TU Delft

Delft University of Technology

Sentiment overflow in the testing stack

Analyzing software testing posts on Stack Overflow

Swillus, Mark; Zaidman, Andy

DOI
10.1016/j.jss.2023.111804

Publication date
2023

Document Version
Final published version

Published in
Journal of Systems and Software

Citation (APA)
Swillus, M., & Zaidman, A. (2023). Sentiment overflow in the testing stack: Analyzing software testing posts on Stack Overflow. *Journal of Systems and Software*, *205*, Article 111804. https://doi.org/10.1016/j.jss.2023.111804

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.
For technical reasons the number of authors shown on this cover page is limited to a maximum of 10.

The Journal of Systems & Software 205 (2023) 111804

Contents lists available at ScienceDirect

The Journal of Systems & Software

journal homepage: www.elsevier.com/locate/jss

Sentiment overflow in the testing stack: Analyzing software testing posts on Stack Overflow☆

Mark Swillus *, Andy Zaidman

Delft University of Technology, Van Mourik Broekmanweg 6, 2628XE Delft, The Netherlands

ARTICLE INFO

Article history:
Received 31 May 2022
Received in revised form 15 March 2023
Accepted 14 July 2023
Available online 22 July 2023

Keywords:
Stack Overflow
Software testing
Sentiment analysis
Grounded theory

ABSTRACT

Software testing is an integral part of modern software engineering practice. Past research has not only underlined its significance, but also revealed its multi-faceted nature. The practice of software testing and its adoption is influenced by many factors that go beyond tools or technology. This paper sets out to investigate the context of software testing from the practitioners' point of view by mining and analyzing sentimental posts on the widely used question and answer website Stack Overflow. By qualitatively analyzing sentimental expressions of practitioners, which we extract from the Stack Overflow dataset using sentiment analysis tools, we discern factors that help us to better understand the lived experience of software engineers with regards to software testing. Grounded in the data that we have analyzed, we argue that sentiments like insecurity, despair and aspiration, have an impact on practitioners' attitude towards testing. We suggest that they are connected to concrete factors like the level of complexity of projects in which software testing is practiced.

Editor's note: Open Science material was validated by the Journal of Systems and Software Open Science Board.

© 2023 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. 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 artifacts that we engineer (Athanasiou et al., 2014; Carstensen and Sørensen, 1995; Hetzel, 1988; Myers et al., 2012; Yourdon, 1988; Zaidman et al., 2008, 2011). 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. Just consider the year 2017, which has been earmarked "The Year That Software Bugs Ate the World" because of the astonishing software failures that cost the economy $1.7 trillion in 2017 alone (McCracken, 2017). Crucially, Ko et al. (2014) report on software failures that can be directly linked to the loss of 1500 human lives. 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 signaled by others with a call to arms to better understand the testing process (Bertolino, 2007; Mäntylä et al., 2012).

We have recently seen studies emerge that have observed how software developers test. Beller et al. (2019, 2015a,b) 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 (Mohanani et al., 2020; Çalıklı and Bener, 2013; Salman et al., 2022), estimations of the time it takes to write tests are often inaccurate (Beller et al., 2019; Kasurinen et al., 2009), availability of documentation shapes the development of tests (Aniche et al., 2022), and that the cost/benefit of testing is often unclear (Begel and Zimmermann, 2014). Additionally, Kasurinen et al. (2009), Runeson (2006), and Daka and Fraser (2014) 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. Like Prado and Vincenzi (2018) who studied the perspective of developers during the review process of unit tests to build tools that encourage testing, we follow and put the human into the center of attention. This paper sets out to investigate the circumstances that influence software

☆ Editor: Nicole Novielli.

* Corresponding author.
E-mail addresses: m.swillus@tudelft.nl (M. Swillus), a.e.zaidman@tudelft.nl (A. Zaidman).

https://doi.org/10.1016/j.jss.2023.111804
0164-1212/© 2023 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

engineers when engineering tests going beyond technical aspects of the discipline. Like Sharp et al. (2000), we believe that in order to improve the discipline, it is essential to understand the socio-technical world in which software engineering is practiced. For example, software development practices which form social circumstances, like pair programming, are very likely to have an impact on testing. To gain a broad overview of what these circumstances are, we take negative and positive sentiments on the process of automated testing as a proxy. To gather documents which describe the experience of software developers from their point of view, we mine the most popular question and answer platform for software engineers, namely *Stack Overflow* (Baltes and Diehl, 2019).

RQ1

How do software engineers express sentiment about testing on Stack Overflow?

On the Q&A platform Stack Overflow, on which social interaction plays a key role, practitioners ask questions about software development which are answered by a global community of software developers (Mustafa et al., 2022). Others have used the Stack Overflow dataset to investigate technical and non-technical aspects of software engineering. For example, Lopez et al. (2018) have analyzed security questions on Stack Overflow, and provide an overview of the most discussed topics but also discuss the way in which authors discuss security questions. Our goal is to identify factors that affect practitioners and influence adoption of, or attitude towards testing. We hypothesize that an analysis of sentimental content on Stack Overflow not only reveals technical factors that can influence adoption or attitudes, but also descriptions of the social or human context of practice. To identify those socio-technical factors, we deeply examine 200 testing related questions on Stack Overflow instead of analyzing the whole dataset quantitatively. We do not only scrutinize the question asked by the practitioner, but also incorporate answers, comments and the edit history of questions into our analysis. Going beyond an analysis of *questions* about technical issues, we focus on the broader context that causes sentiment in practitioners. We therefore use the term *post* instead of *question* to refer to the documents we analyzed for the remainder of this paper.

RQ2

Which factors affect sentiment of software engineers towards testing practices?

From research done by other authors we know that only a small fraction of posts on Stack Overflow contains strong opinions and emotional statements as they mostly discuss how to use a piece of technology (Lin et al., 2018; Sengupta and Haythornthwaite, 2020). This motivates us to create an emotionally rich subset by filtering the dataset using a semi-automated approach that employs sentiment analysis tools.

To answer both research questions we apply strategies of Hoda's basic stage for socio-technical grounded theory (STGT) (Hoda, 2022) with a constructivist stance as suggested by Charmaz (2014). STGT provides us with a framework to venture into a broad analysis of testing practice, seen not only as a technical phenomenon, but as a phenomenon in which social factors play an essential role. We focus our analysis on the socio-technical dimension of posts on Stack Overflow and show that such an analysis indeed reveals descriptions of social aspects. Our analysis informs about issues that contribute to problems and attitudes towards software testing. More concretely, we analyze the dataset which consists of 200 posts using initial and focused coding and techniques for systematic comparison of posts, codes and memos like diagramming and clustering. Concluding this paper with a presentation of preliminary categories and a preliminary interpretive theory, we motivate consecutive targeted data collection (theoretic sampling) to test and extend our analysis and conclusions. Grounded in the data we analyze, this paper makes the following contributions:

- We discuss preliminary hypotheses which explore stimuli and inhibitors to testing at a socio-technical level
- We present a computer aided approach for qualitative analysis of sentimental expressions in big datasets
- We motivate a research agenda that includes concrete ideas for targeted data collection (theoretic sampling) to develop a mature theory of stimuli and inhibitors of software testing that go beyond tools and technology

2. Background

2.1. Sentiment analysis

Sentiment analysis is the computational study of opinions, sentiments and emotions expressed in text. It essentially tries to infer people's sentiments based on their language expressions. Sentiment *classification* is a widely studied research topic of sentiment analysis that focuses on the classification of opinionated documents as expressing positive or negative opinion (Indurkhya and Damerau, 2010). Automatic classification of sentiment has been applied in various fields of research over the past 20 years as access to vast amounts of written text about various topics have become available through the internet. Already in 1999 Wiebe et al. (1999) worked on a dataset for automatic classification of news articles to identify whether information is being presented as fact or opinion. While sentiment analysis is still being used to analyze media platforms like those of news agencies (Balahur and Steinberger, 2009; Pelicon et al., 2020), its application today also includes platforms on which a wide variety of people contribute content such as social media or internet forums. Here sentiment analysis has been used recently to identify personal attacks or obscene behavior of users (Saeed et al., 2018).

Techniques for sentiment analysis have also been applied in the context of software engineering. Mantyla et al. (2017) analyzed sentiment in comments on the Jira issue tracker to detect burnout among software developers. They calculated sentiment scores for each sentence using a dictionary that contains ratings for the affective meaning of 13,915 English words. Despite their positive results, they have also raised the issue, echoed by others (Jongeling et al., 2017), that general purpose sentiment analysis tools lack precision when applied to the domain of software engineering. Lin et al. (2018) even question the validity of all quantitative studies in software engineering based on sentiment analysis tools as they demonstrate how hard it is to reproduce results. For example, they judge that there is still a long way to go before researchers and practitioners can use state-of-the-art sentiment analysis tools to identify the sentiment expressed in Stack Overflow discussions. To stimulate more research into the direction of sentiment analysis, they published the dataset that was developed in Lin et al. (2018) which contains 1500 annotated sentences. Similarly, to support empirical research in the direction of emotion detection, Novielli et al. (2018) developed a dataset containing 4800 Stack Overflow posts. Motivated by these voices of criticism and encouragement, others then tried to develop tools tailored to the domain of software engineering like Islam et al. who have developed the dictionary-based tool DEVA (Islam and Zibran, 2018), and a machine-learning based

tool called MarValous that focuses on emotion detection (Islam et al., 2019). In that same period the SentiStrength tool, which already existed as a general purpose tool for sentiment classification, was tweaked for an application in the domain of software engineering by Ahmed et al. (2017), who created the tool SentiCR. In 2020 Zhang et al. (2020) address the issue again, comparing the accuracy of this new generation of tailor-made sentiment analysis tools for software engineering with the accuracy that deep neural network architectures, namely transformer models, achieve. They suggest that transformer models like RoBERTa (Liu et al., 2019) are indeed one big step forward on the long way towards reliable results in sentiment classification for software engineering. Finally, in 2021, Lin et al. (2022) summarize the knowledge gained in one decade of research for opinion mining tools. Among other insights into the field they provide a guideline for the selection, usage and evaluation of opinion mining tools for software engineering research.

2.2. Grounded theory

Grounded theory (GT) is an analytic approach used to construct ethnographic knowledge (Deener, 2018). Its framework is made up of data-gathering techniques and strategies to analyze data. What distinguishes GT from other approaches is its iterative nature. While theory development progresses, the GT approach alternates between data collection and analysis to sustain a high level of involvement with the data (Charmaz, 2014).

GT was suggested as an approach for qualitative research by Glaser and Strauss (2010) and has been reinterpreted by different scholars, resulting in the development of different flavors of GT. Flavors of GT differ in details on how to execute techniques and how tightly strategies need to be followed.[1] Crucially, they also rest on different epistemological stances. Where the original *Glaserian GT* takes an objective, positivist stance, *Constructivist GT* proposed by Charmaz, for example, acknowledges the researchers' subjective perspective. Constructivist GT moves away from positivism, incorporating the beliefs and preconceptions of the researcher into analysis. Situating the GT approach into the field of software engineering research, Hoda has recently proposed another flavor of GT. She designed *Socio-technical GT (STGT)* to ease application of GT in her field, where researchers often struggle to understand and apply it (Hoda, 2022). With STGT, Hoda proposes to divide GT into distinct phases. Embracing the iterative nature of GT, STGT encourages exploration in a *Basic Stage* and helps the socio-technical researcher to transition into an *Advanced Stage* of theory development. The separation into those two stages, which are accompanied by lean and focused literature reviews, help the socio-technical researcher to cover epistemological blind-spots. All flavors of GT use comparative- (e.g., *clustering*, *diagramming*), and analytical methods (e.g., *coding*, *memo writing*), that are accompanied by a continuous collection of new data samples (*theoretical sampling*) to saturate emerging categories that describe data and to enable the development of *mature theories* which transparently emerge from the data. Regarding the analysis of documents, which we set out to do in this paper, Charmaz states that GT of documents is able to address not only content but also their audience, production and presentation. Analysis of documents can reveal what and whom they affect, as they do not only serve as records but explore, explain, justify and/or foretell actions (Charmaz, 2014, p. 46).

In this paper we follow Hoda's STGT and present the results of the *Basic Stage* of our STGT study. Publication of emerging results of this exploratory phase is encouraged by both Hoda and Charmaz. GT guidelines describe steps and a path through a long research process. Depending on the task and project at hand, GT invites using those steps flexibly to raise the analysis to the desired level of theory construction (Charmaz, 2014). Within the framework of STGT we use strategies and epistemology from Charmaz Constructivist GT, raising the data analysis of our dataset that we take from Stack Overflow to a *preliminary theory*. We present our work following Hoda's recommendation who states that publication even of partial results is important to receive feedback from both practitioners and the research community to assesses relevance and improve rigor (Hoda, 2022).

2.3. Stack Overflow

Stack Overflow is the most popular question- and answer website for software developers (Baltes and Diehl, 2019). The website has become an important resource that often complements official documentation of software libraries and tools. Its strong presence on search engines, where a link to the website is very often shown on the first page of results when searching for software development related topics, indicates its reach that goes far beyond the 17 million registered users (Barzilay et al., 2013). Studies that often use the official and open Stack Overflow dataset, have underlined the prominence of Stack Overflow by showing for example that 11% of open source software projects on GitHub that were analyzed in a large scale field study contain source code snippets that were copied from Stack Overflow (Baltes and Diehl, 2019). Over 22 million questions that often contain such code snippets were posted by users in a wide range of topics that are related to software engineering since its launch in 2008.[2] Apart from contributions in the form of questions and answers, users are also encouraged to take part in moderation efforts. Up- and down voting, tagging and editing of questions and answers is rewarded with badges, medals and reputation points. Questions on Stack Overflow generate living documents, which are edited by their authors and moderators, updated, and extended with comments sometimes even a decade after they were asked. Questions and their answers can thus take the character of knowledge base articles. Barzilay et al. (2013) even argue that the moderation and reward system has transformed Stack Overflow from a mere Q&A site into a community project that gives users a sense of belonging which not only generates high quality knowledge but also trust in the content that is accumulated. To emphasize these traits of the content on Stack Overflow that goes beyond questions, we refer to the content on Stack Overflow not as questions but as *posts*.

Before taking part in the community by asking a question for the first time, users can take a virtual tour that explicates the goals of Stack Overflow. It is explained here, that Stack Overflow is *"all about getting answers. It's not a discussion forum. There is no chit-chat"*. Furthermore, it asks users to avoid questions that are primarily opinion-based, or that are likely to generate discussion.[3] The platform's focus to avoid *chit-chat* is also reflected in what Vadlamani and Baysal (2020), and Zagalsky et al. (2016) identified as the primary drivers behind contributions. Going beyond a meta analysis of the platform, scholars used Stack Overflow to investigate various aspects of software engineering, including for example the analysis of trends (Barua et al., 2014; Yang et al., 2016) or developers' interests (Lee and Lo, 2017). Similar to our aim, the Stack Overflow data set has also been used

[1] For a more elaborate discussion of the historic development of GT and a complete comparison of its flavors see Charmaz (2014, p. 4) and Hoda (2022, p. 9).

