Analyzing the Developer's Sentiment in Software Components: A Decade-Long Study of the Apache Project

Tien Rahayu Tulili, Ayushi Rastogi and Andrea Capiluppi
Bernoulli Institute for Mathematics, Computer Science and Artificial Intelligence, Faculty of Science and Engineering, University of Groningen, Groningen, The Netherlands

Keywords: Complexity, Sentiments, Software Components, File Dependencies, Open Source Software, Software Architecture.

Abstract: Open-source software development relies heavily on effective collaboration among developers, with communication often reflecting emotional responses to the technical challenges encountered. The Apache HTTP Server ('httpd') project, a widely used web server, provides a rich dataset to explore how developer sentiment may be influenced by the complexity of software components.
This study aims to investigate the relationship between developer sentiment and software component complexity in the Apache 'httpd' project. Specifically, it seeks to determine whether emotional expressions, captured through sentiment analysis, correlate with the complexity of the components developers work on over a decade of project development (2015–2024).
We utilized two primary datasets: developer communication from the mailing list and commit data. Sentiment analysis was conducted using Sentistrength-SE to classify messages as positive or negative. Software component complexity was measured using static code analysis tools, and a network model of file dependencies was created to examine the architectural structure. Statistical tests, including ANOVA and Tukey HSD, were applied to assess the relationship between sentiment, complexity, and developer contributions.
The results indicate that complexity is not necessarily associated with developers' sentiments. However, the most crucial component was significantly affected by sentiments. Developers contributing to more complex components expressed more negative sentiments, suggesting that complexity may contribute to emotional strain. These findings offer insights into managing developer well-being and improving project management strategies in open-source development environments by addressing both technical and emotional factors.

1 INTRODUCTION

Developers are the driving force behind software development activities. Their contributions (writing code, fixing bugs, participating in code reviews, and submitting commits) are essential for a project's success. How developers manage tasks and collaborate within teams can directly affect their productivity, and ultimately influence the sustainability of the software (Murić et al., 2019; Jalote and Kamma, 2019). In addition to their contributions and collaboration, the way developers interact with software components (such as modules, libraries, tools, and frameworks) plays a critical role in shaping the quality and speed of development.

Software components represent the fundamental building blocks of a project (Lau, 2006). These components (e.g., modules, libraries, frameworks) are crucial for the software's functionality and architecture. Developers must manage them effectively to ensure new features are implemented correctly, dependencies are resolved, and system stability is maintained. However, the interdependent and often complex nature of these components poses challenges. Balancing the maintenance of existing code with the addition of new features can be difficult. Well-planned component systems can accelerate development, whereas poor management can slow progress significantly.

Sentiments are frequently expressed during software development (Robinson et al., 2016; Murgia et al., 2014; Murgia et al., 2018), influencing how developers engage with their work and collaborate with others. Tight deadlines, excessive workloads, and communication challenges often give rise to negative emotions such as frustration, anger, or dissatis-

Tulili, T. R., Rastogi, A. and Capiluppi, A.
Analyzing the Developer's Sentiment in Software Components: A Decade-Long Study of the Apache Project.
DOI: 10.5220/0013162000003912
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 20th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP 2025) - Volume 1: GRAPP, HUCAPP and IVAPP*, pages 483-494
ISBN: 978-989-758-728-3; ISSN: 2184-4321
Proceedings Copyright © 2025 by SCITEPRESS – Science and Technology Publications, Lda.

faction, potentially leading to burnout and a decline in productivity. Conversely, positive emotions, such as satisfaction and happiness, arise when problems are solved successfully, feedback is positive, or the community offers support. These emotional dynamics can affect developer engagement, retention, and collaboration (Sage Sharp, 2015; Philipp Ranzhin, 2015).

This study aims to investigate the relationship between three key aspects of open-source software development: developers, components, and sentiments. Specifically, we focus on understanding how developers' sentiments interact with the complexity of software components in collaborative development environments.

