Sentiment in Software Engineering: Detection and Application

Nathan Cassee
n.w.cassee@tue.nl
Eindhoven University of Technology
Eindhoven, The Netherlands

ABSTRACT

In software engineering the role of human aspects is an important one, especially as developers indicate that they experience a wide range of emotions while developing software. Within software engineering researchers have sought to understand the role emotions and sentiment play in the development of software by studying issues, pull-requests and commit messages. To detect sentiment, automated tools are used, and in this doctoral thesis we plan to study the use of these sentiment analysis tools, their applications, best practices for their usage and the effect of non-natural language on their performance. In addition to studying the application of sentiment analysis tools, we also aim to study self-admitted technical debt and bots in software engineering, to understand why developers express sentiment and what they signal when they express sentiment. Through studying both the application of sentiment analysis tools and the role of sentiment in software engineering, we hope to provide practical recommendations for both researchers and developers.

CCS CONCEPTS

• **Human-centered computing → Empirical studies in collaborative and social computing**.

KEYWORDS

software engineering, human aspects, sentiment, sentiment analysis

ACM Reference Format:
Nathan Cassee. 2022. Sentiment in Software Engineering: Detection and Application. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE '22), November 14–18, 2022, Singapore, Singapore.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3540250.3558908

1 INTRODUCTION

In modern software engineering an important role is fulfilled by open-source software projects. The developers and maintainers of open-source software develop and maintain libraries and software packages that are used by many software applications in both open and closed-source settings. Because so many software projects depend on open-source software, the continued sustainability and health of open-source software projects is increasingly important as open-source software projects are sometimes abandoned [4]. Additionally, while developing software, software engineers self-describe that they experience a wide range of emotions [43]. The way in which emotions are experienced, and their theoretical fundaments, have been extensively studied, both outside of software engineering [14, 35] and within [3, 10, 34, 39]. For instance, Asri *et al.* found that code reviews in which negative emotions were expressed take longer to complete [3]. Meanwhile, Ortu *et al.* found that issues with negative emotions take longer to resolve [34].

The study of emotions and sentiment experienced and expressed by developers can hopefully help researchers develop tools and techniques to support developers with the management of open-source projects [33]. In this doctoral thesis we aim to further the understanding of human aspects in software engineering, in particular by focussing on the study of emotions and sentiment in Software Engineering. Therefore, we pose the following research question:

Research Question

How can the detection of sentiment and emotions expressed by developers be used to improve software development?

As previously stated: The study of sentiment and emotions in Software Engineering is incredibly broad, and it ranges from studies of burnout [27, 37] to studies of sentiment in commit messages [38]. Since it has been found for instance that expression of negative sentiment might negatively affect software development [3, 34] we seek to study two specific areas in the context of this thesis: we study existing sentiment analysis tools and the ways in which they are employed by researchers. Secondly, we attempt understand how sentiment affects software developers through studying the ways in which developers express themselves while developing software.

2 RESEARCH BACKGROUND

Researchers outside of software engineering agree that the study of emotions and sentiment in the workplace is important [2]. According to Posner *et al.* the emotions experienced by humans (fear, anger, happiness, *etc.*) are a continuous function that can be modeled as a combination of valence and arousal [35]. Where valence captures the polarity of the emotion (positive or negative) and arousal the excitement of the emotion (high or low).

Within software engineering emotions and sentiments have been extensively studied. Researchers have used both automatic and manual methods to detect emotions and sentiment expressed by developers. Girardi *et al.* have used biometric sensors to detect emotions experienced by software developers in the workplace [16, 17],

ESEC/FSE '22, November 14–18, 2022, Singapore, Singapore
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9413-0/22/11.
https://doi.org/10.1145/3540250.3558908

while Hermann and Klunder studied the sentiment expressed by students in in-person meetings [19]. Mäntylä *et al.* studied the valence and arousal observed in issue trackers and proposes that it can be used to detect whether developers might burn-out [27]. Raman *et al.* attempt to solve the problem of burn-out by developing a detector that can detect toxic interactions in open-source repositories [37]. While studying the sentiment expressed by developers who perform refactoring tasks, Singh and Singh found that developers are more likely to express negative sentiment in the commit messages of refactoring commits [38].