[2] stackexchange.com/sites

[3] stackoverflow.com/tour

to investigate challenges of software developers. Based on the assumption that questions and answers on Stack Overflow cover a wide range of issues, Alshangiti et al. (2019) analyzed questions in a mixed method study to identified challenges of software engineers when developing machine learning applications.

3. Method

To investigate the lived experience of practitioners on Stack Overflow we take a qualitative approach that aligns with Hoda's Socio-Technical Grounded Theory (STGT) (Hoda, 2022). Acknowledging its iterative nature, we focus on what Hoda defined as STGT's *Basic Stage* for data collection and analysis. We take our initial sample from the Stack Overflow data dump, which we analyze using initial and focused coding while we write memos to constantly compare documents, codes and emerging categories. We then present preliminary hypotheses and an interpretive theory that summarizes our findings. Presenting our findings we motivate for the next iteration of our STGT study that leads to the collection of more data (theoretic sampling) to test and extend our findings. As Hoda (2022) suggests, we publish our initial findings to assess the relevance of our work and to receive feedback from the research community. Successive rounds of data collection and analysis in future work can then lead to the development of more mature theories that are valuable for the field.

Our stance with regard to our research questions is that the reality of testing practices and the experience of practitioners in a complex socio-technical environment is highly individual and not reflected by a Stack Overflow post in its entirety. Within the framework of Hoda's STGT we adopt a subjective, constructivist epistemology. Therefore, we follow Charmaz's version of constructivist Grounded Theory (Charmaz, 2014) to provide our interpretation of these complex matters. Despite our awareness of the limitations that an analysis of non-interactive documents has as they can only provide thin descriptions that lack contextual cues (Hine, 2008), we hypothesize that observation and thorough investigation of attitudes and sentiments expressed by practitioners in posts on Stack Overflow can yield valuable insights into practice. Furthermore, we claim that our analysis contributes to a better understanding of socio-technical dynamics in the context of software testing.

To analyze the Stack Overflow dataset for our specific purpose of investigating the sentiment associated with software testing, we first retrieve Stack Overflow posts related to testing. We then use sentiment analysis tools to identify posts that contain negative and positive sentimental expressions. The whole process, which starts with this filtering process of the Stack Overflow data dump (0) and ends with the construction of preliminary hypotheses and an interpretive theory (22), is visualized in Fig. 1. Grounded theory studies usually undergo a phase of piloting and study preparation as a means to verify that the chosen tools like questionnaires or interview questions are appropriately configured and comprehensible to the studies' subjects. As our study is only involving the analysis of non-interactive documents such a verification process is not applicable. Study preparation in our case is thus limited to the extraction of a subset of posts that we take from the Stack Overflow data dump and the configuration of the sentiment analysis tools that we use ((1) to (5)).

3.1. Filtering by tags

The Stack Overflow dataset contained 53,086,328 posts concerning all domains of software development when we obtained it in August 2021.[4] To extract a subset with a size that is appropriate for manual analysis, we filter all posts using a 2-step process that is outlined in this section. As illustrated in Fig. 1, we begin with the full Stack Overflow *Post*-dataset (0) on the left side and end this process with importing post-documents into a CAQDA-software[5] (7) on the right side. To extract posts related to automated software testing, we first filter the dataset using tags. One or more tags are assigned to every post by their authors. The list of tags is then often edited by moderators to facilitate categorization. Tags represent categories that among others include general concepts or methods (e.g., TESTING, TDD), technologies like programming languages (e.g., JAVA, PYTHON), or specific frameworks and tools (e.g., CODECOV, MOCKITO, REACTJS). Posts are usually tagged with multiple, complementary tags (e.g., post *878848* is tagged with 5 tags: JAVA, UNIT-TESTING, IBDC, MOCKING and RESULTSET). Similar to Yang et al. (2016), we utilized a two-step process to extract posts by searching for tags which represent general concepts and methods related to software testing. We first select all posts from the dataset that are assigned a tag that contains the word TESTING, which produces a set of 134,109 posts (1). We choose the term *testing* as it is used as a suggestion on the Stack Overflow platform whenever the tag TEST is used and because the tags TESTING and UNIT-TESTING are the two most prominent tags when searching for *test* using the tag-search.[6] We then manually analyze the list of 13,006 tags that were assigned to those posts and remove tags that were used less than 6 times, or were not directly referring to general concepts of automated software testing (2). The tag CODECOV for example was removed from the list because it only occurred 5 times, and REACTJS was removed as it relates to a programming framework that is not directly related to automated software testing. We also exclude tags that are related to testing but focus on a particular technology or tool (e.g., MOCKITO), as we try to remain testing tool- and development language agnostic. Following this procedure, we have produced a list of 30 tags that all refer to conceptual aspects of automated software testing, like UNIT-TEST, MOCKING, or TDD. Using this list we again extracted posts from the original dataset. We extract all posts that contain at least one tag that is present on the tag list and obtain a set of 147.833 posts (3). Post *878848* which is tagged with JAVA, UNIT-TESTING, IBDC, MOCKING and RESULTSET was for example selected because the presence of tags MOCKING and UNIT-TESTING. We provide the source code of the program that we used to filter posts and the filtered dataset in our replication package (Swillus and Zaidman, 2022, filter-by-tags.zip).

3.2. Filtering by sentiment

We aimed to examine posts deeply instead of quantitatively which limits our investigation to an analysis of a small subset of the 147,833 posts. From research done by other authors we know that only a small fraction of content posted on Stack Overflow contains strong opinions and emotional statements as they mostly discuss how to use a piece of technology (Lin et al., 2018). Sengupta et al. report that only every 10th comment on Stack Overflow expresses some standalone form of emotion (Sengupta and Haythornthwaite, 2020). This motivated us to create an emotionally rich subset by filtering the dataset using a semi-automated approach that employs sentiment analysis to select posts that contain sentimental expressions. Following the advice of Zhang et al. (2020) to not rely on a single tool we used the

[4] archive.org/details/stackexchange

[5] CAQDA = Computer-Assisted Qualitative Data Analysis software; we have mostly used ATLAS.TI, see: https://atlasti.com.

[6] stackoverflow.com/tags

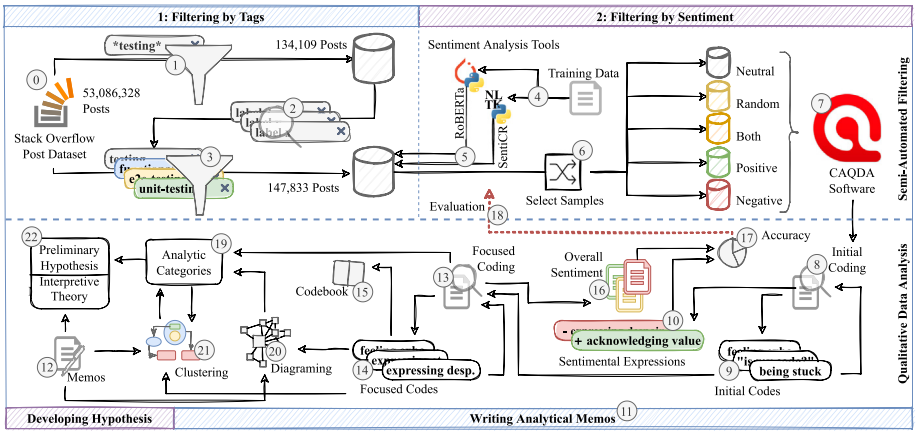


Fig. 1. Filtering and annotating Stack Overflow posts using a semi-automated approach, followed by systematic qualitative data analysis process that leads to the construction of preliminary hypotheses and an interpretive theory.

transformer model RoBERTa (Liu et al., 2019) in combination with the SentiCR tool (Ahmed et al., 2017). We trained both tools with a labeled dataset of Stack Overflow provided by Lin et al. (2018) (4).[7] Their dataset contains 1500 sentences from Stack Overflow posts discussing Java libraries which were manually labeled by the authors with sentiment polarities *positive*, *negative* and *neutral* (Lin et al., 2018). We then used the trained tools, to automatically annotate sentiment polarities to every paragraph of every post of our tag-filtered dataset (5). From this annotated dataset we then randomly extracted posts from 5 categories, using a simple condition for each category (6).

Positive: both tools classified at least one paragraph as positive and none as negative

Negative: both tools classified at least one paragraph as negative and none as positive

Both: both tools classified at least one paragraph as positive and at least one as negative

Neutral: both tools classified all paragraphs as neutral

Random: randomly selected independent of classification

Especially because of concerns raised by Lin et al. (2018) and Jongeling et al. (2017) who state that sentiment analysis tools often do not provide good results for software engineering texts, we used the last two categories *Neutral* and *Random* in a later stage of our analysis to validate our semi-automated filtering approach. We evaluate whether filtering posts with the tools RoBERTa and SentiCR provides a dataset with more sentimental posts than a random selection. We choose paragraphs instead of finer grained sentence-level separation because we hypothesize that a paragraph is more likely to hold a comprehensive and conclusive thought as compared to short sentences that are taken out of context. We argue that sentiment classification done on that level better supports our goal to group posts into categories of positive and negative posts. Contrarily to what we want to achieve, one short and slightly negative remark in a post of an otherwise very positive paragraph, is much more likely to determine a wrong result in a finer grained sentence-level classification. The sentiment analysis tools we used in this study both support the approach of classifying text with multiple sentences. The posts obtained by our semi-automated filtering approach were imported into a CAQDA software (7) that was used to aid all further steps of the data analysis. To avoid bias during our manual assessment of a post's sentiment, we did not include the tool's classification result in those imported posts. Automatically assigned sentiment was not visible to the authors during manual analysis. Initially we analyzed 25 posts from each category (*Random*, *Neutral*, *Positive*, *Negative* and *Both*). We then added another 25 posts from each sentimental category (*Positive*, *Negative* and *Both*), to reach a point at which the analysis of additional posts did not provide new insights or perspectives in the form of new codes. After adding the second batch of 75 posts, and before reaching the 200th post we reached saturation. Posts did not provide new content that did not fit into the categories which had emerged already at this point. We therefore analyzed a total amount of 200 posts. Fig. 2 shows the 20-most occurring tags that were assigned by authors and moderators to those 200 posts. When creating our dataset and selecting the posts, we looked for sentimental discussions about testing without selecting or excluding specific technologies. We do not focus on how practitioners sentimentally evaluate specific tools, e.g., the Java unit testing library *junit*. We instead take a broader, tool agnostic perspective. Nevertheless, to provide context to our dataset, it is interesting to observe which tags (both tool agnostic and tool specific) are assigned to the questions that are included in our dataset. In particular, these tags indicate that our dataset transcends a particular programming language or technology stack. The replication package we provide contains the source code of our implementation of the sentiment analysis pipeline (Swillus and Zaidman, 2022, filter-by-sentiment.zip).

3.3. Data analysis

We employed strategies from grounded theory as recommended by Hoda (2022) and Charmaz (2014) to analyze the filtered Stack Overflow dataset. To begin the iterative process of constructing abstract analytic categories out of which we formulated preliminary hypotheses as illustrated in Fig. 1, we use initial coding (8), applying codes to the dataset line by line in three rounds. We started without any preliminary codes, remaining open to all possible theoretical directions especially during the first coding cycle. In addition to coding posts with gerunds (e.g., *describing* instead of *description*), we use *In-Vivo* codes, which are quotations of what the author of a post wrote in their own language. In-Vivo codes are put in between quotation marks and used whenever authors express themselves in a strong and

[7] Replication package from Lin et al. (2018) containing training data: https://sentiment-se.github.io/replication.zip.

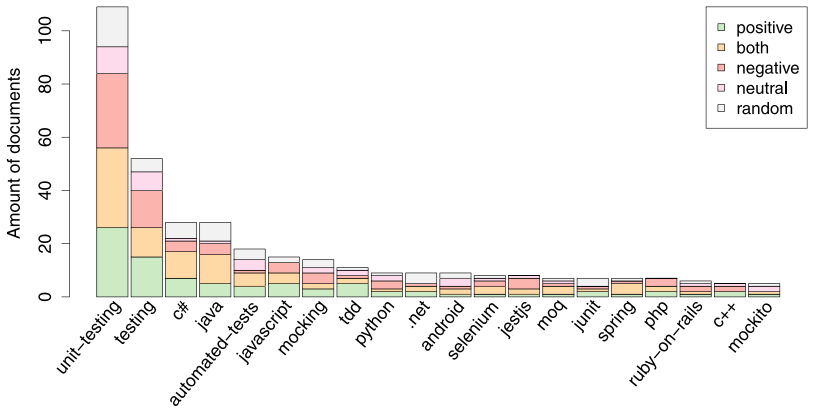


Fig. 2. 20 most occurring tags of the 200 posts we analyzed. The most occurring tags include technology-agnostic tags like TESTING and technology specific tags like JUNIT. Technology specific tags are assigned to posts complementary to the 30 more general tool agnostic tags that we selected during the filtering process.

emotionally rich way (e.g., "IS MY CODE JUST BAD?") (9). In order to provide basic statistical information about the occurrences of negative and positive sentiment in the dataset, we use *magnitude coding* as suggested by Saldaña (2013), adding the symbols + and – to codes where applicable (10). Negative expressions are coded with a minus (e.g., –REFLECTING UNCLEAN APPROACH), and positive sentiments with a plus (e.g., +EMBRACING CHANGE) respectively. We write memos during all stages of our data analysis (11) which we use at a later stage to develop preliminary hypotheses (12).

After three rounds of initial coding, we reassess the significance of all codes to decide which ones contribute most to an incise and complete categorization. As Charmaz suggests, we use this technique to condense the work of the initial coding phase to advance the theoretical direction of the work and to begin with a second cycle of focused coding (13) (Charmaz, 2014). During focused coding cycles we develop focus codes (14) and categorize documents while we construct and continuously refine a codebook (15). In our codebook we spell out details like inclusion- and exclusion criteria, descriptions, and examples for each focused code. Because of suggestions made by Lopez et al. (2019), who have shown that comments on Stack Overflow can reveal expressions of pride and emotional involvement, we also incorporate comments made on Stack Overflow into our analysis. Other additional information obtainable via the Stack Overflow website, like the history of changes made by the original author or a moderator are also considered during focused coding. We understand a post as a potential entryway into a deeper and richer context of an author's question. Details including the sentimental activity in comments, the editing history of a post both by the author and moderators, the reasons for a moderator to close a post, the time it took the community to answer the question or the fact that it was never answered. Where a post offers these details (not all of them do), we capture the information by writing analytical memos. One memo about post *55357595* with the title *Fruitless pursuit* written by one of the authors for example reads:

Memo: *Fruitless Pursuit*

The author of this post did not receive any feedback from the community. But almost a month after posting this question, the author just comments: "Ended up setting up a webpack from the ground up" Which I think indicates that this person has gone through quite some torment. However, they do not express this explicitly.