In open-source software projects, developers often contribute to multiple components or packages, and several developers may work on the same component. To coordinate these efforts and resolve technical issues, developers rely on communication channels such as mailing lists and chat platforms. These exchanges often carry emotional undertones, both positive and negative (Murgia et al., 2018; Murgia et al., 2014). At times, negative emotions can escalate, leading to toxic interactions that damage team cohesion (Sage Sharp, 2015; Philipp Ranzhin, 2015). Research shows that emotions can significantly impact aspects of development such as productivity and project retention. For example, developers experiencing dissatisfaction may disengage from their work or leave the project entirely (Graziotin et al., 2018; Garcia et al., 2013). Previous studies have also linked negative sentiments in commit messages to bug-related work (Huq et al., 2020), while others have observed that agile teams frequently express emotions in response to changing requirements (Madampe et al., 2020).

Despite growing interest in the role of emotions in software development, no studies have specifically focused on how developer sentiment affects software components. Additionally, the architectural implications of developer sentiment have yet to be explored. Our study addresses this gap by investigating the Apache 'httpd' project [1]. We aim to determine whether developer sentiment has a measurable impact on software components and how component complexity influences sentiments during the development process. To guide our investigation, we pose the following research questions:

RQ1. Are there differences among developers in terms of their sentiment involvement during software development?

Rationale: This question aims to explore whether distinct groups of developers, based on their sentiment expressions during communication, emerge in the software development process. We also examine their commit activities, contributions, and communication patterns.

RQ2. To what extent does the complexity of software components impact developers' sentiments?

Rationale: This question investigates the relationship between the complexity of software components and developers' sentiments. We aim to understand whether complex components elicit more negative sentiments.

This paper is structured as follows: Section 2 reviews related literature. Section 3 details the methodology and Section 4 presents results and discusses their implications. Section 5 addresses threats to validity while Section 6 concludes with key insights and future work.

2 RELATED WORK

We structured our review of related work along three key axes: "sentiments and developers", "developers and software components", and "sentiments and software components". Previous studies have predominantly focused on the first two axes: "sentiments and developers" and "developers and components." However, the "sentiments and software components" topic, particularly in Free/Libre Open Source Software (FLOSS), remains unexplored. Specifically, no research has examined how software components influence developers' sentiments and how this, in turn, impacts project dynamics. Below, we summarize prior research on the first two axes.

2.1 Sentiments and Developers

Research on developer sentiment during software development has gained significant attention. For instance, Tourani et al. (Tourani et al., 2014) analyzed developer mailing lists to identify expressions of happiness and distress during development, using sentiment analysis techniques. Similarly, Fucci et al. (Fucci et al., 2021) explored developers' habits around self-admitted technical debt by classifying issue comments, finding that functional issues tended to evoke more negative sentiments. In a related study, Pletea et al. (Pletea et al., 2014) focused on security discussions on GitHub, analyzing commit and pull request comments. They confirmed that security-related discussions contained more negative emotions compared to non-security-related ones. However, these studies focused their emotional investigation on topics-based discussion only in whole projects during

[1] https://httpd.apache.org/

the development. They did not investigate the inner level of a project, such as at the components or package level. Therefore, our study addresses this gap by investigating the sentiments of developers at the granular level.

Ortu et al. (Ortu et al., 2016) also studied developer interactions, investigating how psychological conditions influence the tone of their communication. Using the Sentistrength tool, they found that developers often responded to negative comments with either positive or negative remarks. In terms of productivity, some studies have examined the correlation between developer sentiment and activity levels, including how sentiments relate to resolved/unresolved issues (Valdez et al., 2020), peak productivity days (Valdez et al., 2020; Guzman et al., 2014), and the time required to fix issues (Ortu et al., 2015). Nevertheless, these sentiment studies focused more on the developer's aspect relating to their emotional condition and activity during the project development as a whole. They did not take into account the granular level of the project that our study will focus on.

Several other studies have explored the relationship between developer sentiment and software development activities. Huq et al. (Huq et al., 2020) investigated the connection between developer sentiment and software bugs by analyzing GitHub commits. Their findings revealed that commits associated with bug-related activities, such as introducing, fixing, or preceding bugs, were generally more negative. Similarly, Robinson et al. (Robinson et al., 2016) identified a statistical correlation between changes in developer routines and shifts in sentiment, suggesting that behavioural changes are often accompanied by emotional changes. However, similar to the aforementioned studies earlier, there is still a lack of consideration for studying more at the granular level, such as at the packages level, which this study intends to investigate.

2.2 Developers and Components

Wu et al. (Wu et al., 2023) examined the role of social and technical dependency networks in open-source software (OSS) communities. They analyzed network metrics, such as degree centrality, betweenness centrality, and closeness centrality, to understand their impact on project success. The study found that nonlinear relationships exist between the number of connections within social and technical networks and OSS success—suggesting that an increase in connections does not always correlate with project success.