Researchers have used a wide variety of tools to detect sentiment. These tools vary from dictionary based approaches [1] to machine learning based approaches [7, 12, 20] and deep learning tools [9, 46]. In addition to employing these tools, researchers have also extensively studied them to understand how best to employ these sentiment analysis tools. Jongeling *et al.* found that general-purpose sentiment analysis tools do not work well for software engineering data [21]. In a more recent benchmarking study Novielli *et al.* came to the conclusion that even sentiment-analysis tools tailored for software engineering do not generalize across different domains [31]. For instance, sentiment analysis tools trained on GitHub data do not perform well on StackOverflow data. In a follow-up study Novielli *et al.* found that the application of software-engineering specific sentiment analysis tools off-the-shelf might introduce a threat to conclusion validity [32]. While researchers have sought to understand how to best apply sentiment analysis tools, it is still unknown whether the predictive performance of sentiment analysis tools can be further improved.

3 TOOLING FOR SENTIMENT ANALYSIS

The first area we discuss in the doctoral thesis is the application of sentiment analysis tools to software engineering. As discussed in Section 2 a wide variety of studies has sought to apply sentiment analysis and emotion detection to the analysis of software engineering data. However, a comprehensive overview of the different studies, sentiment analysis tools and limitations discussed in these studies does not exist. Therefore we pose:

RQ_1: *How are sentiment analysis tools applied by researchers, and what are their conclusions?*

By studying the applications of sentiment analysis tools we hope to get a better overview of the different tools, their limitations, and the way in which these tools have been applied by researchers. This overview can be used by researchers to understand what tools exist, and how these tools have been evaluated and applied.

Researchers have extensively studied the limitations of the application of sentiment analysis tools for software engineering [21, 31, 32]. Several recommendations on how sentiment analysis should be applied to software engineering data exist [21, 31]. Additionally, while replicating software engineering studies that derive conclusions related to sentiment Novielli *et al.* found that the application of out-of-the-box sentiment analysis tools threatens conclusion validity [32]. However, there is no single place where all recommendations, implications of the recommendations, and evidence for these recommendations is gathered. Therefore we pose:

RQ_2: *What are best practices for the application of sentiment analysis tools to software engineering data?*

By studying this question we hope to validate and gather the latest recommendations on the application of sentiment analysis tools for software engineering. Moreover, where possible we will try to verify these recommendation. Ideally this will help researchers derive reliable conclusions when applying sentiment analysis tools.

Data gathered from social coding platforms for sentiment analysis often contains non-natural language [5, 30]. Examples of non-natural language are URLs, references to pull-requests (#432), *etc.* Efstathiou and Spinellis argue that these non-natural language tokens should be replaced with meta-tokens that capture the type of the non-natural language elements [13]. The strategies used by sentiment analysis tools to address non-natural language, and the effect that meta-tokens might have on the performance of sentiment analysis tools has not yet been studied, therefore we pose:

RQ_3: *How do existing sentiment analysis tools address non-natural language and does meta-tokenization of non-natural language affect their performance?*

If the replacement of non-natural language elements through meta-tokenization improves the performance of sentiment analysis tools, this technique can be used to further improve sentiment analysis, and ensure that conclusions derived from the application of sentiment analysis tools are accurate.

3.1 Approach

To answer *RQ_1* and *RQ_2* we perform a systematic literature review (SLR) according to the guidelines of Kitchenham and Charters [22]. For both RQs we select studies that are peer-reviewed academic studies, published at respectable venues who are related to software development and apply automated sentiment analysis tools. For *RQ_2* we further constrain our search space and we only select studies that benchmark, compare, or discuss recommendations for the application of sentiment analysis tools for software engineering. Secondly, where possible we will replicate the experiments on which the replications in the source-paper are based.