During the process of focused coding, we also assign a sentiment of positive, negative, both, or neutral to each post. Here the assigned sentiment represents the overall attitude of the author towards testing practices (16). We use both the coding of sentiment (10) and assignment of the overall sentiment (16), to determine the accuracy of the sentiment analysis pipeline (17) and to evaluate its use in our filtering process (18). During the focused coding cycles, preliminary analytic categories became visible to us (19). A large amount of negative posts containing expressions of desperation for example, developed into the category *Discouragement* early on. We refine categories that become visible through the process of coding, using a diagramming technique described by Saldaña (2013) (20). Starting with a code like *Expressing desperation* or a post that creates ambiguity when assigned with a category, we sketch a network of connections to other posts, categories or codes on paper to explore detailed features of the coded dataset from different angles. We then use the clustering strategy as described by Charmaz (2014), grouping posts together and writing memos, concentrating on commonalities and differences among those groups of posts (21). Taking a different perspective each time, we find different explanations for the meaning and context of sentiment expressed by practitioners in posts. We continue the process of analyzing the dataset using these strategies, until they no longer yield new perspectives and we were able to formulate preliminary hypothesis and an interpretive theory that emerged from the process (22).

3.4. Constructing interpretive theory

Synthesizing the insights and hypothesis we obtained by engaging with the data through the whole data analysis process described above, we formulate an interpretive theory. Interpretive theory aims to offer accounts for what is happening, how it arises and explains why it happens (Charmaz, 2014, p. 230). In this work we approach interpretive theory and its construction from a pragmatist viewpoint. We recognize that our statements can only correlate our interpretation of the experience of individuals with our own experience, and the body of knowledge from the field that is available and known to us (Mead et al., 2015). Taking this viewpoint we emphasize practice and action rather than trying to explain the empirical phenomena described in the analyzed data by providing laws that are testable by empirical objective observation. Concretely, interpretive theory in this paper concerns what authors of posts assume about what they describe, how these assumptions or views might have been constructed, and how the authors seem to act on their views. By taking this approach of theory construction, we want to make phenomena and relationships between them visible in order to

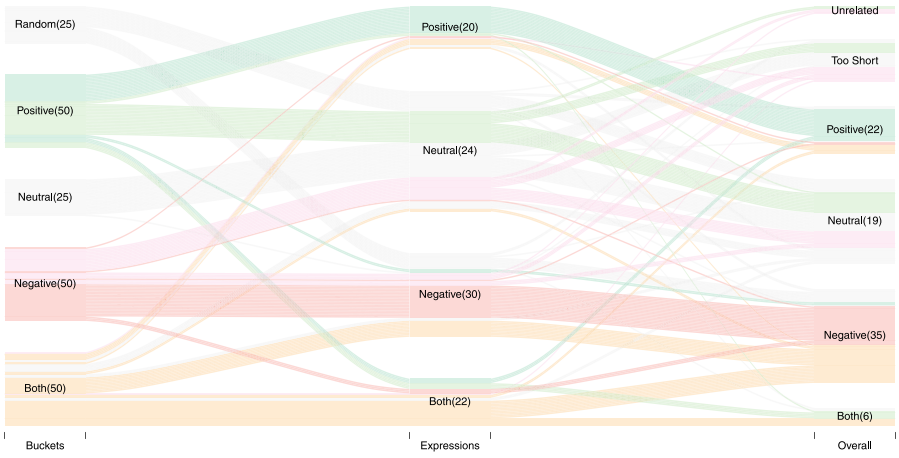


Fig. 3. Visualization of flows that display how our manual classification of posts (column two and three) matches with the automatic classification of our sentiment analysis pipeline (column one). The numbers in brackets in column two and three indicate the agreement of our manual classification with the automatic classification (first column). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

open up new vantage points for our own and the future work of others. We understand theorizing as an ongoing activity that can be continued through this future work (Charmaz, 2014).

4. Results

In this section we describe our findings and offer an interpretation of the data we analyze to answer the research questions. We first discuss the result of applying sentiment analysis tools to create a dataset that is rich in sentimental expression. We then present the results of our qualitative data analysis of this dataset to first show how software engineers express sentiment about testing and which underlying factors contribute to their sentiment. We then present a preliminary interpretive theory that synthesizes our findings. The data in which this preliminary theory is grounded, and all artifacts that are discussed in this section are contained in our replication package (Swillus and Zaidman, 2022, coded-dataset.qdpx).

We invite the reader to import the dataset contained in the replication package in the CAQDA-Software of their choice, and we also want to invite the reader to follow our analysis by using the online content on Stack Overflow. We enable this by providing a link to the original post on the Stack Overflow website that can be followed by clicking on the ID next to the quotation of a post. Example quotation: "This is all working as I would expect"(*3340677*) .

4.1. Sentiment analysis for qualitative research

Our sentiment analysis pipeline takes a Stack Overflow post as its input, classifies each paragraph of the post independently using two different sentiment analysis tools and takes the result of both tools into account to indicate if a post is likely to be positive, negative, neutral, or mixed in sentiment. Using this pipeline we created buckets of positive, negative and mixed sentiment posts, containing 50 documents each and added 25 neutral and 25 randomly selected posts to our analysis in order to validate our method. Our motivation to filter the dataset using sentiment analysis tools stems from research by Sengupta and Haythornthwaite (2020), which indicates that randomly selecting posts from the Stack Overflow dataset will only provide few sentimental posts, as the majority of posts is objective or focused on technical issues. Our approach relies on multiple sentiment analysis tools to address a problem that was identified by Lin et al. (2018) demonstrating that sentiment analysis can introduce a strong bias when relying on a single tool. In Fig. 3 we compare the classification of our sentiment analysis pipeline (left column) with the sentiment that we *actually* identified in posts during initial coding (center and right column). We differentiate between occurrences of sentimental expressions in documents (center column) and the overall sentiment of a document (right column). Using the metrics which are visualized in Fig. 3 we evaluate, how suitable our method is to create a dataset that can be used to find answers for our research questions and if it is applicable for other qualitative studies on Stack Overflow.

4.1.1. Occurrences of sentimental expressions in posts

Occurrences of sentimental *expressions* in posts were identified and annotated during the first coding cycle when posts were coded line by line. The line "I understand that using aunit can be a time-saver"(*3412892*) was classified as positive for example, but the same post also contains the expression "I looked at the aunit manual and I didn't find easy examples to start with" , which was classified as negative. Post *3412892*, which we took from the *positive* bucket, was therefore assigned the category of *both* sentiments at the level of expressions. The flow from the first to the second column in Fig. 3 shows this relation, presenting which posts from each of the sample buckets contained expressions of the respective sentiment. 20 posts from the bucket of positive posts for example indeed contained one or more positive sentimental expressions and no negative ones. In Fig. 3 this relation is represented by the flow from *positive* in column one to *positive* in column two, highlighted in green. However, 2 of the 50 posts from the same bucket did not contain a positive expression but at least one negative expression (flow from positive to negative), 7 posts contained at least one expression of each sentiment (flow from positive to both) and 21 posts from the positive bucket did not contain any sentimental expressions (flow from positive to neutral). Flows from the *negative* and *positive* buckets to the *neutral* category in column two indicate that a lot of posts identified as positive or negative by our pipeline in fact did not contain any sentimental expressions. Comparing this lack of accuracy with the results for documents that we obtained from the *random* bucket suggests however that our sentiment analysis pipeline indeed managed to select more sentimental posts than a random selection would have. Crucially, we did not find a single positive expression in the set of 25 randomly selected posts. Additionally, comparing the

remaining flows between column one and two in Fig. 3, we see that the majority of posts that turned out to contain sentimental expressions were indeed extracted from the respective bucket. The findings of this first analysis of the accuracy of the sentiment analysis pipeline therefore supports our hypothesis that a semi-automated approach proves beneficial when used to create and analyze a subset of Stack Overflow posts with both negative and positive sentiment.

4.1.2. Overall sentiment of posts

In Fig. 3 the last column shows the conformity or difference of the overall sentiment of posts determined manually by us in comparison with our tool pipeline. We determined the overall sentiment of a user towards software testing during the second, focused coding cycle and assigned a polarity of *neutral, positive, negative*, or *both* to each post. During this analysis, we realized that 39 posts were not usable for further inquiry. The majority of those posts were too short (34); one author simply asks "Which is the best framework for automatic testing in octave? Why?" (*20732 44*) . The other five of those unusable posts were identified as unrelated to our work, like a post in which a practitioner asks "How to use Jquery Ajax Cache"(*2398092*) , mentioning testing but referring to something that is unrelated to automated testing. The dark green and dark red flows in Fig. 3, from column one via column two to column three show that posts from the positive and negative buckets that contain expressions with that sentiment were mostly leaning into that direction overall as well. There are only a few outliers of posts that were for example classified as negative by our pipeline and indeed only contained negative expressions but were found to express an overall positive sentiment. One such post contains the *negative* expression that "[it] is copy-paste code, which I thought was generally not recommended"(*9271925*) , not mentioning anything positive or negative apart from that. However, the overall sentiment of the post was interpreted as positive as the author shows a constructive willingness to improve while being open and concious of their own mistakes. In total, there were only 12 such cases where the sentiment classification of the pipeline completely diverged from our classification. Documents from the *both* bucket of our dataset, even when they indeed contained expressions of both sentiments were in most cases negative overall. The analysis also shows that the *both* bucket contributed the most sentimental posts to our dataset. Our analysis of the overall sentiment of posts indicates that subtle remarks and the context of a sentimental expression makes the overall classification of posts difficult. Subtracting unrelated (5) posts, randomly selected posts (25) and those that were too short for analysis (34), we can report that the sentiment prediction was correct for 46% of all documents (65 of 141). Overall our approach yielded a dataset in which approximately half of all documents were sentimental (108 of 200). We provide an annotation file with our replication package that contains sentiment annotations for each post that we analyzed on both the level of expression and overall, including the source code to generate graphs and statistics from that annotation file (Swillus and Zaidman, 2022, data/annotations.json).

4.2. Sentiments that affect attitudes

Before describing and comparing occurrences of sentimental expressions which we identify in the dataset by presenting focused codes and analytical categories, we provide examples which demonstrate how we moved from the data, through codes, towards a more abstract interpretive theory. Document *878848* was first coded line by line and was assigned, among others, the initial code –Expecting a lot of Work From Mocking. The code with the prefix "–", which indicates that the expression reflects negative sentiment, was assigned to the following line: *"Use EasyMock, write looooong mocking sequence. VERY BAD solution: hard to add initial data, hard to change data, big test debugging promices"*.. During the second and third initial coding cycle the code was then changed to – Expecting Mocking to be Bad Solution. Other posts hold similar notions and were coded with the same code (e.g., "There is no point in mocking out a whole ngrx entity store, so I would just like the selector to return exactly that object and be done with it."(*58840818*)). During focused coding, the code changed once again and became more abstract and analytical: "Judging subjectively". The comparison of posts with similar codes revealed that expectations which are expressed sentimentally, like the examples above, are not based on objective observations but on subjective perceptions often connected to personal experience. The intention (or action) of the author here does not seem to be the objective revelation of their expectations, but the subjective judgement in order to position themselves. In one memo titled *Experienced ambiguity* this notion of subjective judgement and ambiguity was noted by one of the authors during focused coding.

Memo: *Experienced ambiguity*

The practitioner is struggling with adopting a new framework. Some things are easy and some are challenging. The practitioner is faced with a situation in which there is no easy or obvious way forward. They are stuck and forced to make an uncomfortable decision. However the willingness to resolve the ambiguity here still reflects a very positive attitude. The practitioner already has some clues and they are reasoning from experience. Looking at the comments, I realized that the post was closed quite quickly. It only took about 10 h and the issue was solved by a maintainer of the framework project which is mentioned in the post. The fact that the author of the post reacts very enthusiastically supports my hunch that their attitude was actually quite positive all along.

The memo was originally created when analyzing another post (*823276*), but was then connected to post *878848* as well. Later, during a diagramming session, the aforementioned memo, some related focused codes and both posts (*823276* and *878848*) were assigned to a collection labeled *Confidence* which generated new memos and more abstract perspectives. Both this collection and the memo mentioned above also contributed to the forming of the categories *Aspiration* and *Exploration*. Post *878848*, which ultimately ended up in the category *Aspiration* and was categorized to reflect both positive and negative sentiment, further revealed what might be the conditions for aspiration to arise in the context of software testing. We compared the post with others of the same category and identified that knowledge and experience seems to enable practitioners to stay positive despite being stuck in situations where there is no obvious way forward. Concretely, we hypothesize that the notion of explicitly comparing capabilities of approaches, not only in terms of features, but also in terms of maintainability, indicates confidence and experience of the author on Stack Overflow. Ultimately, memos written about those considerations and others enabled us to construct the preliminary interpretive theory which we present at the end of this section. Specifically, the aforementioned post *878848* supports the hypothesis that experience and knowledge can give practitioners an extra degree of trust and confidence, from which an aspirational attitude towards testing seems to emerge.

4.2.1. Focused codes

Using focused coding techniques as recommended by Charmaz (2014), we identified 22 codes that were assigned to a total of almost 700 different text sections of the 200 posts that we analyzed. Table 1 lists all codes, a description for each, and a diagram that indicates how many posts that contained the code were identified to be either *positive, negative, neutral* or of *both* sentiments. The full codebook that we provide as part of the replication package of this paper contains inclusion and exclusion criteria, and examples for each code (Swillus and Zaidman, 2022, codebook.ods).

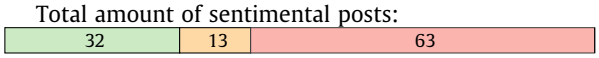
Comparing the codes and corresponding posts with each other reveals underlying sentiment of practitioners that relate to testing practice. The codes reveal patterns that affect attitude and testing practices of software engineers and allow us to propose answers to **RQ1**.

RQ1

How do software engineers express sentiment about testing on Stack Overflow?

In total, the dataset that we have analyzed contains 108 sentimental posts. In 32 posts, practitioners expressed positive sentiments, 63 posts were negative, and 13 contain both sentiments.

Total amount of sentimental posts:



To highlight some of the patterns which show how sentiment is expressed, we elaborate on the eight most occurring codes from Table 1 and explain with examples what was captured with those codes.

Judging Subjectively (F.7). About one third of sentimental posts (32 of 108) contained an explicit subjective statement about apparent characteristics or value. Subjective expressions like that of one practitioner who "FELL IN LOVE WITH THE CRISP SYNTAX [OF A FRAMEWORK] IMMEDIATELY"(*1072952*) underline the attitude of the author. Negative attitudes connected to judgement like one practitioner reflecting on a specific practice which "SEEMS LIKE A WASTE OF TIME"(*29894788*) were rarer in the dataset than positive attitudes. One practitioner for example reflects positively "THAT [RUNNING TESTS CONCURRENTLY] WILL FORCE [THEM] TO REFACTOR SOME CODE TO MAKE IT THREAD-SAFE, BUT [THEY] CONSIDER THAT TO BE A GOOD THING :-)"(*4970907*) . In total, more than one third of all positive posts (14 of 32) contained a subjective judgement compared to only every fifth negative post (12 of 63).