However, Wu et al.'s study did not consider the sentiment dynamics of developer interactions within these technical networks. Our research fills this gap by investigating how sentiments evolve at the component level. We explore how sentiment responses during communication affect individual software components, thus linking the technical and sentiment dimensions of OSS development.

Software modularity is a crucial aspect of component-based software engineering, enabling system evolution and maintenance (Weide and Gibson, 1997; MacCormack et al., 2007). Modular designs facilitate future adaptations and create 'option value' for improved designs (MacCormack et al., 2007). Research suggests that higher modularity in software development can lead to increased productivity and quality. Studies have shown that modular reuse enhances productivity, quality, and reduces costs in embedded software development (Sun et al., 2014; Sun et al., 2016). Higher modularity is associated with improved development productivity and fewer software failures (Cataldo and Herbsleb, 2013). It also reduces development time and coordination efforts (Gomes and Joglekar, 2008). Developers tend to invest less cognitive effort in understanding modular code, although they may spend more time on it (Segalotto et al., 2023). Modularity positively interacts with developers' temporal work styles to influence software quality and job satisfaction (Foerderer et al., 2016). However, some developers do not fully utilize advanced modularity techniques, often sticking to basic object-oriented programming (Fukuda and Leger, 2015). Additionally, while higher modularity improves developer retention and reduces bug-fixing time, increased complexity has the opposite effect (V. and C. Palvia, 2007). Nevertheless, even though these studies focus on the modularity aspect of software development, they disregard the aspect of the developer, such as sentiment that may also play an important role in improving developers' productivity and software quality. Hence, by taking into account the sentiments, our study explores the sentiments at the level of component or packages.

Moreover, Mockus et al. (Mockus et al., 2000), in their case study report, scrutinized the development process of an open-source project, the Apache web server, to acknowledge the possibility of combining the development process of open-source software and commercial. They initially quantified several aspects such as developer participation, core team size, code ownership, productivity, defect density, and problem resolution interval by deeply analyzing the email archives of source code change history and problem reports of the project. Their study concluded with some proposals of hypotheses. One of the hypotheses relates to the suggested number (e.g. 10-15 people) of core developers who control the code base or would create approximately 80% or more of the new functionality of the project. Additionally, in a separate study (Mockus et al., 2002), they tested and refined their proposed hypotheses with another OSS application, the Mozilla browser. Still using similar types of archives in their previous study and methodology, they revisited the hypotheses and made some refinements. The refined hypothesis includes the size of core developers that would not be higher than the aforementioned size if the core group used only informal ad hoc means of coordinating their work.

However, Mockus et al.'s studies only investigated the aspect of developers's participation during project development without looking at the developer's sentiment. Hence, we fill the gap by adding the developer's sentiment and considering the components and developers' activity.

3 METHODOLOGY

3.1 Definitions

The key terms used in this analysis are defined below:

- **Active Developer:** refers to any developer who pushed commits to the Apache repository between 2015 and 2024.
- **Negative Message:** refers to a message containing sentences with negative scores between -3 and -5 and positive scores between +1 and +2. Strongly negative words such as "*really hate*," "*awful*," and "*suffer*" characterize this range. Messages containing sentences with equal positive and negative scores and do not contain any range of the aforementioned scores are ignored (e.g., a message scoring -3 and +3) .
- **Positive Message:** refers to a message containing sentences with positive scores between +3 and +5 and negative scores between -1 and -2. Strongly positive phrases like "*very cool*," "*excellent*," and "*thanks*" define this range. Messages containing sentences with similar balanced scores and that do not contain any range of the aforementioned score are excluded.
- **Sentence:** In this paper we divided long messages in several sentences. A 'sentence' refers to a group of words starting with a capital letter and ending with punctuation (e.g., periods, question marks, and exclamation points).
- **Developer Writing Negative Sentences (DWNs):** refers to developers who predominantly write negative sentences. DWNs are identified as those in the top 5% of developers by the number of negative sentences, based on a quartile analysis.
- **Developer Writing Positive Sentences (DWPs):** refers to developers who primarily write positive sentences, with a similar identification criteria for DWNs, but focusing on positive sentences.
- **Component:** refers to a community of files linked by dependencies. Further details on constructing the component network are in section 3.4.