To address *RQ_3* we benchmark state of the art sentiment analysis, and we evaluate them in an experimental setup similar to the one used by Novielli *et al.* in their benchmark study [31]. We first study the existing preprocessing techniques used by the state-of-the-art sentiment analysis tools. We then take existing datasets for sentiment analysis in software engineering and in these datasets we manually identify all non-natural language elements. We do this by sampling a small number of items from a dataset, and we manually annotate all non-natural language elements. Based this manual identification of non-natural language elements, we create a meta-tokenized version of each dataset in which we replace the identified non-natural language elements. To assess whether there is a difference in performance, we compare the performance of sentiment analysis tools trained on original and meta-tokenized version of each dataset. For this study, performance is measured using metrics like precision, recall and f1 scores, together with the agreement of the sentiment analysis tools with each other.

3.2 Findings

The study answering *RQ_1* has been published at TOSEM [23]. A total of 185 primary studies were found that match the specified inclusion criteria. From the found studies we conclude that opinion mining techniques have been applied to a wide variety of topics, from requirements, to team management and software design. Moreover, from the SLR we also learn that a wide variety of sentiment analysis and other opinion mining tools have been used in the literature. Additionally, there has also been a large amount of benchmarks, especially for sentiment analysis tools, that compare these tools with each other to determine which tools are more accurate.

The studies answering *RQ_2* and *RQ_3* are still in progress and have not been published yet.

4 APPLICATIONS OF SENTIMENT AND EMOTIONS

From the SLR on the application of sentiment analysis and opinion mining we know that a lot of work has sought to understand the emotions expressed by developers [24]. However, so far no research has focused on understanding the role of sentiment in Self-admitted technical debt (SATD) and software bots. Within this doctoral thesis we seek to continue this work by focussing on both these areas. For both of these areas we seek to understand whether and how developers use and interpret the expression of negative sentiment.

4.1 Self-Admitted Technical Debt:

Self-Admitted Technical Debt (SATD) is the study of technical debt that is admitted by developers in source-code [36]. While expressing SATD, developers describe a wide variety of issues. These issues include items such as code debt, design debt, requirement debt, *etc.* [6, 26]. When developers record SATD, they might also express negative sentiment in the source-code comment. Developers might use the expression of negative sentiment as a proxy for priority, or importance. However, as of yet it is unclear whether developers express negative sentiment in SATD, whether they interpret negative sentiment as a proxy for priority. Therefore we pose:

***RQ_4*:** *What role does negative sentiment play in the expression of Self-Admitted Technical Debt?*

As SATD is frequently expressed in source-code comments a better understanding of whether and how negative developers express themselves might help maintainers identify high priority SATD items. Additionally, by understanding the perceptions of developers towards negative sentiment in SATD we can better understand what developers attempt to convey when they express negative sentiment in SATD and whether and how maintainers should prioritize SATD in which negative sentiment is expressed.

4.1.1 Background. The concept of Self-Admitted Technical Debt (SATD) was first described by Potdar and Shibab, who found that SATD occurs in up-to 30% of source-code files [36]. Several researchers have studied the content of SATD. Both Maldonado and Shibab and Bavota and Russo have studied the content of SATD, and they categorize SATD based on where in the software-development lifecycle the technical debt occurs [6, 26]. Zampetti *et al.* studied the differences between SATD admitted in open-source and closed-source, finding that admitting SATD in closed-source might be implicitly discouraged [45]. While SATD is often studied in source-code Xavier *et al.* studied whether and how SATD is reported in issue trackers, finding that only 29% of SATD in issue trackers can also be found in source-code [44].

4.1.2 Approach. To address *RQ_4* we analyze two different data-sources. We manually label existing SATD instances from open-source software projects to record whether negative sentiment is present. Secondly, to understand whether the type of SATD issue described in a comment influences whether comments are more likely to be negative we analyze the distribution of sentiment polarity per SATD category. While studying existing comments allows us to determine whether developers express negative sentiment this does not allow us to understand how developers interpret and perceive negative sentiment in SATD. Therefore, we expand our study of existing SATD comments with a survey targeting open-source developers. In this survey we:

— ask developers whether they express negative sentiment when writing SATD,
— whether they believe the expression of negative sentiment in SATD is acceptable,
— whether they interpret negative sentiment in SATD as a proxy for priority, and
— we present several vignettes [28], that have been inspired by existing SATD items, to the respondents and ask them to draft a SATD comment for the particular item.