Lack of Knowledge (F.8), Facing Uncertainties (F.5) and Reassuring the Reader (F.2). Outlining the limits or lack of their own knowledge and abilities by stating for example that they are "A NEWBIE"(*29894788*) , or indirectly pointing out that they are "STUCK TRYING TO [...] TEST AN EXTREMELY SIMPLE PROJECT" (*62177256*) occurs both in positive and negative posts in around a quarter (27 of 108) of all sentimental posts. In addition to describing their own limits by admitting a lack of knowledge, we identified descriptions of ambivalence ("WHICH IS THE CORRECT WAY?"(*41262775*)), doubt ("HAS ANYONE DONE ANYTHING SIMILAR BEFORE OR IS THIS CRAZY?"(*7213917*)), or uncertainty ("IT SEEMS TO ME THAT, I MAYBE SHOULD BE CREATING A FAKE MATERIALREPOSITORY, RATHER THAN MOCKING IT?"(*23534123*)) expressing insecurity in around a third (35 of 108) of all sentimental posts. We also found statements indicating that the author is trying to maintain or restore their confidence by reassuring the reader in more than a third of sentimental posts (43 of 108). One author for example is stuck in a situation where they observe something unexpected and they "WANT TO UNDERSTAND WHY THAT IS LIKE THIS"(*39592949*) , wondering if "THERE IS A BETTER WAY" , even being afraid that their "CODE IS JUST BAD" but still holding on to their approach as they reassure the audience that "WHEN [THEY] CHANGE [SOMETHING,] EVERYTHING WORKS FINE" .

Pursuing Ambition (F.3) and Willing to Improve (F.4). *Uncertainties* and a *lack of knowledge* were found equally frequent in negative and positive posts, but descriptions of constructive attitudes to achieve a goal that goes beyond just getting the job done were mostly found in positive posts, or posts that contain both sentiments. We identified direct expressions of ambition by practitioners for example "TO CREATE A SUPPORT LIBRARY THAT COULD BE USED BY ALL TEST PROJECTS"(*18399610*) , or mentioning the context of a challenge that underlines its ambitious nature like "WRITING ACCEPTANCE TESTS FOR A SINGLE FEATURE OF A LARGE APP [, NEEDING] A LOT OF DATA FOR THIS AND [HAVING] A LOT OF SCENARIOS TO TEST"(*28129825*) . Those expressions were found in over a third of positive posts (13 of 32) but contrary only in around one fourth of negative posts (15 of 63). Related, and very similar to these expressions are verbalized intentions to improve, for example by wanting to "STRUCTURE [A] UNIT TEST IN A BETTER WAY"(*43275116*) or by asking for "THE BEST PRACTICE IN [A PARTICULAR] CASE"(*46177956*) . Just like mentions of ambitions, expressions of a willingness to improve occurred in more than one third of all positive posts (14 of 32), and contrarily only in less than a fourth of negative posts (11 of 63). Together, expressions of ambition and willingness to improve cover almost three quarters of all positive posts (23 of 32).

Expressing Desperation (F.6) and Unexpected Behavior (F.1). Contrarily to ambitions we also found expressions of despair by practitioners who are stuck saying that they for example "GOOGLED WIDE AND FAR, BUT DID NOT GET ANY ANSWER"(*58840818*) , or remain completely helpless, begging for support like one practitioners who asks: "CAN SOMEBODY PLEASE, PLEASE, PLEASE FOR PETE'S SAKE [...] FIX THIS BUG THAT THOUSANDS ARE HAVING?" (*44762082*) . We did not observe expressions of desperation in positive posts or posts with both sentiments, but we did find them in almost half (31 of 63) of negative posts. Additionally, we identify descriptions of unexpected behavior in more than half of negative posts (41 of 63). Covering a big fraction of the dataset, unexpected behavior is experienced by practitioners in many different contexts, referring to testing practices or the development environment ("WHEN I TEST IT IN BROWSER, EVERYTHING IS OK, BECAUSE APP\USER EXISTS, BUT WHEN I TEST MY PLUGIN, APP\USER DOESN'T EXISTS"(*52760148*)), or referring to something that is not directly related to testing but discovered through it like facing a floating point precision error for the first time, noticing that "WHEN I'M RUNNING THE TESTS IT'S BROKEN BECAUSE 0.1 IS NOT EQUAL TO 10%"(*63886733*) .

4.2.2. From codes to categories

We use codes to compare posts with each other in a structured way. Codes enable us to scrutinize the dataset from different perspectives. Co-occurrences of codes within posts for example reveal patterns in the data that can be indicators for categories. We identified four major factors that describe the non-technical, situational context of sentimental posts with which we can categorize the posts. In this section we present each category and their characteristics, highlighting key insights that emerged from the data during our analysis when categories were outlined. The categories reveal underlying currents that affect the testing practices of software engineers. Categories which highlight what influences their attitude and motivation are the basis of what we propose as answers to **RQ2**.

RQ2

Which factors affect sentiment of software engineers towards testing practices?

Table 1
Focused codes, their description and the co-occurrences with respective sentiments. The co-occurrence bar-chart indicates in how many documents of the overall sentiments positive, negative, neutral or both the respective code was identified. Codes are ordered by the amount of documents in which they were found.

Focused Code (F.X) Description	Sentiment occurrence Positive / Both / Negative / Neutral	(Σ)
Observing Unexpected Behavior (F.1) An expression that reveals that something does not work as the author expects. Like a dump of error logs that seem to be nonsensical to the author.	1 / 41 / 20	(62)
Reassuring the Reader (F.2) Making a statement to restore confidence. Like a claim that a manual has been read, or a tutorial has been followed.	13 / 4 / 26 / 18	(61)
Pursuing Ambition (F.3) Constructive attitude to achieve a goal. The implementation of something, extension of knowledge or something else that goes beyond just getting the job done.	13 / 4 / 15 / 15	(47)
Willing to Improve (F.4) Author indicates that they have an ambition to change and improve something.	14 / 4 / 11 / 13	(42)
Facing Uncertainties (F.5) Expression of insecurity through description of ambivalence or doubt.	10 / 9 / 16 / 8	(43)
Expressing Desperation (F.6) Author expresses their desperation directly, either by asking a question or by indicating that they are clueless.	31 / 7	(38)
Judging Subjectively (F.7) Explicit subjective valuation of the apparent characteristics, behavior or value of something.	14 / 6 / 12 / 2	(34)
Admitting Lack of Knowledge (F.8) Direct or indirect expression of a lack of knowledge.	5 / 6 / 16 / 9	(36)
Searching for a New Path (F.9) The goal or approach has been thought through but the author hunches that there is another, better way.	10 / 5 / 5 / 14	(34)
Contemplating Complexity (F.10) Author is describing something that has to do with the complexity of a setup or use-case. Complexity is either highlighted reflected implicitly.	7 / 5 / 10 / 9	(31)
Missing Capability (F.11) Description of issues, circumstances, hurdles or other discomforts that stop one from reaching a goal. Capabilities can be the capabilities of a software, its limitations, but also the own capabilities to solve an issue.	2 / 4 / 13 / 11	(30)
Referring to External Information (F.12) Reference is made to a resource that is accessible to the author. Documentation, blog posts, books etc.	8 / 3 / 10 / 8	(29)
Contemplating Failure / Difficulties (F.13) Author shares their opinion about what they find difficult or failure they are facing.	5 / 3 / 15 / 4	(27)
Looking for Starting Point (F.14) Request for a starting point to tackle something that is unknown or unclear.	7 / 1 / 11 / 8	(27)
Facing an Obstacle (F.15) An obstacle makes it impossible to continue with a task. The author is stuck because of the obstacle.	1 / 1 / 14 / 3	(19)
Reflecting Experience (F.16) Positive or negative reflection which is related to past experience.	6 / 4 / 4 / 1	(15)
Struggling to Understand (F.17) Author is struggling to grasp the meaning of a faced problem or a concept they want to learn. Like admitting that they are not able to comprehend something or that something is hindering them to learn something.	1 / 3 / 9 / 2	(15)
Seeing Own Mistakes (F.18) Realization of an error or a misconception. Revelation of having done something in the wrong way or in a way that can be improved.	4 / 1 / 3 / 7	(15)
Comparing Different Approaches (F.19) Description of multiple angles to solve an issue or a task.	6 / 2 / 2 / 5	(15)
Trial and Error (F.20) Describing different attempts to get to a solution which are all unsuccessful.	7 / 5	(12)
Aiming at a workaround (F.21) Practitioner identifies that a situation can be solved by using some workaround which is probably not the ideal solution.	6 / 5	(11)
Excluding Solution (F.22) There is a solution for a problem but the author does not want or cannot use it.	2 / 2	(4)

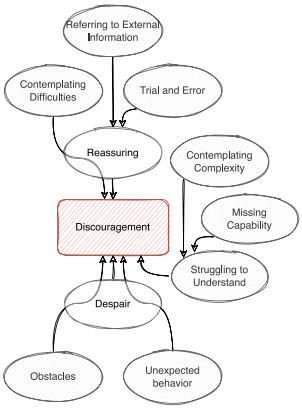


Fig. 4. Focused codes and how they are related to the analytical category *Discouragement*.

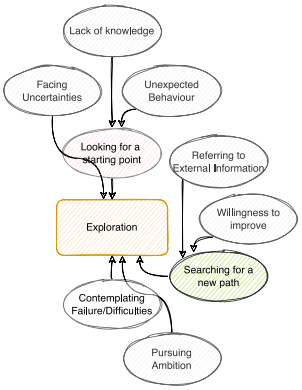


Fig. 5. Focused codes and how they are related to the analytical category *Exploration*.

Discouragement (C.1)

42 | 10 (52)

We found that attitude in negative sentimental posts is often (42 out of 63) expressing *discouragement (C.1)* from testing. *Unexpected behavior (F.1)* can bring efforts to a halt [6441026, 37439708, 53935108, 44095109, 31052776, 32408965, 55644155, 56577906], sometimes made explicit in posts by references to *an obstacle that is faced (F.15)* [8338348, 32622060, 20480791, 38932495, 6376925, 6376925]. *Expressions of despair (F.6)* underline the weight of these setbacks in those posts [3736614, 58840818, 67734277, 53935108, 17068154, 33607092, 44762082, 61782427]. When authors sentimentally express discouraging setbacks in their testing efforts by *contemplating difficulties or failure (F.13)* they are at the same time often *reassuring the reader (F.2)*, implying that the problem cannot be blamed on them [63795587, 14942409, 19490583, 18083834, 19799393, 25264248, 26370705]. Statements that a *tutorial or documentation (F.12)* was followed and thoroughly read, or reports of *elaborate debugging (F.20)* demonstrate the confidence of the author [13309278, 32009877, 34889215, 6579379, 14701609]. A *complex development environment (F.10)*, including company policies or unique infrastructure configuration is mentioned in the context of such cases [6475042, 42211311, 14554366, 18038203, 17068154, 43435227]. In complex situations, even a small step, like writing a unit test, can cause a lengthy and often fruitless pursuit [52760148, 37527179, 55357595, 36608077, 67734277, 14942409]. When tools, methods, and concepts are *not easily understandable (F.17)*, especially when *documentation is not extensive enough (F.11)*, practitioners are discouraged to hold on to their ambition [44010437, 63795587, 61769730, 19799393, 43435227, 7292700, 62177256] (see Fig. 4).

Recapitulation: Emergent Category *Discouragement*

Discouraging sentiment about testing is provoked in complex development environments. This includes company policies or unique infrastructure configuration. When such factors combine with technical issues, experienced by the practitioner as unexpected behavior, they create obstacles that discourage practitioners from testing. A complex environment makes the usage of a standard testing tool chain unexpectedly challenging, especially when practitioners lack experience in testing. Documentation or other external resources do not help in these cases and long fruitless pursuits of trial and error are reported (see Fig. 4).

Exploration (C.2)

13 | 5 | 10 | 16 (44)

Contrary to posts in which a discouraging sentiment is expressed, posts of practitioners who approach testing with an *exploratory (C.2)* sentiment, reflect both positive and negative attitudes. In the context of exploration, reaching out to the Stack Overflow community is motivated by an *ambition (F.3)* to overcome *difficulties or failure (F.13)*. [57299238, 29305776, 53376098, 3340677]. We observe that practitioners have a positive attitude when they indicate a *willingness to improve(F.4)*, [53657417, 41135403]especially if they are *searching for a new path (F.9)* [14602848, 946069, 6022092, 7213917]to solve a problem by asking for available best practices in a particular situation. *Resources (F.12)* like a blog post or documentation or other external factors seem to trigger positive ambitions of practitioners in those cases [67709670, 32046670, 4659714]. When practitioners are struggling with basic concepts however, for example by *looking for a starting point (F.14)*, they show a negative attitude [61342139, 55176792, 3412892, 57609818]. Exploration with this negative attitude is connected to acknowledgement of a *lack of knowledge (F.8)* [59729159, 46713912], or *uncertainties (F.5)* about practices. [12659810, 7960832, 4288448, 2894608, 823276]. Crucially, in cases where practitioners that explore testing report *unexpected behaviour (F.1)*, their attitude is exclusively negative [49480999, 37439708, 29305776](see Fig. 5).

Recapitulation: Emergent Category *Exploration*

Exploratory sentiment to discover and learn is expressed both positively and negatively by practitioners. Trust into method or technology based on experience or inspiring external impulses arouses positive attitudes. When exploration serves clarification in situations of uncertainty, it is the experience of unexpected behavior of technology that causes negativity especially when practitioners lack experience (see Fig. 5).

Reflection (C.3)

8 | 3 | 11 | 8 (30)

We identify negative and positive posts in which practitioners sentimentally and critically *reflect* on their testing practices or understanding. *Reflection of experiences (F.16)* and expressions of an ambition to *improve (F.4)* when they are *facing uncertainties (F.5)* form the baseline of this category [398004, 59781140, 49713083]. Similar to the posts we categorized as *exploration*, *uncertainties of practitioners (F.5)* are directly expressed or indicated through attempts to *reassure the reader (F.2)* [41262775, 58684292, 687748, 29894788]. In this category however, we observe that practitioners are more *aware of their mistakes (F.18)* or their *struggle to understand (F.17)* aspects of

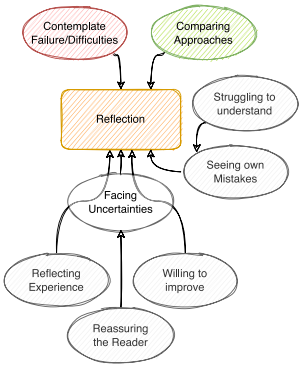


Fig. 6. Focused codes and how they are related to the analytical category *Reflection*.