3.2 Data Sets

In this study, we investigated the Apache community, an open-source software community that has existed since the 1990s, focusing on the project 'httpd', which powers one of the world's most widely used servers. The project employs various communication channels to coordinate its development efforts. This component-based development project is built upon over 60 standard Apache modules comprising hundreds of files. This community fits with the requirements of our study. These requirements include implementing component-based architecture and employing text-based communication for discussing technical issues, and they existed for over ten years. Our study spanned a ten-year interval, from January 2015 to May 2024, encompassing multiple programming languages, including SQL, Java, Python, and R.[2]

For the empirical analysis we utilized three primary data sources: i) the developer mailing list, which captures discussions about technical issues related to the software; ii) the GitHub commits, which we used to quantify the complexity of components; and iii) the source code to extract the Apache components. All the datasets are publicly accessible.

1. **Mailing List for Sentiments.** We collected data from the mailing list dedicated to discussing source code changes and technical issues related to the HTTP server[3]. The publicly available archive of this mailing list[4] contains 47,104 emails.
2. **Commits by Developers.** We collected and analyzed 162,289 commits from the project's GitHub repository[5]. From the commits metadata, developers were classified into two groups: 1) Develop-

[2]We describe all the steps of our methodology in this link: https://shorturl.at/RAT7m

[3]https://httpd.apache.org/lists.html#http-dev

[4]https://marc.info/?l=apache-httpd-dev&r=1&w=1

[5]https://github.com/apache/httpd

ers Writing Negative Sentences (DWNs), and 2) Developers Writing Positive Sentences (DWPs), as defined in section 3.1.

3. **Source code for Complexity and Dependencies.** We used complexity data provided in the commit dataset and employed a static code analysis tool[6] to retrieve file dependencies. The code files used are 591 files. A component network based on these dependencies was then generated using Infomap (Edler et al., 2024).

3.3 Data Extraction, Preprocessing, and Sentiment Classification

We followed several steps in our analysis, beginning with data extraction and preprocessing, followed by sentiment labelling to address our research inquiries.
Data Extraction – We collected two distinct datasets, scraping the Python source code from the mailing list archive and extracting commit data using Git commands. The mailing list archive was used for the first dataset mentioned in Section 3.2. Meanwhile, the commit data was used for the second and third datasets. The *metadata* for the mailing list includes *'Subject,' 'Sender,' 'Date,'* and *'Message-Body,'* while the commit metadata includes details such as *'hash,'*, *'msg'*, *'committer_email'*, *'committer_name'*, *'committer_date,'*, *'filename,'* and *'complexity.'*. Furthermore, we extracted the source files and the complexity of each source file provided in Github commits. All datasets were stored in a database for subsequent analysis
Data Preprocessing – As we intend to analyze the message content for our sentiment analysis, we need to remove unnecessary lines. Message body content was cleaned by removing lines starting with '>', URLs, names, signatures, and greetings, along with any code syntax or HTML/XML tags. The output was divided into files for DWNs, DWPs, and sentiment-tagged messages. Multiple email addresses for a single developer were standardized to maintain consistency.
Sentiments Classification – In our study, we utilized the sentiment analysis framework proposed by Yadollahi et al. (Yadollahi et al., 2017) to classify opinion polarity as either positive, negative, or neutral. To conduct this analysis, we employed SentiStrength-SE (Islam and Zibran, 2018), a sentiment tool developed for the software engineering domain. In addition, to classify messages from the mailing list to sentiments classes, negative and positive, we employed the tool, designed specifically for Software Engineering and utilised in previous studies (Calefato et al., 2018; Pletea et al., 2014; Chen et al., 2021; El Asri et al., 2019; Girardi et al., 2021). This tool provides positive and negative sentiment scores ranging from +1 to +5 and -1 to -5, respectively.

SentiStrength-SE is a dictionary-based classifier that enhances the original SentiStrength by utilizing a domain-specific dictionary. It analyzes 490,000 commit messages from 50 open-source projects on GitHub, providing bipolar scores for sentiment classification, with scores ranging from +1 to +5 for positive sentiments and -1 to -5 for negative ones. The tool is based on six basic emotions: joy, love, anger, sadness, fear, and surprise.