Using questions on perceptions and believes we can better understand what developers seek to convey when they author comments containing negative SATD. Additionally, through the vignettes we can compare the occurrence of negative sentiment in SATD for the comments written based on the vignettes, and the comments extracted from source-code. Through a comparison of the distribution of sentiment polarity per SATD category we can hopefully better understand what role negative sentiment plays in SATD.

4.1.3 Findings. The study in which we address *RQ_4* has been accepted for publication at the Special Issue of MSR 2021 of EMSE [8]. In this work we analyzed an existing dataset of 1038 SATD instances published by Maldonado *et al.* [25] and we combined this analysis with a survey of 46 open-source developers.

After manually labeling the SATD comments from Maldonado *et al.* we find that negative sentiment is expressed in SATD. However, the occurrence of negative sentiment varies significantly between SATD categories. For instance, negative sentiment is more common in SATD describing functional issues such as bugs, compared to SATD describing partially implemented functionality or poor implementation choices.

From the survey with developers we learn that roughly 30% of the respondents interpret negative sentiment in SATD as a proxy for priority. However, this is contrasted by a slightly larger group of respondents who do not interpret negative sentiment in SATD as a proxy for priority. When asked whether developers believe if negative sentiment is an acceptable practice to indicate priority we find that the majority of respondents does not believe expressing negative sentiment is acceptable. Moreover, about half of the respondents indicates that SATD should not be recorded in source-code but instead should be recorded in issue trackers. In response to the vignettes for which respondents were asked to draft SATD comments we find that in the setting of the survey respondents are more likely to be neutral than in the existing SATD comments from Maldonado *et al.*.

4.2 Bots

The adoption of bots by software engineers has not been without concerns. Various studies have found that developers are affected and distracted by the noise of adopting a bot in software projects [29, 40, 42]. One possible signal that software engineers can give to indicate that they dislike certain bot actions are GitHub reactions. From previous work by Farah *et al.* we know that the actions of bots on GitHub are more likely to be disliked [15]. However, currently it is not clear whether disliked bot actions are disliked because the bot introduced noise, or because of other factors. Therefore we pose:

> ***RQ_5:*** *How do developers react to, and evaluate, the actions of bots?*

With RQ_5 we seek to understand how certain characteristics of bots influence the reactions of software engineers towards the actions of bots. Moreover, we seek to understand what developers seek to achieve when they react negatively to bot interactions. Do they just seek to express their annoyance, or do they hope to enact change by disliking the actions of bots?

4.2.1 Background. Since bots have been adopted by software developers they have been studied and characterized by researchers. Storey and Zagalsky outlined and described the role bots fullfil in software engineering projects [40]. Wessel *et al.* found that developers use bots for a variety of purposes, from maintenance tasks, to the reviewing of pull requests and welcoming newcomers [41]. Dey *et al.* focus on bots that commit to repositories, and find that bots which commit code often change documentation and web-pages, as opposed to the actual source-code. [11].

With the adoption of bots, researchers have also sought to understand how the adoption of bots impacts and affects developers. Mirhosseini *et al.* found that software projects which adopt bots to notify developers of out of data dependencies get overwhelmed by the amount of notifications that are sent by these bots [29]. Wessel *et al.* further studied bots and found that developers identify the noise introduced by bots as a recurrent problem [42].

4.2.2 Approach. To address RQ_5 we take an existing dataset of manually validated bot accounts published by Golzadeh *et al.* and we collect all comments in pull-requests and issues for the repositories in the dataset [18]. In addition to mining the comment we also mine all reactions (thumbs ups, thumbs downs, *etc.*) that users on GitHub have given on the comments.

Using this data we aim to understand the bot actions that gather a large amount of negative reactions. By qualitatively studying these bot actions, the context surrounding the comments, and users that give the reactions we hope to better understand the ways in which bot design might influence developer perception of the bots. Additionally, we also aim to understand who responds to bot actions. Are they project members of the project itself? Or are they outsiders? Secondly, using a survey we also seek to study what developers attempt to convey when they respond negatively to bot actions. Using this survey, we seek to understand the motivations of developers for responding to bot accounts, and the message they seek to convey.