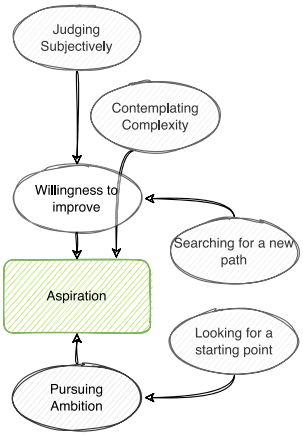
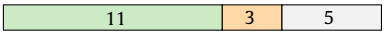


Fig. 7. Focused codes and how they are related to the analytical category *Aspiration*.

testing [42275344, 4991264, 4970907, 67746901, 39892949]. Posts that reflect a positive attitude contain analysis and *comparisons of approaches (F.19)* [46177956, 41816229]. In contrast, when practitioners *contemplate failure (F.13)* or *complexity (F.10)* their attitude is almost exclusively negative [64464005, 1064403, 18941509, 25325133, 42374231](see Fig. 6).

Recapitulation: Emergent Category *Reflection*

Application of testing practices can lead to ambiguity. Applying the right method in a particular situation for example can be challenging. Awareness of blind spots and knowledge of the great variety of tools and methods, is a factor that allows practitioners to keep a positive attitude. Variety and ambiguity can than even be appreciated. When failure or complications cause ambiguity however, sentimental reflection is negative (see Fig. 6).

Aspiration (C.4)  (19)

Opposite to posts from the category of *discouragement*, we identify *aspiration* in posts which express almost exclusively positive attitudes towards testing. Specifically, aspiration reflects a degree of freedom that allows exploration and discovery in a constructive way. In particular, the motivation is not to find a workaround or to overcome an obstacle, nor do authors elaborate on extensive debugging or trial and error. Instead, authors *pursue ambitions (F.3)* that go beyond a particular situation [34657563, 22246656]and express *intentions to improve (F.4)* their testing practices [16938742, 14961412, 48113464, 6684337, 878848]. Facing *complex situations (F.10)* is here not a cause for distress, but rather a motivation to improve testing practices [28129825, 1072952, 23062243]. Motivation is expressed by authors through explicit positive *judgments of value (F.7)* of testing [280645]. The post on Stack Overflow can in those cases be an attempt to *find a new way (F.9)* to tackle a problem [9271925]or to probe for a *starting point (F.14)* [1006189, 52539907](see Fig. 7).

Recapitulation: Emergent Category *Aspiration*

Understanding of long term goals and the value of testing arouses aspirational sentiment. Not being trapped in a problematic or complicated situation and not having to deal with an immediate obstacle creates space that is required for this aspirational attitude. It allows practitioners to build essential knowledge before their ignorance produces problems (see Fig. 7).

4.2.3. Factors that arouse sentiment

To answer **RQ2** (Which factors affect sentiment of software engineers towards testing practices?), we summarize key insights we gained by developing the above categories. We identify that practitioners on Stack Overflow express sentiments when they are either discouraged (C.1) from pursuing their goal, aspiring (C.4) towards something that goes beyond their usual practice, reflect (C.3) on their testing experience and knowledge, or when they are exploring (C.2) what is still unfamiliar to them. Posts which indicate aspiration (C.4) are positive in sentiment, and posts that describe notions of discouragement (C.1) from testing mostly reflect negative sentiment. Common factors can be identified even among those two almost inverse categories. Concretely, we identify that the experience of unexpected behavior is an important factor that leads to negative sentiment expressed through discouragement. Even when exploring (C.2) or reflecting on (C.3) testing practices to learn and gain knowledge practitioners express negative sentiments when they face unexpected behavior that causes ambiguity. Additionally, data suggests that an absence of those unexpected setbacks enables conditions for practitioners to aspire. Through reflection and exploration, these conditions allow them to build knowledge and experience. Experience, which is likely to prevent those unexpected setbacks in the future. Trust in testing practices that is established through these experiences contributes to positive sentiments when new practices are explored. We find the same to be the case for an awareness of blind spots. Reflection (C.3) on their testing practices that express an awareness of blind spots reflects positive sentiment and attitude. Uncertainty in those cases inspire practitioners instead of discouraging them.

4.3. Trust, complexity and testing – preliminary theory

We set out to discover what makes practitioners sentimental about testing by looking at how they express sentiment on Stack Overflow. We want to know which factors and situations contribute to sentiment. By analyzing, categorizing, and comparing the dataset, we got a glimpse of what the experience of practitioners, who ask questions on Stack Overflow must be like. Codes and categories described in the previous sections enabled us to analyze the dataset systematically using techniques like clustering and diagramming. In this section we present a preliminary

interpretive theory that describes what became visible from our perspective, which is grounded in the analyzed dataset. To let the data speak for itself, we provide references to the original posts on Stack Overflow immediately in the text. With each quotation from posts, we also provide a reference to the code that was assigned to the respective text section where applicable. Fig. 8 illustrates our preliminary theory as an interplay of the most crucial factors which we identified to have an influence on sentiment towards testing on Stack Overflow. We first elaborate on the right side of the figure, which shows discouragement (C.1) in the context of software testing, and how the negative sentiment around it is aroused in situations where complexity plays a central role. We then turn to the left side of the figure, elaborating which role exploration (C.2), reflection (C.3) and aspiration (C.4) play in the context of testing.

"I was starting to break as much as I was fixing. So I decided I'll start from scratch, with TDD this time"(*29894788*) (F.3). Testing practices and approaches are multi faceted. Even in cases where practitioners are just "having a play with testing" (*28129825*) (F.4) to improve their code base, or just to "understand the essence of it"(*44202672*) (F.3), they are quickly faced with multiple tools and have to make difficult choices regarding the technique or tools to adopt for a use-case. The dataset that we analyzed demonstrates that testing software is not a single tool or single method practice. We observe that the big landscape of software testing tools and the resulting diversity of possibilities to practice testing amplifies ambivalence when practitioners lack experience and knowledge [*878848*, *1006189*, *12950163*, *601973*, *17320143*]. The question whether or not "I [am] missing something in my pursuit of cool and trendy stuff [...] ditching the old proven [ways]"(*2894608*) (F.5) expresses the lingering insecurities of practitioners who are plunging into a world where many and often unexpected aspects of software engineering suddenly come together [*823276*, *43435227*, *1454949*]. As software projects get more complex, the ambition "to fully automate testing [...] in the most simple way possible"(*16938742*) (F.3) using advanced practices that are able to tackle this increased complexity grows as well. Our investigation indicates that this clash of lack of experience in testing on the one hand, and complicated challenges on the other hand drives attitudes around software testing [*4991264*, *43435227*]. As shown in Fig. 8, as a circular pattern, we identify that a growth in complexity of either the development environment or the software project itself makes practitioners ambitious to learn (more) about software testing [*1072952*, *16938742*, *1006189*]. But a high level of complexity of production code (top of Fig. 8) also requires complex testing code which in turn requires more than basic knowledge of testing (bottom of Fig. 8). The interplay of growing ambition, a complex environment, and a lack of knowledge is reflected in a question about an *easy* way to write a unit test. The practitioner asks: "I'm refactoring one big complicated piece of code [...]. So, I need to write a unit test" (F.3) "[...]. After googling I came up with 2 ideas" (F.8) "[...]. Am I missing some silver bullet? Possibly, DBUnit is the tool for this?"(*878848*) (F.9). Unfortunately, practitioners only start to face their ambiguities and insecurities around testing when they are "starting a new project, that promises to be much bigger and more involved than anything [they] have done in the past"(*6684337*) (F.4). In other words: instead of learning testing practices, starting with simple comprehensible setups and then iteratively building knowledge as the complexity of test suites and source code under test grow simultaneously, practitioners throw themselves into cold water when it is too late for simple, approachable solutions [*19490583*, *6475042*, *53657417*, *4659714*]. When the *silver bullet* is not found, they get discouraged to continue with their ambition [*878848*, *63795587*, *14942409*, *7960832*]. Our data analysis suggests that discouragement (C.1) is often connected to this phenomenon as expressions of desperation (F.6) indicate strong negative sentiment when practitioners are stuck (F.15), sometimes after they already "googled wide and far"(*58840818*) (F.6), "searching for days to find an answer"(*43435227*) (F.6). Unhelpful gathered information (F.12) which is often referenced in Stack Overflow posts only increase negative sentiment, and sometimes leads practitioners to identify unexpected behavior (F.1) of testing tools and libraries as *weird* or "strange behavior, because documentation says [that something should work. But:] Well, this is not happening."(*63795587*) (F.7) [*19490583*, *26370705*]. An explanation for this could be that documentation of testing tools and tutorials for beginners are more likely to focus on simple and standard use-cases [*57609818*, *6475042*, *13309278*, *37527179*, *34889215*, *14701609*]. Based on our anecdotal experience as software engineers using testing practices, we hypothesize that a divergence from best-practices in both software design and development environment, requires practitioners to rely on testing experience. In the context of highly inventive or original approaches, simple tutorials for testing are not applicable. It is very likely that more than one testing library is required in those complex non-standard software environments.

Complexity in Testing Practice

Before we set out to investigate what lies behind sentiment around software testing on Stack Overflow, we assumed that it will mostly be connected to tool failure or bugs. We expected to find sentimental complaints about specific (missing) features in a specific version of libraries for example. Our analysis shows however that it is more likely to be a struggle in overcoming overwhelming complexity with methods or combinations of tools that practitioners are not experienced enough with which causes negative sentiment.

Testing software can confront practitioners with misconceptions or flaws of their software projects. One practitioners asks: "Is this a valid unit test? If not, is it because I have bad design [...]? Because currently, I see absolutely no benefit in writing this test"(*44202672*) (F.5). Even as the majority of sentimental post that we analyzed reveal discouragement and negativity as described in the preceding paragraphs, some authors maintain a constructive and even aspirational attitude (C.4), even when they are facing difficulties (F.15). We observe that positive posts rarely contain descriptions of unexpected behavior or expressions of desperation. In contrast, even in difficult situations, practitioners even express hope [*59729159*, *1072952*, *34657563*, *53376098*, *41135403*]. In a post of a practitioner looking for a way to test a WebAPI, they contemplate that"Back when WCF was the coolest thing, I did tests like this [...]. All programatically. It worked like a charm"(*25325133*) (F.16). Even though they experience difficulties (F.13), explaining that "for some reason [it] is REALLY hard to get to work (as in, I haven't succeeded yet)" (F.13), they do not seem to be discouraged and eventually find a solution that works for them. Another practitioner mentions that"in Katalon [there] is a very nice way to parameterize the selectors for GUI elements"(*52539907*) (F.16), searching for a way (F.9) to make their testing code cleaner. Yet another practitioner judges enthusiastically (F.7) that"[validating the correctness of every component in their system is] obviously going to be quite a lot of work! It could take years, but for this kind of project it's worth it"(*1006189*) (F.7), also emphasizing that they already"have a very comprehensive unit-test suite" (F.7) and going so far as defining what they believe to be meaningful tests (F.10). We find that a commonality of positive posts is a sign of confidence of practitioners, or a trust in tools or methods that

is grounded in positive experience (F.16) [67709670, 46177956, 14961412, 1072952]. We also identify that ambition (F.3) and aspiration (C.4) in positive posts is connected by practitioners to their long term goals. One practitioner contemplates that "THE CODE WORKS 'PROPERLY' [...] BUT [THEY] THINK AUTOMATED TESTS WOULD BE GOOD FOR THE LONGEVITY OF THE PROGRAM"(48113464) (F.7), and another reports that they are "STARTING A NEW PROJECT, THAT PROMISES TO BE MUCH BIGGER AND MORE INVOLVED THAN ANYTHING [THEY] HAVE DONE IN THE PAST."(6684337) (F.4), which motivates them to "KEEP A GOOD WORKFLOW WITH [THEIR] TEST AND MAKE SURE [THEY ARE] NOT CREATING GAPS IN [THEIR] TESTING AS [THEY] GO" (F.9). As indicated in Fig. 8, it is experience and knowledge that gives those practitioners an extra degree of trust and confidence, from which an aspirational attitude (C.4) towards testing seems to emerge. Their attitude enables them to reflect (C.3) on and explore (C.4) solutions for long term goals [4659714]. They build knowledge proactively without experiencing setbacks that discouraged (C.1) practitioners report [57609818]. On the left side in Fig. 8 we visualize that exploration (C.2) and reflection (C.3) contribute to building knowledge that will eventually allow them to build trust and confidence. But, more crucially, seen at the top of the figure, we indicate that it is the context in which the ambition to test arises, that determines the sentiment towards testing when they engage in this process of building up knowledge. More concretely, when their environment and experience gives them confidence and if their ambition is grounded in an aspirational attitude, they remain positive [1006189, 3340677, 23062243, 16938742, 53657417, 1072952, 4659714]. But when their ambition to test emerges in situations when the complexity of their software projects begins to overwhelm them, the process of reflection (C.3) and exploration (C.2) is negative [37527179, 67746901, 58840818, 4991264, 7960832, 25325133, 6475042, 18941509]. Testing is then perceived as an obstacle that might even push complexity further and not as something that is good for the future of a project.

Trust and Confidence – Degrees for Aspiration

Knowledge and experience in testing practices allow practitioners to aspire and enables them to consider and realize long term goals. It also enables them to reflect on their practice and explore new possibilities in a positive light. When exploration and reflection of testing practices are however motivated by pressure, for example an increase in complexity of a project, which rendered manual testing impossible, their ambition might be abandoned. Testing then turns into yet another obstacle.

5. Discussion

The qualitative analysis of 200 Stack Overflow posts revealed many different facets of software testing to us. In this section, we revisit our research questions in the light of these observations, their implications, and the recommendations we draw from them. We then present threats to the validity of these findings and close the chapter elaborating future work, that will open the next stage of our grounded theory research. Before revisiting our research questions and elaborating future work, we want to turn the focus once more on the filtering process that yielded the dataset that was analyzed in this paper.

5.1. Semi-automated filtering of datasets for qualitative and quantitative research

To narrow down our qualitative analysis of the Stack Overflow dataset we have used a semi-automated two-step process. We first filtered the dataset using tags and then employed sentiment analysis tools to extract posts which contain sentimental expressions. We therefore consider the first, tag based filtering approach that is inspired by Yang et al. (2016) suitable for qualitative studies like ours. The low failure rate of the method in our case suggests that the approach is also suitable for quantitative studies of testing posts on Stack Overflow.