In line with previous research by Ortu et al. (Ortu et al., 2015), joy and love indicate positive polarity, while anger, sadness, and fear indicate negative polarity. Surprise is categorized into two sets based on context: negative (surprise-) and positive (surprise+). However, our focus remained strictly on the bipolar sentiments of negative and positive.

Email messages typically contain multiple sentences, each with its own sentiment score. Acknowledging the potential impact of highly positive or negative sentences on both the writer and reader (Richter et al., 2010), we adopted a sentence-level analysis approach. We divided all email messages into individual sentences, analysing both positive and negative sentences separately. Sentence-ending punctuation (e.g., periods, question marks, exclamation points) served as indicators to delineate sentence boundaries. We only considered the sentence written by the sender and ignored the quoted sentences referring to the previous email (sender).

Hereby the example of a message[7]: '*So do you mean, aprlib.apache.org? Who's on the PMC for it? What's its charter? Is it really a big enough deal to create a whole new project it to its own project once we figure out answers to all the above questions? I'd really hate to break the nomenclature conventions that we'd all talked about and I thought decided upon, regarding the hierarchy of projects and components underneath them. Or is this an "incubator" project?*' The message as a whole contains 6 sentences: the Sentistrength-SE gave negative scores (-1, -1, -1, -1, -5, -1), and positive scores (1, 1, 1, 1, 1, 1, 1). We classified it as a negative message as it matched the criteria of Negative Message described in Section 3.1.

[6]Understand, http://scitools.com

[7]The message does not contain personal data, so it does not infringe the GDPR. In addition, the mailing list messages are publicly accessible and have been made public by the Apache community.

3.4 Post-Processing of the Data

As post-processing activities, we firstly linked the mailing list and commit datasets by timestamp (year, month, day) and developer name, manually unifying inconsistent names and emails across both datasets. These linked datasets were used in our analysis to answer RQ1 and RQ2.

Secondly, in order to measure complexity, we utilized cyclomatic complexity values obtained from the PyDriller library (Spadini et al., 2018). The process involved three steps:

- First, we calculated yearly mean complexity values for each file to account for annual fluctuations.
- Second, these values were aggregated to compute the yearly mean complexity for each component.
- Finally, we calculated the overall mean complexity for each component over the years. These values were used to address RQ2 (see Section 4.2).

Thirdly, we normalized the differences between positive and negative sentiment counts across all components annually, using a z-score transformation. This normalization was critical for answering RQ2.

Finally, the component network was constructed by analyzing all files in the Apache 'httpd' project as of May 2024 using a static code analysis tool. Dependencies between files were represented as node pairs with weights, and Infomap (Edler et al., 2024) was used to generate the network. Our analysis focused on the 20 largest components out of a total of 35.

3.5 Statistical Approaches

We employed various statistical methods to address our research questions. For RQ1, we used the Mann-Whitney test with Bonferroni correction to evaluate differences. To address RQ2, an analysis of variance (ANOVA) was conducted to assess complexity differences across components. Pairwise differences were further examined using Tukey's Honestly Significant Difference (HSD) method, considering both normalized sentiment scores and mean complexity values (see Section 3.4).

4 RESULTS AND DISCUSSION

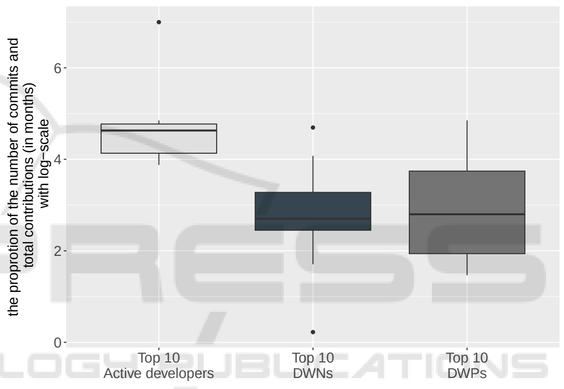
4.1 RQ1: Comparing Apache Developers

We quantified and analyzed the activity of developers, specifically focusing on DWNs, DWPs, and the most active developers, in terms of their commit activities and the frequency of sentimental interactions during their contributions to the 'httpd' project.