4.2.3 Findings. The study through which we seek to address RQ_5 is currently in progress and no results have been obtained yet.

5 CONCLUSIONS AND FUTURE WORK

While developing software developers describe that they experience emotions [43]. Researchers have sought to understand the ways in which developers express emotions by studying a variety of software development activities [3, 10, 34, 39].

For this doctoral thesis we plan to focus on how researchers apply tools that can be used to detect sentiment in software engineering data, and we aim to study how developers apply these tools (*RQ_1*), what the best practices for the usage of sentiment analysis tools are (*RQ_2*), and whether meta-tokenization of non-natural language improves performance (*RQ_3*). By studying these three research questions we aim to provide researchers with practical recommendations on how to reliably apply sentiment analysis tools.

In addition to studying the usage of sentiment analysis tools in academic studies we also aim to understand why and how developers express sentiment when contributing to open-source software. For this, we plan to focus on two areas of software engineering: we seek to study the expression of negative sentiment in SATD (*RQ_4*), in an attempt to understand how developers interpret and perceive negative sentiment in SATD. Secondly we aim to study how developers react to bot actions on GitHub (*RQ_5*), in an attempt to understand what bot actions developers negatively react to, and what developers seek to signal using these reactions.

In the context of this doctoral thesis the work for *RQ_1* and *RQ_4* have been published at TOSEM and EMSE respectively [8, 24]. The work for the other research questions (*RQ_2*, *RQ_3*, & *RQ_5*) are currently in progress.

REFERENCES

[1] Toufique Ahmed, Amiangshu Bosu, Anindya Iqbal, and Shahram Rahimi. 2017. SentiCR: A customized sentiment analysis tool for code review interactions. *ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering* (2017), 106–111.

[2] Neal M. Ashkanasy. 2004. Emotion and Performance. *Human Performance* 17 (4 2004), 137–144. Issue 2.

[3] Ikram El Asri, Noureddine Kerzazi, Gias Uddin, Foutse Khomh, and M. A. Janati Idrissi. 2019. An empirical study of sentiments in code reviews. *Information and Software Technology* 114, October 2018 (2019), 37–54.

[4] Guilherme Avelino, Eleni Constantinou, Marco Tulio Valente, and Alexander Serebrenik. 2019. On the abandonment and survival of open source projects: An empirical investigation. *International Symposium on Empirical Software Engineering and Measurement* 2019-Septemer (2019). arXiv:1906.08058

[5] Alberto Bacchelli, Marco D'Ambros, and Michele Lanza. 2010. Extracting Source Code from E-Mails. *2010 IEEE 18th International Conference on Program Comprehension*, 24–33.

[6] Gabriele Bavota and Barbara Russo. 2016. A large-scale empirical study on self-admitted technical debt. *Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016* (2016), 315–326.

[7] Fabio Calefato, Filippo Lanubile, Federico Maiorano, and Nicole Novielli. 2018. Sentiment Polarity Detection for Software Development. *Empirical Software Engineering* 23 (6 2018), 1352–1382. Issue 3.

[8] Nathan Cassee, Fiorella Zampetti, Nicole Novielli, Alexander Serebrenik, and Massimiliano Di Penta. 2022. Self-Admitted Technical Debt and Comments' Polarity: An Empirical Study. *Empirical Software Engineering* (2022).

[9] Zhenpeng Chen, Yanbin Cao, Xuan Lu, Qiaozhu Mei, and Xuanzhe Liu. 2019. SEntiMoji: An emoji-powered learning approach for sentiment analysis in software engineering. *ESEC/FSE 2019 - Proceedings of the 2019 27th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering* (2019), 841–852.

[10] Jonathan Cheruvelil and Bruno C. Da Silva. 2019. Developers' sentiment and issue reopening. *Proceedings - 2019 IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering, SEmotion 2019* (2019), 29–33.

[11] Tapajit Dey, Sara Mousavi, Eduardo Ponce, Tanner Fry, Bogdan Vasilescu, Anna Filippova, and Audris Mockus. 2020. Detecting and Characterizing Bots that Commit Code. *Proceedings - 2020 IEEE/ACM 17th International Conference on Mining Software Repositories, MSR 2020* (2020), 209–219.