Regarding the second step, for which sentiment analysis tools were used, our evaluation is more differentiated. Our analysis supports previous observations by Lin et al. (2018) and Sengupta and Haythornthwaite (2020): authors on Stack Overflow indeed tend to discuss technology in an objective, non-sentimental way. Our analysis of 25 randomly selected (only tag-filtered) posts indicates that authors who express sentiment when asking questions about testing topics on Stack Overflow are more often expressing negative sentiment than positive. Out of those 25 posts, not a single one contained positive sentiment. In the light of those observations we argue that sentiment analysis indeed supported the goal to extract a subset of posts that contains both positive and negative sentiment. Deliberately extracting positive and negative sentimental posts provided an improvement in terms of balance in sentiment. In other words: a random selection would have only provided very few positive posts. However, we do not consider our approach applicable for quantitative studies where results and implications are directly discerned from the output of sentiment analysis tools. The accuracy of predictions for sentiment was simply not accurate enough to provide meaningful insights when only evaluating numbers. Posts predicted as positive and negative only turned out to be correct in 50% of all cases (50 out of 100). In 5 cases the sentiment was even the opposite of what was predicted. We also learned that the sentiment analysis pipeline is most accurate in identifying neutral posts. Out of 25 samples that were predicted to be neutral only 2 contained sentiment. Depending on the research question, an approach to identify content with neutral sentiment could therefore yield good results. We identified that 28 posts of our dataset were too short for meaningful analysis. For studies similar to ours we recommend to exclude short posts. Posts are more likely to contain subjective opinions and valuable content, when they contain more than 2 paragraphs of text. Our experience with analyzing the dataset by focusing on sentiment taught us that finding the right approach and selecting the right tools is challenging. We acknowledge that low accuracy of the tools we used is also due to the choices we made. For example, instead of using a training dataset containing sentences, we could have used a dataset with paragraphs (Wang et al., 2019), and instead of focusing on sentiment we could have focused on emotion detection (Novielli et al., 2018). The choices we made were founded on the literature that was known to us at the time. In the meantime however, Lin et al. (2022) published a literature review that contains a guideline for the appropriate usage of tools and approaches for opinion mining in software engineering. We can only encourage using their recommendations to navigate the field and to gain confidence in making the right choices.

5.2. How and why is sentiment expressed

We set out with our analysis of Stack Overflow posts to investigate how practitioners express sentiment in the context of software testing and which factors play a role when sentiment is expressed. We identified 22 codes which describe different expressions that are used by practitioners on Stack Overflow.

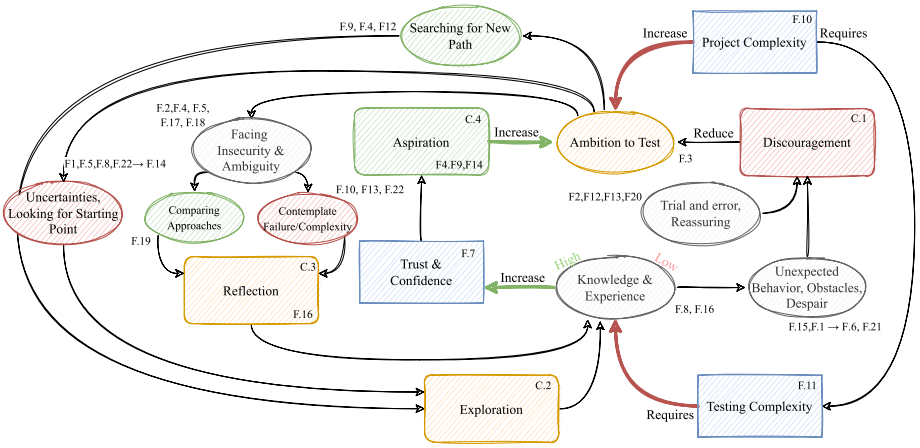


Fig. 8. Interdependence of factors which lead to sentiments around testing and how they are aroused and amplified in the context of complexity, trust and confidence through (the lack of) knowledge and experience.

RQ1

How do software engineers express sentiment about testing on Stack Overflow?

In sentimental posts on Stack Overflow practitioners are referring to external information like blogs or documentation, they reassure readers, share their ambition and subjective judgement of the value of testing practices and tools, compare different approaches, inquire for workarounds or new ways to solve a problem, admit their own lack of knowledge and their mistakes, reflect experiences, contemplate failure and sometimes exclude solutions that could solve their issues. Sentiment is expressed when desperation, unexpected behavior, uncertainties, complex issues, missing capabilities, or a willingness to improve is described.

The categorization of posts has allowed us to take our analysis beyond the level of expressions. We developed the four mayor categories discouragement, exploration, reflection, and aspiration, which illuminate factors that can lead to sentimentality.

RQ2

Which factors affect sentiment of software engineers towards testing practices?

Lack of experience and knowledge, especially in complex environments is often indicated in posts with negative sentiment on Stack Overflow, when practitioners describe discouraging experiences. Trust and confidence into practice and understanding of long term goals on the other hand gives practitioners space for aspiration, expressed with positive sentiment. Practitioners who explore testing express negative sentiment when they experience unexpected behavior and positive sentiment when they are inspired by resources like books and blog entries. When reflecting on their practice, an awareness of their own blind-spots allows practitioners to be positive, while ambiguity, when practitioners are completely in the dark, is reflected negatively.

Going beyond this analysis which highlights factors that lead to sentiment, we presented a preliminary theory that suggests how those factors go hand in hand in manifesting sentiment around testing. The preliminary theory also describes situational elements that seem to lead to sentiment.

Preliminary Interpretive Theory

On Stack Overflow we see complexity and aspiration as important factors that make people ambitious about testing. Complexity of projects can make manual testing impossible and motivates (or forces) practitioners to use testing. Trust and confidence in testing practices on the other side makes people aspire to pursue long term goals using testing practices. In both cases experience and knowledge influences whether this ambition leads to a positive or negative experience.

5.3. Implications

The results of our analysis of Stack Overflow posts about software testing carries implications for education of software developers, and management of software development teams. Based on the data we have seen, we hypothesize that the implementation of automated testing practices in simple projects, when manual testing is still possible, could allow an iterative development of testing skills while reducing the likelihood of discouraging experiences. Having obtained these skills, we argue, would then also influence the experience of testing complex systems in a positive way. Rejecting or approving this hypothesis could help to clarify the role that teaching of software testing can have in the early stages of software engineering careers (e.g., in undergraduate courses of universities). Connected to this hypothesis, our preliminary theory suggests that on Stack Overflow, testing practices are perceived as especially valuable when the complexity of a software project grows. Refining and testing this theory in other contexts could generate new insights into how practitioners and students of software engineering can be motivated to learn software testing. Pham et al. (2014) for example identify the same issue in a study with bachelor students. Their study confirms that the perception of the complexity of code affects students' motivation to practice testing. They also report that students see the cost of testing but fail to understand its benefit as projects are often not critical or complex enough. (Re-)introduction of testing practices, when complex software development methods are taught, so we hypothesize, could teach students the value of software testing. Introducing testing practices like mocking in the context of distributed systems and socket programming is one example. Regarding managers of software engineering teams, our preliminary theory implies that giving employees time and space to develop simple test cases for

simple projects is beneficial. Being comfortable with simple test practices, practitioners seem to gain confidence and trust. As a recommendation that should be tested in future work, we suggest that the development process should allow a steady increase of complexity instead of tackling huge challenges directly. The words of one author reflecting his work in a project where they introduced testing echoes this last implication of our interpretation: "While I no longer work on this project [...], I think it gave me some enormous insight into how bad some projects can be written, and steps one developer can take to make things a lot cleaner, readable and just flat out better with small, incremental steps over time."(*1064403*)

5.4. Threats to validity

Our systematic analysis of 200 Stack Overflow posts has led to insights that have enabled us to formulate preliminary hypotheses to answer our research questions and an interpretive theory. In this section we present the threats to the validity of our findings.

5.4.1. Internal validity

To select samples from the Stack Overflow dataset we filtered using user-assigned tags and the sentiment analysis tools SentiCR and RoBERTa. The dataset from Lin et al. (2018), which we used to train the tools, was evaluated by Zhang et al. (2020), who report macro- and micro-averaged F1-scores of 0.59 and 0.82 for SentiCR and 0.80 and 0.90 for RoBERTa respectively. However, their evaluation was done with a dataset of sentences and not at the level of paragraphs. We do not know if applying the tools on paragraphs, like we did, leads to poorer performance. We combined both tools to reduce inaccuracy as suggested by Zhang et al. (2020). We only selected posts that were classified with the same sentiment polarity by both tools. We checked the accuracy of the filtering approach by including and evaluating two groups of test samples in our analysis (25 random and 25 neutral posts) and classifying the sentiment of each post. Even though the precision of the tools combined provided only a 50% accuracy for positive posts, we argue that the inaccuracy does not pose a threat to our results. The results presented in this paper were produced by deep and thorough qualitative analysis for which the sentiment analysis was only a tool to narrow down the focus. The accuracy has no direct influence on the results of our analysis. To avoid mistakes in the implementation of the sentiment analysis tools, we used the open-source implementation of SentiCR from the replication package of Zhang et al. (2020),[8] and the open-source library PyTorch[9] which provides an implementation of roBERTa.

To extract posts from the dataset that are relevant to software testing we extended an existing open-source tool.[10] With our extension of the tool we first filtered for all post with a tag that includes the word *testing*. We then generated an include list of tags by manually removing all irrelevant tags that occurred in this subset of posts. Starting with a generic wild-card and then snowballing to generate a more accurate list of tags was found to be a valid method by Yang et al. (2016). Errors in the implementation of the filtering tool and mistakes during the manual selection of tags pose a possible threat to the validity of our results. To reduce the chance of implementation errors we only made minimal changes to the open-source software that was used for filtering. To minimize errors in the manual tag selection process, the final list was reviewed by two software engineering researchers who were otherwise not involved in this study.

5.4.2. Experimenter bias

We took measures to ensure that the influence of the authors' subjectiveness on the results of this paper stays within the boundaries of what is reasonable and expected in the context of a constructivist GT study. It is possible that the authors made mistakes in the interpretation of the dataset. To reduce the likelihood of a misinterpretation that would pose a threat to the validity of our results, the interpretation of the data recorded in memos and developed into sentiment classification, codes, categories and theory was discussed between the first and second author. Disagreements were resolved in a cooperative manner. We do 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 (McDonald et al., 2019). Aligned with our epistemological stance and the interpretive nature of constructivist GT, we instead acknowledge our biased perspective. Instead of claiming a high level of absolute objectivity, we argue that taking the view from nowhere, would not be appropriate to answer the research question that we propose. Instead, we present a transparent account of the grounds on which our interpretation rests. We use pertinent quotes and provide references to original documents whenever we explain our interpretations. The reader is invited to go through all the references in the text and the analyzed material that we provide with our replication package. Inspection of the material should reveal to the reader that we only make the material to speak for itself (Swillus and Zaidman, 2022, coded-dataset.qdpx). High involvement with the data, enabled by following the systematic strategies of constructivist GT, and not our preconceptions led to what we present in this paper.

We use sentiment analysis tools to filter the Stack Overflow dataset. It allowed us to narrow down the dataset to what is relevant for our study. To ensure that our own, manual evaluation of sentiments of posts and expressions is not biased by the outcome of this tool-based classification, documents were analyzed in random order and the results of the tool's classification were hidden during analysis.

5.4.3. External validity

Qualitative research searches for a deep understanding of the particular. Knowledge generated from such research is context dependent. We therefore cannot claim that the preliminary result that our analysis produces has a high external validity that goes beyond the scope of the Stack Overflow community. Stack Overflow posts, which are non-interactive documents, cannot provide a full or *thick* description of sociological circumstances (Geertz and Darnton, 2017; Hine, 2008). In other words: Stack Overflow posts only provided us a shallow view of the circumstances that practitioners experience; there are many things we are unable to see through an analysis of Stack Overflow posts. By sharing our preliminary interpretive theory we motivate inquiries that add more depth. More in-depth inquiries that either challenge the generalizability of what we have learned on Stack Overflow, or extend on it to fit a broader context than the one we investigated. To broaden the context of the posts, we considered comments, edits, and links that are referred to in posts and evaluated post's edit-history and the profiles of users that posted content. Further, the conclusions that allowed us to construct the results of this paper are based on the qualitative analysis of a small part of the full Stack Overflow dataset. As analyzing the full dataset is not feasible, we choose to focus our analysis on a fraction of sentimental posts. By not analyzing the whole dataset we risk missing details that could lead to different interpretations and hence different theories. We reduced this risk by consecutively adding posts to our analysis until we reach a point, when the analysis of further posts does not reveal any new answers to

[8] GitHub sorasmu/SA4SE
[9] PyTorch fairseq/roberta
[10] GitHub SkobelvIgor/stackexchange-xml-converter

the research questions we pose. We are aware that reaching such a point does not rule out the possibility that adding more posts can reveal new insights. It only signals that the effort required to obtain these insights gets disproportionate. Instead, concluding at this point allows moving forward to obtain insights from richer sources of data. Our analysis concluded in this way after reviewing 200 posts.

5.4.4. Construct validity

We investigate the role of sentiment in software testing posts to learn about the experience of software developers when they practice software testing. We use sentimentality as a construct and proxy to analyze content that goes beyond technical discussions and touches on this experience. By analyzing sentimental Stack Overflow posts we infer interpretations about how sentiments come about and how they affect testing practices. The root causes for sentiment of practitioners are manifold and might be due to variables which we were not able to consider in our investigation. This poses a threat to the validity of our results. We reduced this threat by analyzing the data qualitatively, taking contextual information of posts like comments, edit history and the time it took for the question to be answered into account. We are therefore not only relying on sentimentality as a variable to understand what affects practitioners.

5.5. Future work

The analysis described in this paper brought us closer to understanding what arouses sentiment in practitioners in the context of testing. However, as mentioned in the threats to external validity, the implications we present need to be taken with a grain of salt. Before suggesting which steps can be taken to raise our work to a higher level of maturity, we reflect on the limitations of the analysis presented in this paper.

5.5.1. Limitations

Stack Exchange, the parent website of Stack Overflow, provides insights about Stack Overflow by conducting an annual user survey. Their surveys' results and independent research about diversity on the platform reveals that the user base lacks diversity when it comes to ethnicity and gender (Ford et al., 2016). In their own report it is stated that people of color are underrepresented among professional developers on Stack Overflow and that the company has considerable work to do, to ensure the platform is inclusive.[11] According to Vadlamani and Baysal (2020), and Zagalsky et al. (2016) it is not only ethnicity and gender, but also professional factors that are strong reasons for (a lack of) engagement in the community. They lead to an *expert-bias* as novice contributors may even be confronted with subtle or overt bullying on Stack Overflow. Another bias is introduced through strict community guidelines.[12] During our investigation we were directly confronted with this limitation. Two posts that were rich in sentiment were closed because they violate the community guidelines. In one of those post, the message posted by a moderator reads: "AS IT CURRENTLY STANDS, THIS QUESTION IS NOT A GOOD FIT FOR OUR Q&A FORMAT. WE EXPECT ANSWERS TO BE SUPPORTED BY FACTS, REFERENCES, OR EXPERTISE, BUT THIS QUESTION WILL LIKELY SOLICIT DEBATE, ARGUMENTS, POLLING, OR EXTENDED DISCUSSION"(*16938742*) . In the other post, an author who has "BEEN BANGING [THEIR] HEAD AGAINST THE WALL TRYING TO UNDERSTAND [...] CONCEPTS FOR A WEEK"(*2978843*) simply suggested a "VERY UNDERSTANDABLE AND SIMPLE" explanation so that others can also enjoy an 'aha' moment. Examples like this make it evident that practitioners cannot express themselves freely on Stack Overflow. When they post exclusively sentimental content or ask questions that provoke discussion, they are sanctioned. The aforementioned post also suggests another limitation: practitioners posting on Stack Overflow are biased towards negativity. What is discussed on Stack Overflow are problems. If there is no problem to solve, the post is closed. Success stories or exclusively positive accounts of practitioners on Stack Overflow are therefore rare.