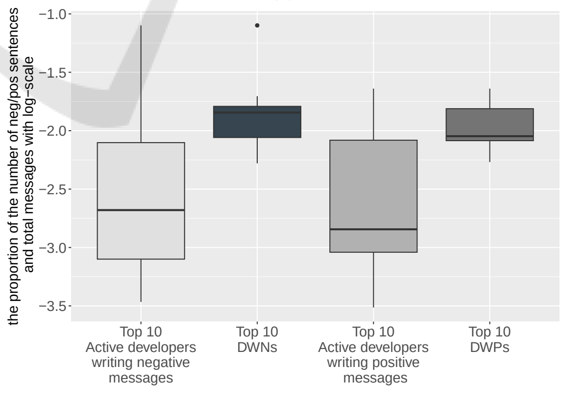
Significant differences were observed among these groups. Figure 1 illustrates these differences through boxplots:

(a) The boxplots depict the number of commits/contributions per month. The very dark grey represents the top ten DWNs, dark grey represents the top ten DWPs, and very light grey represents the top ten most active developers.

(b) Another set of boxplots illustrates the number of positive and negative messages as a fraction of the total messages sent. In this context, very dark grey and dark grey correspond to the top ten DWNs and DWPs, respectively, while light grey and very light grey indicate the top ten active developers who wrote positive and negative messages, respectively.



(a)



(b)

Figure 1: Boxplots of the top 10 of active developers, DWNs, DWPs regarding a relative number of commits done (a) and of positive and negative sentences written (b).

These visualizations highlight the varying levels of engagement and sentimental expressions among different developer groups within the 'httpd' project.

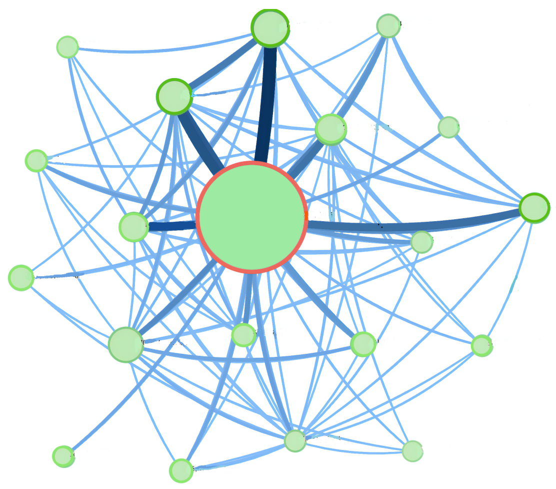


Figure 2: The Infomap dependency network of twenty software components illustrates the structure of the Apache 'httpd' project.

From Figures 1a and 1b, we observed that the most active developers (those who wrote negative messages) tend to use fewer negative sentences when communicating with other developers during software development. To statistically validate this observation, we conducted a Mann-Whitney U Test with Bonferroni correction, comparing the two groups: the most active developers writing negative messages and the top ten DWNs. The resulting p-value was 0.023, leading us to reject the null hypothesis that these groups contribute equally in terms of the proportion of negative messages to total messages.

Additionally, we performed a similar statistical comparison between the top ten active developers writing positive messages and DWPs using the Mann-Whitney U Test with Bonferroni correction. Here, the obtained p-value was 0.006, prompting us to reject the null hypothesis ('these groups contribute equally in terms of the proportion of positive messages to total messages').

Based on these findings, we concluded that despite their high number of commits and extensive contributions, the top ten developers are more likely to communicate neutrally or use fewer negative or positive words. In contrast, the top ten DWNs and DWPs exhibit similarities in their communication patterns, as shown in the figures. Upon further investigation of the developers within these groups, we identified overlaps, indicating that these developers tend to express both negative and positive sentiments when communicating with others. Furthermore, it is human nature that developers as human beings involve their sentiments or emotions in a situation, as this is in line with the previous study that found how negative emotion (e.g. anger) negatively correlates with situational power in the workplace (Fitness, 2000).

Moreover, it was noted that many of these developers worked within the same component, identified as having high complexity and a significant negative impact. For detailed insights, refer to Section 4.2.

While prior studies (Pletea et al., 2014; Huq et al., 2020; Fucci et al., 2021; Valdez et al., 2020) suggest a general prevalence of negative emotions in specific contexts (e.g., security, bug-related activities, resolved/unresolve issues, or functional issues), our findings indicate that the most active developers—despite their extensive contributions—communicate in a more neutral tone, using fewer emotionally charged words overall. This contrasts with DWNs and DWPs, who consistently use both positive and negative sentiments. Furthermore, our findings connect these sentiment patterns to component-level contexts, uncovering that these expressive developers often work in high-complexity areas with notable negative impacts. These insights not only fill the gap in the granular-level analysis highlighted in prior studies but also introduce the novel observation of how communication styles relate to developer roles and the technical complexity of their work.