[12] Jin Ding, Hailong Sun, Xu Wang, and Xudong Liu. 2018. Entity-level sentiment analysis of issue comments. *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, 7–13.

[13] Vasiliki Efstathiou and Diomidis Spinellis. 2018. Code review comments. *ICSE-NIER 2018*, 69–72.

[14] Paul Ekman. 2005. Basic Emotions. , 45-60 pages.

[15] Juan Carlos Farah, Basile Spaenlehauer, Xinyang Lu, Sandy Ingram, and Denis Gillet. 2022. *An Exploratory Study of Reactions to Bot Comments on GitHub*. Vol. 1. Association for Computing Machinery.

[16] Daniela Girardi, Filippo Lanubile, Nicole Novielli, Luigi Quaranta, and Alexander Serebrenik. 2019. Towards Recognizing the Emotions of Developers Using Biometrics: The Design of a Field Study. *2019 IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, 13–16.

[17] Daniela Girardi, Nicole Novielli, Davide Fucci, and Filippo Lanubile. 2020. Recognizing developers' emotions while programming. *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, 666–677.

[18] Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. 2021. A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. *Journal of Systems and Software* 175 (2021), 110911.

[19] Marc Herrmann and Jil Klunder. 2021. From Textual to Verbal Communication: Towards Applying Sentiment Analysis to a Software Project Meeting. *REW 2021*, 371–376.

[20] Md Rakibul Islam and Minhaz F. Zibran. 2018. SentiStrength-SE: Exploiting domain specificity for improved sentiment analysis in software engineering text. *Journal of Systems and Software* 145 (2018), 125–146. Issue July 2017.

[21] Robbert Jongeling, Proshanta Sarkar, Subhajit Datta, and Alexander Serebrenik. 2017. On negative results when using sentiment analysis tools for software engineering research. *Empirical Software Engineering* 22, 5 (2017), 2543–2584.

[22] Barbara Kitchenham and Stuart Charters. 2007. Guidelines for performing Systematic Literature Reviews in Software Engineering.

[23] Bin Lin, Nathan Cassee, Alexander Serebrenik, Gabriele Bavota, Nicole Novielli, and Michele Lanza. 2022. Opinion Mining for Software Development: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology* 31 (7 2022), 1–41. Issue 3.

[24] Bin Lin, Nathan Cassee, Alexander Serebrenik, Gabriele Bavota, Nicole Novielli, and Michele Lanza. 2022. Opinion Mining for Software Development: A Systematic Literature Review. *ACM Trans. Softw. Eng. Methodol.* 31, 3, Article 38 (mar 2022), 41 pages.

[25] Everton Da S. Maldonado, Rabe Abdalkareem, Emad Shihab, and Alexander Serebrenik. 2017. An Empirical Study on the Removal of Self-Admitted Technical Debt. *2017 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 238–248.

[26] Everton Da S. Maldonado and Emad Shihab. 2015. Detecting and quantifying different types of self-admitted technical Debt. *2015 IEEE 7th International Workshop on Managing Technical Debt, MTD 2015 - Proceedings* (2015), 9–15.

[27] Mika Mäntyl, Bram Adams, Giuseppe Destefanis, Daniel Graziotin, and Marco Ortu. 2016. Mining Valence, arousal, and Dominance - Possibilities for detecting burnout and productivity? *Proceedings - 13th Working Conference on Mining Software Repositories, MSR 2016* (2016), 247–258.

[28] Andrew McNamara, Justin Smith, and Emerson Murphy-Hill. 2018. Does ACM's code of ethics change ethical decision making in software development? *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 729–733.

[29] Samim Mirhosseini and Chris Parnin. 2017. Can automated pull requests encourage software developers to upgrade out-of-date dependencies? *ASE 2017 - Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering* (2017), 84–94.

[30] Mika V. Mäntylä, Fabio Calefato, and Maelick Claes. 2018. Natural language or not (NLON). *Proceedings of the 15th International Conference on Mining Software Repositories*, 387–391.