5.5.2. Theoretic sampling

Early stages in grounded theory are supposed to open up discussion and motivate for focused inquiries to follow. Theories mature as they are refined and backed by collection and analysis of more data. In grounded theory, this crucial process is called theoretic sampling (Charmaz, 2014). Apart from refining, verifying or rejecting our theory, such a focused collection of samples can answer questions that we derive directly from our analysis.

1. If ambition to test arises when practitioners are suddenly confronted with overwhelming project complexity, how do project management frameworks like Agile affect adoption of testing methods compared to projects that use long term fixed planning?
2. How are practitioners first confronted with testing practices? How does this first encounter with testing in a professional setting influence their ambitions to adopt testing in other contexts?
3. If the complexity of projects under test and the required complexity of techniques to test them grows proportionally like our preliminary theory suggests, how do developers of testing tools relate to this connection in terms of provided documentation and design of tools?
4. The analysis showed that sentiment around testing highly depends on context. In this study we looked at expressions of practitioners. How do researchers and educators in software engineering relate to testing in comparison to what we observed in our study? How does ambition differ, especially in cases where they have not been confronted with the factors that cause discouragement which we described in this paper?

As we highlight in Section 5.5.1, the dataset which was analyzed in this paper only provides a narrow perspective on the lived experience of practitioners. While Stack Overflow provides insights into what testers do outside their Integrated Development Environment (IDE), it only rarely provides insights into what testers do when they are not working on their computer. Posts rarely describe the social world in which testing is practiced. Derived from the things we did not see in the dataset, we propose the following questions for future inquiries:

1. How does the social context of individuals affect sentiment of testers when they are exploring or reflecting experiences?
2. Which role does the experience of peers play in shaping the testing experience of individual practitioners?
3. How do practitioners express sentiment about testing in informal settings?
4. How do practitioners express sentiment online, when ambiguous and sentimental content which provokes discussion is not sanctioned but encouraged?

In order to investigate the above questions, we propose different approaches. Through a *quantitative analysis* of Stack Overflow, Alshangiti et al. (2019) revealed that different challenges in the field of machine learning are present because implementation of application requires a wide set of skills. More concretely, they suggest that data preprocessing is especially challenging as it

11 insights.stackoverflow.com/survey/2021#section-demographics-gender
12 stackoverflow.com/help/how-to-ask

is often overlooked in education of practitioners. A quantitative content analysis like the one of Alshangiti et al. (2019) about testing posts on Stack Overflow could identify aspects of testing that are difficult to handle for practitioners on a more technical level. Further *qualitative studies* of non-interactive documents from platforms like Reddit[13] or Twitter,[14] which encourage sentimental and ambiguous content, can complement our analysis on a non-technical level. Conducting a *meta analysis* of publications on socio-technical aspect of software testing is another way of grounding our work in more theoretical and empirical data that others investigated in the past. But most crucially, we want to meet practitioners where they are confronted with testing practices. *Field studies* in which individuals or groups of practitioners are observed and interviewed during practice can provide insights that go beyond what non-interactive documents can reveal. Direct observations of practitioners will provide crucial insights into lived experience that allow the formulation of a mature theory.

6. Related work

With our investigation of sentimental posts on Stack Overflow, the categorization of posts and the development of a preliminary theory we highlighted different aspects that influence motivation of practitioners, the effect of emotions on practice, and the role of software testing as a part of software development. In this section we relate our findings to what others have uncovered in relation to those topics.

A study by Graziotin et al. (2018) emphasizes the detrimental effects that unhappiness can have on software engineering practitioners. Some of what they describe what happens when developers are (un)happy is relevant to our paper. According to their report, developers distance themselves from tasks to which their unhappiness relates. Our analysis reveals that confrontation with testing can under some circumstances cause negative feelings of discouragement. Discouragement can thus lead to withdrawal from testing resulting in process deviation and reduced code quality. On the positive side, findings of Graziotin et al. (2018) show that emotions related to happiness like aspiration increase process adherence and stimulate creativity, leading to a stronger commitment to writing tests.

A literature review by Beecham et al. (2008) compares the findings of 92 papers about the topic of motivation of software engineers from the 1980s to 2006. The review highlights that software engineers display a very high need for growth and that they are concerned about learning new technology. Software engineers are motivated by the exploration of new techniques and want to work on identifiable pieces of quality work. According to the review, problem-solving and the confrontation with challenges can be an enhancing factor for motivation. While those factors are present in many studies, the literature review concludes that the needs of software engineers are highly dependent on the context of individuals. Our study confirms this conclusion. Exploration can increase motivation or ambition in the case of software testing, but we indeed see that whether challenges or exploration lead to increased motivation highly depends on context. Contrary to the studies included in the review, we see that a confrontation with challenges can also lead to discouragement. Our results on this aspect are more aligned with the results of a qualitative study by Sharp et al. (2009), that suggests that challenges, even when mentioned as a reason to stay in the job, are not so much a factor that gives practitioners satisfaction. Not challenges, but creativity and being able to make a difference is what makes software engineering worthwhile (Sharp et al., 2009). Similarly, Meyer et al. (2021) 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 infrastructure issues; what makes a workday typical and therefore good is primarily assessed by the match between developers' expectations and reality (Meyer et al., 2021). Two things here relate to our own findings. First, we also find that practitioners who already identify testing as good and meaningful practice, for example because they are motivated by books or blogs about testing, are indeed ambitious and aspirational about testing. Second, we also see that challenges created by infrastructure issues, for example in complicated development environments lead to discouragement because of unexpected behavior. With a survey study conducted in multiple companies Runeson (2006) also found supporting evidence for the negative impact of unexpected challenges caused by complexity. A good integration of unit testing into the internal tool landscape that is provided by the company is key for the adoption of testing. However, this integration is especially hard when the modules under test interact with a complex system state or a complex system environment (Runeson, 2006). When an integration of testing into practice is too challenging it is mostly perceived as de-motivating for software developers. In this context, Daka and Fraser (2014) report that practitioners rank the isolation of testing code as one of most challenging tasks. Crucially, it is perceived as a difficult challenge more often by novice software developers. We see the same in our investigation. Our analysis suggests that inexperienced practitioners are often discouraged from testing by complicated environments in which an isolation of the method under test becomes difficult. On the other hand, aligned with our results, Pham et al. (2014) 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. Complexity can thus, as we saw on Stack Overflow as well, be a motivating factor. Pham et al. (2014) and Daka and Fraser (2014) also report that developers' feelings about unit testing are often negative. Concretely, only half of the practitioners interviewed by Daka and Fraser (2014) had positive feelings about testing and students interviewed by Pham et al. (2014) were not fond of testing because to them writing tests did not feel like an accomplishment. Some students even developed an anxious attitude towards testing. This aligns with our observation in so far that we saw an overwhelming amount of negative posts in random samples. A general negative bias towards testing could therefore also be an explanation for the high amount of negative post that we saw in our dataset.

In relation to Sharp et al. (2009) and Meyer et al. (2021) and their finding that meaningful contributions and being able to make a difference are important. However, from our own work it is not evident that testing in itself is always recognized as a meaningful contribution to projects by practitioners and their peers. Positive ambitions mentioned in posts on Stack Overflow mostly seem to be self-aroused for example through engagement with inspiring resources like books or blogs. Daka and Fraser (2014) indeed 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.

Finally, Kasurinen et al. (2011) investigated how new testing practices are adopted by companies and found out that when confronted with new techniques that could improve testing processes, most companies are not interested in adoption if there is no first-hand knowledge in the team or company. Only rarely they do give new practices a try, and if they do, they only evaluate new techniques in small projects. However, Kasurinen et al. (2011) also report that companies adopt new techniques when

13 Reddit /r/softwaretesting

14 Twitter #softwaretesting

clear need arises. According to the theory they propose in their study, development of processes only happens when the existing process obviously has a need to develop; required resources for adoption of new practices like testing need to be justified. Our preliminary theory has at its core this very point. We observe on Stack Overflow, that an increase of complexity of a project leads to spontaneous adoption of testing practices. While it is not clear from the report of Kasurinen et al. (2011), what the motivation or rational reason of a company that evaluates testing practices in small projects is, a suggestion could be taken from our own study. We suggest that evaluation of techniques in small projects leads to an advantage when the need for those techniques can no longer be ignored. In other words, first-hand knowledge should develop in a company before it is really needed.

7. Conclusion

In this study we set out to understand the sentiments of software engineers regarding software testing in the context of the popular question and answer platform Stack Overflow. In order to do so, we have used a semi-automated approach to detect sentiment in Stack Overflow posts. In particular, we start out by using automatic sentiment analysis tools to classify posts, after which we perform an in-depth, qualitative analysis.

Through this in-depth study of 200 posts we find that developers are in fact sentimental about software testing on Stack Overflow; we find that they express their sentiment when unexpected behavior, uncertainties, complex issues, missing capabilities, or a willingness to improve is part of the post. Additionally, we have observed that lack of experience and knowledge, especially in complex environments can lead to a negative sentiment. On the other hand, software engineers express positive sentiment when they have trust and confidence in their practice, especially if they have an understanding of long term goals of their projects.

Through the observations that we have made, we construct a preliminary interpretive theory that explains how a projects' complexity and the tacit knowledge of individuals shapes the experience and attitude of practitioners in the context of software testing. Practitioners, we argue, get motivated to practice software testing as the complexity of their project increases. Reaching that point without enough knowledge of testing practices leads to discouraging experiences. We argue that testing practices are also seen by practitioners as something to aspire to, especially when considered for example in the context of long term goals. This has implications for both the education of software engineers, and for managing software development teams that engineer complex software. Our findings suggest that taking both motivation and complexity into account in future studies of software testing practices can reveal more about practitioners' sentimental perspectives. Our preliminary results show that an investigation of the motivation and capabilities of software engineers to engage in effective testing practices needs to go beyond the analysis of technical tools and their usage.

We acknowledge that we need to extend and deepen our interpretive theory, and our overall understanding of software engineers' sentiments towards testing. In particular, in our future work we envision to study the social context and its relation to sentiment, the connection to the experience levels of software engineers, their sentimental expressions in informal settings, and finally how project management culture influences attitudes and motivation of individual software engineers in the area of testing.

CRediT authorship contribution statement

Mark Swillus: Data curation, Formal analysis, Investigation, Methodology, Software, Writing – original draft. **Andy Zaidman:** Funding acquisition, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Andy Zaidman reports financial support was provided by Dutch Research Council Domain Science.

Data availability

Data will be made available on request.

Acknowledgment

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

References

Ahmed, Toufique, Bosu, Amiangshu, Iqbal, Anindya, Rahimi, Shahram, 2017. SentiCR: A customized sentiment analysis tool for code review interactions. In: 2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, pp. 106–111. http://dx.doi.org/10.1109/ASE.2017.8115623.

Alshangiti, Moayad, Sapkota, Hitesh, Murukannaiah, Pradeep K., Liu, Xumin, Yu, Qi, 2019. Why is developing machine learning applications challenging? a study on stack overflow posts. In: 2019 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM). IEEE, pp. 1–11. http://dx.doi.org/10.1109/ESEM.2019.8870187.

Aniche, Maurício, Treude, Christoph, Zaidman, Andy, 2022. How developers engineer test cases: An observational study. IEEE Trans. Softw. Eng. 48 (12), 4925–4946. http://dx.doi.org/10.1109/TSE.2021.3129889.

Athanasiou, Dimitrios, Nugroho, Ariadi, Visser, Joost, Zaidman, Andy, 2014. Test code quality and its relation to issue handling performance. IEEE Trans. Softw. Eng. 40 (11), 1100–1125. http://dx.doi.org/10.1109/TSE.2014.2342227.

Balahur, Alexandra, Steinberger, Ralf, 2009. Rethinking sentiment analysis in the news: from theory to practice and back. Proc. WOMSA 9, 1–12.

Baltes, Sebastian, Diehl, Stephan, 2019. Usage and attribution of Stack Overflow code snippets in GitHub projects. Empir. Softw. Eng. 24 (3), 1259–1295. http://dx.doi.org/10.1007/s10664-018-9650-5.

Barua, Anton, Thomas, Stephen W., Hassan, Ahmed E., 2014. What are developers talking about? An analysis of topics and trends in Stack Overflow. Empir. Softw. Eng. 19 (3), 619–654. http://dx.doi.org/10.1007/s10664-012-9231-y.

Barzilay, Ohad, Treude, Christoph, Zagalsky, Alexey, 2013. Facilitating crowd sourced software engineering via stack overflow. In: Finding Source Code on the Web for Remix and Reuse. Springer New York, pp. 289–308. http://dx.doi.org/10.1007/978-1-4614-6596-6_15.

Beecham, Sarah, Baddoo, Nathan, Hall, Tracy, Robinson, Hugh, Sharp, Helen, 2008. Motivation in Software Engineering: A systematic literature review. Inf. Softw. Technol. 50 (9–10), 860–878. http://dx.doi.org/10.1016/j.infsof.2007.09.004.

Begel, Andrew, Zimmermann, Thomas, 2014. Analyze this! 145 questions for data scientists in software engineering. In: Proceedings of the 36th International Conference on Software Engineering. ACM, pp. 12–23. http://dx.doi.org/10.1145/2568225.2568233.

Beller, Moritz, Gousios, Georgios, Panichella, Annibale, Proksch, Sebastian, Amann, Sven, Zaidman, Andy, 2019. Developer testingin the IDE: Patterns, beliefs, and behavior. IEEE Trans. Softw. Eng. 45 (3), 261–284. http://dx.doi.org/10.1109/TSE.2017.2776152, URL https://ieeexplore.ieee.org/document/8116886/.

Beller, Moritz, Gousios, Georgios, Panichella, Annibale, Zaidman, Andy, 2015a. When, how, and why developers (do not) test in their IDEs. In: Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering. In: ESEC/FSE 2015, ACM, pp. 179–190. http://dx.doi.org/10.1145/2786805.2786843.

Beller, Moritz, Gousios, Georgios, Zaidman, Andy, 2015b. How (much) do developers test? In: 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering. 2, pp. 559–562. http://dx.doi.org/10.1109/ICSE.2015.193.

Bertolino, Antonia, 2007. Software testing research: Achievements, challenges, dreams. In: Future of Software Engineering (FOSE '07). pp. 85–103. http://dx.doi.org/10.1109/FOSE.2007.25.

Carstensen, Peter H., Sørensen, Carsten, 1995. Let's talk about bugs!. 7.

Çalıklı, Gül, Bener, Ayşe Başar, 2013. Influence of confirmation biases of developers on software quality: an empirical study. Softw. Qual. J. 21 (2), 377–416. http://dx.doi.org/10.1007/s11219-012-9180-0.