4.2 RQ2: Sentiment-Driven Components

Figure 2 presents the network diagram of Apache 'httpd', consisting of twenty components, using the dependencies between source files (C and Python) and summarized by Infomap. A variable diameter circle represents each component, while variable width lines depict the undirected flow of the connections. The larger the circle, the higher the dependencies, while the thicker the links, the stronger the relations between the components. Detailed information about each component is publicly available[8].

Upon observation, we noted that one component, highlighted with a dark orange line, stands out as the largest and the most crucial, significantly larger than the others. Additionally, this component shows significant high dependencies, showing that inside this component contains many sub-components that are more dependent on each other than other outside smaller components. Utilizing this component network, we further analyzed the complexity of these twenty components, as depicted in Figure 3.

Figure 3 displays these components, showcasing sentiments and complexity. Each box's color indicates the normalized differences between positive

[8]The list of components, identified by the IDs 1:1 to 1:20, is available at https://shorturl.at/IcQRw

and negative messages, while the number inside each box represents the mean complexity values across all years for that component. Vertical and horizontal lines denote periods with no commits.

It's evident from the heatmap that '1:1', the largest and most crucial component, consistently experiences high negative sentiments across all years. '1:2' and '1:4' show spikes in negative sentiments in 2022, while '1:3' exhibits a peak in positive emotions in 2017. The remaining components appear to be less influenced by either positive or negative sentiments.

Furthermore, we conducted statistical comparisons of complexity across these components and found a highly significant p-value ($p < 2e-16$), indicating significant differences among the components.

Subsequently, we performed Tukey HSD post-hoc tests to identify specific differences between components, particularly focusing on comparisons involving '1:1'. Our analysis revealed that components '1:15', '1:18', '1:17', '1:16', '1:10', '1:6', and '1:7', while less impacted by emotional sentiments, differed significantly from '1:1' in terms of complexity. Notably, component '1:15', which ranked highest in complexity, did not significantly differ from '1:6' and '1:16', but showed significant differences with '1:3', '1:18', '1:17', and '1:10'.

Based on these findings, we concluded that while sentiments play a notable role in some software components, only '1:1' is distinctly impacted by emotional expressions. Complexity, on the other hand, appears to vary significantly across different components and does not necessarily correlate with the emotional sentiments expressed by developers. Additionally, it's noteworthy that many developers associated with the top ten DWNs and DWPs worked within '1:1', suggesting their influence on this component's dynamics.

Our results highlights a significant shift in focus from macro-level analyses of sentiment in software development to a more granular, component-level perspective. Prior studies (Sun et al., 2014; Sun et al., 2016; Wu et al., 2023; Mockus et al., 2000; Cataldo and Herbsleb, 2013; Foerderer et al., 2016; V. and C. Palvia, 2007) explored developers' interactions or modularity without addressing sentiment. In contrast, our study uniquely bridges these gaps by investigating how sentiments manifest at the component level, uncovering patterns that align with technical complexity and dependencies. For example, while prior research (Cataldo and Herbsleb, 2013; Sun et al., 2014; Sun et al., 2016; Foerderer et al., 2016) associated productivity and developer satisfaction with modularity, our findings reveal that components with high dependencies, such as '1:1', consistently exhibit higher negative sentiment, linking sentiments to technical complexity and dependencies in ways previously unexplored.

Furthermore, our results provide a nuanced understanding of developer communication. Unlike previous studies, which often focused on general sentiment trends among developers or their impact on productivity (Valdez et al., 2020; Guzman et al., 2014; Ortu et al., 2015), our analysis uncovers specific dynamics between sentimentally expressive developers (DWNs/DWPs) and their activity within critical components. The observation that the most active developers communicate in a more neutral tone, while DWNs and DWPs are often associated with highly dependent and complex components, offers new insights into how sentiments might influence—and might be influenced by—the technical characteristics of software components. Moreover, our statistical findings underscore that while complexity significantly differs among components, only '1:1' demonstrates a strong correlation between high complexity and negative sentiment. This adds a new dimension to the study of modularity by integrating sentiment as a factor, thereby extending prior research on modularity's impact on productivity and software quality into the emotional domain.

