[24] N. E. Gold and J. Krinke, "Ethics in the mining of software repositories," *Empirical Software Engineering*, vol. 27, no. 1, p. 17, Nov. 2021. [Online]. Available: https://doi.org/10.1007/s10664-021-10057-7

[25] H. Nissenbaum, "A Contextual Approach to Privacy Online," *Daedalus*, vol. 140, no. 4, pp. 32–48, Oct. 2011. [Online]. Available: https://doi.org/10.1162/DAED_a_00113

[26] N. McDonald, S. Schoenebeck, and A. Forte, "Reliability and Inter-rater Reliability in Qualitative Research: Norms and Guidelines for CSCW and HCI Practice," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, pp. 72:1–72:23, Nov. 2019. [Online]. Available: http://doi.org/10.1145/3359174

[27] J. Saldaña, *The coding manual for qualitative researchers*, 2nd ed. SAGE, 2013.

[28] M.-A. Storey, N. A. Ernst, C. Williams, and E. Kalliamvakou, "The who, what, how of software engineering research: a socio-technical framework," *Empirical Software Engineering*, vol. 25, no. 5, pp. 4097–4129, Sep. 2020. [Online]. Available: https://doi.org/10.1007/s10664-020-09858-z

[29] GitHub, "GitHub Acceptable Use Policies," Apr. 2023. [Online]. Available: https://github.com/github/docs/blob/main/content/site-policy/acceptable-use-policies/github-acceptable-use-policies.md#7-information-usage-restrictions

[30] A. E. Marwick and d. boyd, "I tweet honestly, I tweet passionately: Twitter users, context collapse, and the imagined audience," *New Media & Society*, vol. 13, no. 1, pp. 114–133, Feb. 2011. [Online]. Available: http://journals.sagepub.com/doi/10.1177/1461444810365313