Charmaz, Kathy, 2014. Constructing grounded theory, second ed. In: Introducing Qualitative Methods, Sage.

Daka, Ermira, Fraser, Gordon, 2014. A survey on unit testing practices and problems. In: 2014 IEEE 25th International Symposium on Software Reliability Engineering. IEEE, pp. 201–211. http://dx.doi.org/10.1109/ISSRE.2014.11.

Deener, Andrew, 2018. The architecture of ethnographic knowledge: Narrowing down data and contexts in search of sociological cases. Sociol. Perspect. 61 (2), 295–313. http://dx.doi.org/10.1177/0731121418755121.

Ford, Denae, Smith, Justin, Guo, Philip J., Parnin, Chris, 2016. Paradise unplugged: identifying barriers for female participation on stack overflow. In: Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, pp. 846–857. http://dx.doi.org/10.1145/2950290.2950331.

Geertz, Clifford, Darnton, Robert, 2017. The interpretation of cultures: selected essays, third ed. In: Basic book-s, Basic Books.

Glaser, Barney G., Strauss, Anselm L., 2010. The Discovery of Grounded Theory: Strategies for Qualitative Research, 5. paperback print Aldine Transaction.

Graziotin, Daniel, Fagerholm, Fabian, Wang, Xiaofeng, Abrahamsson, Pekka, 2018. What happens when software developers are (un)happy. J. Syst. Softw. 140, 32–47. http://dx.doi.org/10.1016/j.jss.2018.02.041.

Hetzel, William C., 1988. The Complete Guide to Software Testing, second ed. QED Information Sciences.

Hine, Christine, 2008. Virtual ethnography: Modes, varieties, affordances. In: The SAGE Handbook of Online Research Methods. SAGE Publications, Ltd, pp. 257–270. http://dx.doi.org/10.4135/9780857020055.n14.

Hoda, Rashina, 2022. Socio-technical grounded theory for software engineering. IEEE Trans. Softw. Eng. 48 (10), 3808–3832. http://dx.doi.org/10.1109/TSE.2021.3106280.

Indurkhya, Nitin, Damerau, Fred J. (Eds.), 2010. Handbook of Natural Language Processing. Chapman and Hall/CRC, http://dx.doi.org/10.1201/9781420085938.

Islam, Md Rakibul, Ahmmed, Md Kauser, Zibran, Minhaz F., 2019. MarValous: machine learning based detection of emotions in the valence-arousal space in software engineering text. In: Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing. ACM, pp. 1786–1793. http://dx.doi.org/10.1145/3297280.3297455.

Islam, Md Rakibul, Zibran, Minhaz F., 2018. DEVA: sensing emotions in the valence arousal space in software engineering text. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. ACM, pp. 1536–1543. http://dx.doi.org/10.1145/3167132.3167296.

Jongeling, Robbert, Sarkar, Proshanta, Datta, Subhajit, Serebrenik, Alexander, 2017. On negative results when using sentiment analysis tools for software engineering research. Empir. Softw. Eng. 22 (5), 2543–2584. http://dx.doi.org/10.1007/s10664-016-9493-x.

Kasurinen, Jussi, Taipale, Ossi, Smolander, Kari, 2009. Analysis of problems in testing practices. In: 2009 16th Asia-Pacific Software Engineering Conference. IEEE, pp. 309–315. http://dx.doi.org/10.1109/APSEC.2009.17.

Kasurinen, Jussi, Taipale, Ossi, Smolander, Kari, 2011. How test organizations adopt new testing practices and methods? In: 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops. IEEE, pp. 553–558. http://dx.doi.org/10.1109/ICSTW.2011.63, URL http://ieeexplore.ieee.org/document/5954463/.

Ko, Amy J., Dosono, Bryan, Duriseti, Neeraja, 2014. Thirty years of software problems in the news. In: Proceedings of the 7th International Workshop on Cooperative and Human Aspects of Software Engineering. In: CHASE 2014, ACM, pp. 32–39. http://dx.doi.org/10.1145/2593702.2593719.

Lee, Roy Ka-Wei, Lo, David, 2017. GitHub and stack overflow: Analyzing developer interests across multiple social collaborative platforms. In: Ciampaglia, Giovanni Luca, Mashhadi, Afra, Yasseri, Taha (Eds.), Social Informatics. Springer International Publishing, pp. 245–256. http://dx.doi.org/10.1007/978-3-319-67256-4_19.

Lin, Bin, Cassee, Nathan, Serebrenik, Alexander, Bavota, Gabriele, Novielli, Nicole, Lanza, Michele, 2022. Opinion mining for software development: A systematic literature review. ACM Trans. Softw. Eng. Methodol. 31 (3), 38:1–38:41. http://dx.doi.org/10.1145/3490388.

Lin, Bin, Zampetti, Fiorella, Bavota, Gabriele, Di Penta, Massimiliano, Lanza, Michele, Oliveto, Rocco, 2018. Sentiment analysis for software engineering: how far can we go? In: Proceedings of the 40th International Conference on Software Engineering. ACM, pp. 94–104. http://dx.doi.org/10.1145/3180155.3180195.

Liu, Yinhan, Ott, Myle, Goyal, Naman, Du, Jingfei, Joshi, Mandar, Chen, Danqi, Levy, Omer, Lewis, Mike, Zettlemoyer, Luke, Stoyanov, Veselin, 2019. RoBERTa: A robustly optimized BERT pretraining approach. arXiv:1907.11692 [cs], URL http://arxiv.org/abs/1907.11692.

Lopez, Tamara, Tun, Thein T., Bandara, Arosha, Levine, Mark, Nuseibeh, Bashar, Sharp, Helen, 2018. An investigation of security conversations in stack overflow: perceptions of security and community involvement. In: Proceedings of the 1st International Workshop on Security Awareness from Design To Deployment. ACM, pp. 26–32. http://dx.doi.org/10.1145/3194707.3194713.

Lopez, Tamara, Tun, Thein, Bandara, Arosha, Mark, Levine, Nuseibeh, Bashar, Sharp, Helen, 2019. An anatomy of security conversations in stack overflow. In: 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS). IEEE, pp. 31–40. http://dx.doi.org/10.1109/ICSE-SEIS.2019.00012.

Mäntylä, Mika V., Itkonen, Juha, Iivonen, Joonas, 2012. Who tested my software? Testing as an organizationally cross-cutting activity. Softw. Qual. J. 20 (1), 145–172. http://dx.doi.org/10.1007/s11219-011-9157-4.

Mantyla, Mika V., Novielli, Nicole, Lanubile, Filippo, Claes, Maelick, Kuutila, Miikka, 2017. Bootstrapping a lexicon for emotional arousal in software engineering. In: 2017 IEEE/ACM 14th International Conference on Mining Software Repositories (MSR). IEEE, pp. 198–202. http://dx.doi.org/10.1109/MSR.2017.47.

McCracken, Harry, 2017. The Year That Software Bugs Ate The World. Fast Company, URL https://web.archive.org/web/20230307155438/https://www.fastcompany.com/40505226/the-year-that-software-bugs-ate-the-world.

McDonald, Nora, Schoenebeck, Sarita, Forte, Andrea, 2019. Reliability and inter-rater reliability in qualitative research: Normsand guidelines for CSCW and HCI practice. Proc. ACM Hum.-Comput. Interact. 3 (CSCW), 72:1–72:23. http://dx.doi.org/10.1145/3359174.

Mead, George Herbert, Morris, Charles W., Huebner, Daniel R., Joas, Hans, 2015. Mind, Self, and Society, The definitive edition University of Chicago Press.

Meyer, André, Barr, Earl T, Bird, Christian, Zimmermann, Thomas, 2021. Today was a good day: The daily life of software developers. IEEE Trans. Softw. Eng. 47 (5), 863–880. http://dx.doi.org/10.1109/TSE.2019.2904957.

Mohanani, Rahul, Salman, Iflaah, Turhan, Burak, Rodriguez, Pilar, Ralph, Paul, 2020. Cognitive biases in software engineering: A systematic mapping study. IEEE Trans. Softw. Eng. 46 (12), 1318–1339. http://dx.doi.org/10.1109/TSE.2018.2877759.

Mustafa, Sohaib, Zhang, Wen, Naveed, Muhammad Mateen, 2022. What motivates online community contributors to contribute consistently? A case study on Stackoverflow netizens. Curr. Psychol. http://dx.doi.org/10.1007/s12144-022-03307-4.

Myers, Glenford J., Sandler, Corey, Badgett, Tom, 2012. The Art of Software Testing, third ed. John Wiley & Sons.

Novielli, Nicole, Calefato, Fabio, Lanubile, Filippo, 2018. A gold standard for emotion annotation in stack overflow. In: Proceedings of the 15th International Conference on Mining Software Repositories. MSR '18, ACM, pp. 14–17. http://dx.doi.org/10.1145/3196398.3196453.

Pelicon, Andraž, Pranjić, Marko, Miljković, Dragana, Škrlj, Blaž, Pollak, Senja, 2020. Zero-shot learning for cross-lingual news sentiment classification. Appl. Sci. 10 (17), 5993. http://dx.doi.org/10.3390/app10175993.

Pham, Raphael, Kiesling, Stephan, Liskin, Olga, Singer, Leif, Schneider, Kurt, 2014. Enablers, inhibitors, and perceptions of testing in novice software teams. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. In: FSE 2014, ACM, pp. 30–40. http://dx.doi.org/10.1145/2635868.2635925.

Prado, Marllos Paiva, Vincenzi, Auri Marcelo Rizzo, 2018. Towards cognitive support for unit testing: A qualitative study with practitioners. J. Syst. Softw. 141, 66–84. http://dx.doi.org/10.1016/j.jss.2018.03.052.

Runeson, P., 2006. A survey of unit testing practices. IEEE Softw. 23 (4), 22–29. http://dx.doi.org/10.1109/MS.2006.91.

Saeed, Hafiz Hassaan, Shahzad, Khurram, Kamiran, Faisal, 2018. Overlapping toxic sentiment classification using deep neural architectures. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW). pp. 1361–1366. http://dx.doi.org/10.1109/ICDMW.2018.00193.

Saldaña, Johnny, 2013. The Coding Manual for Qualitative Researchers, second ed. SAGE.

Salman, Iflaah, Rodriguez, Pilar, Turhan, Burak, Tosun, Ayse, Gureller, Arda, 2022. What leads to a confirmatory or disconfirmatory behaviour of software testers? IEEE Trans. Softw. Eng. 48 (4), 1351–1368. http://dx.doi.org/10.1109/TSE.2020.3019892.

Sengupta, Subhasree, Haythornthwaite, Caroline, 2020. Learning with comments: An analysis of comments and community on Stack Overflow. In: Proceedings of the 53rd Hawaii International Conference on System Sciences. URL https://core.ac.uk/reader/286030414.

Sharp, Helen, Baddoo, Nathan, Beecham, Sarah, Hall, Tracy, Robinson, Hugh, 2009. Models of motivation in software engineering. Inf. Softw. Technol. 51 (1), 219–233. http://dx.doi.org/10.1016/j.infsof.2008.05.009, URL https://linkinghub.elsevier.com/retrieve/pii/S0950584908000827.

Sharp, H., Robinson, H., Woodman, M., 2000. Software engineering: community and culture. IEEE Softw. 17 (1), 40–47. http://dx.doi.org/10.1109/52.819967.

Swillus, Mark, Zaidman, Andy, 2022. Replication package for sentiment overflow in the testing stack. http://dx.doi.org/10.5281/zenodo.6595110.

Vadlamani, Sri Lakshmi, Baysal, Olga, 2020. Studying software developer expertise and contributions in stack overflow and GitHub. In: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, pp. 312–323. http://dx.doi.org/10.1109/ICSME46990.2020.00038.

Wang, Shaohua, Phan, NhatHai, Wang, Yan, Zhao, Yong, 2019. Extracting API tips from developer question and answer websites. In: 2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR). pp. 321–332. http://dx.doi.org/10.1109/MSR.2019.00058.

Wiebe, Janyce M., Bruce, Rebecca F., O'Hara, Thomas P., 1999. Development and use of a gold-standard data set for subjectivity classifications. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics. Association for Computational Linguistics, pp. 246–253. http://dx.doi.org/10.3115/1034678.1034721.

Yang, Xin-Li, Lo, David, Xia, Xin, Wan, Zhi-Yuan, Sun, Jian-Ling, 2016. What security questions do developers ask? A large-scale study of stack overflow posts. J. Comput. Sci. Tech. 31 (5), 910–924. http://dx.doi.org/10.1007/s11390-016-1672-0.

Yourdon, Edward, 1988. Managing the system life cycle, second ed. In: Yourdon Press Computing Series, Yourdon Press.

Zagalsky, Alexey, Teshima, Carlos Gómez, German, Daniel M., Storey, Margaret-Anne, Poo-Caamaño, Germán, 2016. How the R community creates and curates knowledge: a comparative study of stack overflow and mailing lists. In: Proceedings of the 13th International Conference on Mining Software Repositories. ACM, pp. 441–451. http://dx.doi.org/10.1145/2901739.2901772.

Zaidman, Andy, Van Rompaey, Bart, Demeyer, Serge, van Deursen, Arie, 2008. Mining software repositories to study co-evolution of production & test code. In: 1st International Conference on Software Testing, Verification and Validation 2008. pp. 220–229. http://dx.doi.org/10.1109/ICST.2008.47.

Zaidman, Andy, Van Rompaey, Bart, van Deursen, Arie, Demeyer, Serge, 2011. Studying the co-evolution of production and test code in open source and industrial developer test processes through repository mining. Empir. Softw. Eng. 16 (3), 325–364. http://dx.doi.org/10.1007/s10664-010-9143-7.

Zhang, Ting, Xu, Bowen, Thung, Ferdian, Haryono, Stefanus Agus, Lo, David, Jiang, Lingxiao, 2020. Sentiment analysis for software engineering: How far can pre-trained transformer models go? In: 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME). IEEE, pp. 70–80. http://dx.doi.org/10.1109/ICSME46990.2020.00017.

Mark Swillus, is a Ph.D. candidate in software engineering at Delft University of Technology, The Netherlands. He received his M.Eng in Information Systems Engineering from the University of Applied Sciences in Aachen, Germany, in 2018. He currently researches the interplay of social and technical aspects of software engineering, focusing on software quality and testing practices. Having worked as a software engineer for multiple open source software projects, he is particularly interested in the diverse human aspects of cumbersome processes that lead to a finished software product.

Andy Zaidman, is a Full Professor in software engineering at Delft University of Technology, The Netherlands. He received his M.Sc. and Ph.D. degrees in Computer Science from the University of Antwerp, Belgium, in 2002 and 2006, respectively. His main research interests include software evolution, program comprehension, mining software repositories, software quality, and software testing. He is an active member of the research community and involved in the organization of numerous conferences (WCRE'08, WCRE'09, VISSOFT'14 and MSR'18). In 2013 he was the laureate of a prestigious Vidi midcareer grant, while in 2019 he received the most prestigious Vici career grant from the Dutch science foundation NWO.