[31] Nicole Novielli, Fabio Calefato, Davide Dongiovanni, Daniela Girardi, and Filippo Lanubile. 2020. Can We Use SE-specific Sentiment Analysis Tools in a Cross-Platform Setting? *MSR 2020* (2020), 158–168.

[32] Nicole Novielli, Fabio Calefato, Filippo Lanubile, and Alexander Serebrenik. 2021. Assessment of off-the-shelf SE-specific sentiment analysis tools: An extended replication study. *Empirical Software Engineering* 26 (2021). Issue 4.

[33] Nicole Novielli and Alexander Serebrenik. 2019. Sentiment and Emotion in Software Engineering. *IEEE Software* 36 (9 2019), 6–23. Issue 5.

[34] Marco Ortu, Bram Adams, Giuseppe Destefanis, Parastou Tourani, Michele Marchesi, and Roberto Tonelli. 2015. Are bullies more productive? Empirical study of affectiveness vs. issue fixing time. *IEEE International Working Conference on Mining Software Repositories* 2015-August, Section II (2015), 303–313.

[35] JONATHAN POSNER, JAMES A. RUSSELL, and BRADLEY S. PETERSON. 2005. The circumplex model of affect: An integrative approach to affective neuroscience, cognitive development, and psychopathology. *Development and Psychopathology* 17 (9 2005). Issue 03.

[36] Aniket Potdar and Emad Shihab. 2014. An Exploratory Study on Self-Admitted Technical Debt. *2014 IEEE International Conference on Software Maintenance and Evolution*, 91–100.

[37] Naveen Raman, Minxuan Cao, Yulia Tsvetkov, Christian Kästner, and Bogdan Vasilescu. 2020. Stress and burnout in open source. *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, 57–60.

[38] Navdeep Singh and Paramvir Singh. 2018. How Do Code Refactoring Activities Impact Software Developers' Sentiments? - An Empirical Investigation into GitHub Commits. *Proceedings - Asia-Pacific Software Engineering Conference, APSEC* 2017-Decem (2018), 648–653. Issue December.

[39] Rodrigo Souza and Bruno Silva. 2017. Sentiment Analysis of Travis CI Builds. *IEEE International Working Conference on Mining Software Repositories* (2017), 459–462.

[40] Margaret-Anne Storey and Alexey Zagalsky. 2016. Disrupting {Developer} {Productivity} {One} {Bot} at a {Time}. In *Proceedings of the 2016 24th {ACM} {SIGSOFT} {International} {Symposium} on {Foundations} of {Software} {Engineering} ({FSE} 2016)*. ACM, New York, NY, USA, 928–931.

[41] Mairieli Wessel, Bruno Mendes de Souza, Igor Steinmacher, Igor S. Wiese, Ivanilton Polato, Ana Paula Chaves, and Marco A. Gerosa. 2018. The Power of Bots: Understanding Bots in OSS Projects. *Proceedings of the ACM on Human-Computer Interaction* 2 (11 2018), 1–19. Issue CSCW.

[42] Mairieli Wessel, Igor Wiese, Igor Steinmacher, and Marco Aurelio Gerosa. 2021. Don't Disturb Me: Challenges of Interacting with Software Bots on Open Source Software Projects. *Proceedings of the ACM on Human-Computer Interaction* 5, CSCW2 (2021). arXiv:2103.13950

[43] Michal R. Wrobel. 2013. Emotions in the software development process. *HSI 2013* (2013), 518–523.

[44] Laerte Xavier, Fabio Ferreira, Rodrigo Brito, and Marco Tulio Valente. 2020. Beyond the Code: Mining self-admitted technical debt in issue tracker systems. *MSR 2020*, 137–146.

[45] Fiorella Zampetti, Gianmarco Fucci, Alexander Serebrenik, and Massimiliano Di Penta. 2021. Self-admitted technical debt practices: a comparison between industry and open-source. *Empirical Software Engineering* 26 (11 2021). Issue 6.

[46] Ting Zhang, Bowen Xu, Ferdian Thung, Stefanus Agus Haryono, David Lo, and Lingxiao Jiang. 2020. Sentiment Analysis for Software Engineering: How Far Can Pre-trained Transformer Models Go? *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 70–80.