4.3 Implications for Practitioners and Researchers

Observation on Developer Behaviour. The observation that the most active developers tend to use fewer negative messages than the top negative senders highlights potential differences in communication styles and their impact on project dynamics. Practitioners can analyze these patterns to better understand how individual developer behavior influences the overall tone and effectiveness of communication within the community. For instance, profiling the behavior of individual developers based on their communication patterns, activity levels, and contributions to the project may assist leading developers in making informed talent management, team formation, and resource allocation decisions within the community.

Additionally, our findings indicate that top developers, who often express strong emotions, play a pivotal role in shaping the software's architecture. Components with higher complexity attract more negative sentiments, underscoring the challenging relationship between emotional involvement and technical intricacies. This highlights the necessity for project managers to monitor both technical metrics and the emotional well-being of developers.

Insights from the Communication Within a Compo-

nent. Our earlier findings highlight that some software components may be influenced by sentiments expressed by developers during software development, particularly large components with high dependencies, which are simultaneously handled by more developers. Practitioners can leverage this information regarding specific components within the project where negative communication is prevalent and implement targeted interventions to address underlying issues. For example, regular code reviews could be implemented specifically focused on critical components prone to issues and bugs.

Qualitative Research. While our quantitative analysis provides valuable insights, researchers can complement these findings with qualitative methods such as interviews and surveys to gain more detailed information about the specific components managed along with the sentiments expressed during their handling. This can further enrich the understanding of the underlying motivations, perceptions, and experiences of developers within the open-source software community, providing deeper insights into sentiments and software component dynamics.

Comparative Studies and Generalisability. Our methodology may be expanded to other open-source projects or software development communities. Researchers can conduct comparative studies analyzing communication patterns, developer behaviors, and project outcomes across diverse contexts. This can help identify common trends and context-specific factors influencing dynamics.

5 THREATS TO VALIDITY

Our study's results are subject to several limitations that impact their validity. Therefore, we described several threats to validity in this study.

Construct Validity – One significant concern is the potential lack of correlation between the two datasets we linked: the mailing list archive and the commit datasets, which were linked based on time and developer names. To assess this, we manually checked 100 random samples from both datasets to ensure their linkage and correlation. This involved conducting stratified random sampling across different years to ensure representative sampling. In addition, we did not conduct further investigation to distinguish between negative comments containing specific topics, such as the development process or the code in the repository.

Conclusion Validity – Our methods, such as picking up the top ten DWNs, DWPs, and Active developers, may bias the results in finding the differences among these groups of developers. However, as these are only our preliminary results, we will extend the groups by considering all the data points in our next future work.

External Validity – Another limitation arises from our reliance on the Sentistrength-SE tool for sentiment labelling of messages. This tool may generalize the meaning of sentences without thoroughly examining their context or the entirety of the message, potentially leading to inaccuracies in sentiment classification. Furthermore, the top ten subpopulations may not generalize to the broader population.

Addressing these limitations is crucial for ensuring the reliability and accuracy of our findings, particularly in understanding the nuanced relationships between developer sentiments, commit activities, and project dynamics within the 'httpd' project.

6 CONCLUSION AND FUTURE WORK

This study highlights the relationship between developer sentiment and software component complexity within the Apache 'httpd' project over nearly a decade. Our results indicate that components with higher dependencies tend to generate more negative sentiments, suggesting that managing technical issues, such as file dependencies and complexity, is not just a matter of code but also of maintaining developer well-being.

The findings suggest that technical complexity and sentiment are not necessarily interconnected; however, the complexity across the components varies. In addition, the most crucial component contained more negative than the rest. By identifying components that are prone to sentiment responses, project managers can implement strategies to mitigate negative sentiments, such as more frequent code reviews or targeted support for developers working on complex tasks.

Future work should explore these dynamics in other open-source projects to validate the generality of these results and to investigate how sentiment responses can be better managed in the context of large distributed teams. Additionally, incorporating qualitative approaches, such as developer interviews, could provide further insights into how sentiment strain influences software development processes. Furthermore, further investigation into the cause relationship between developers and the software and vice versa should be explored, particularly on the granular level, such as components.

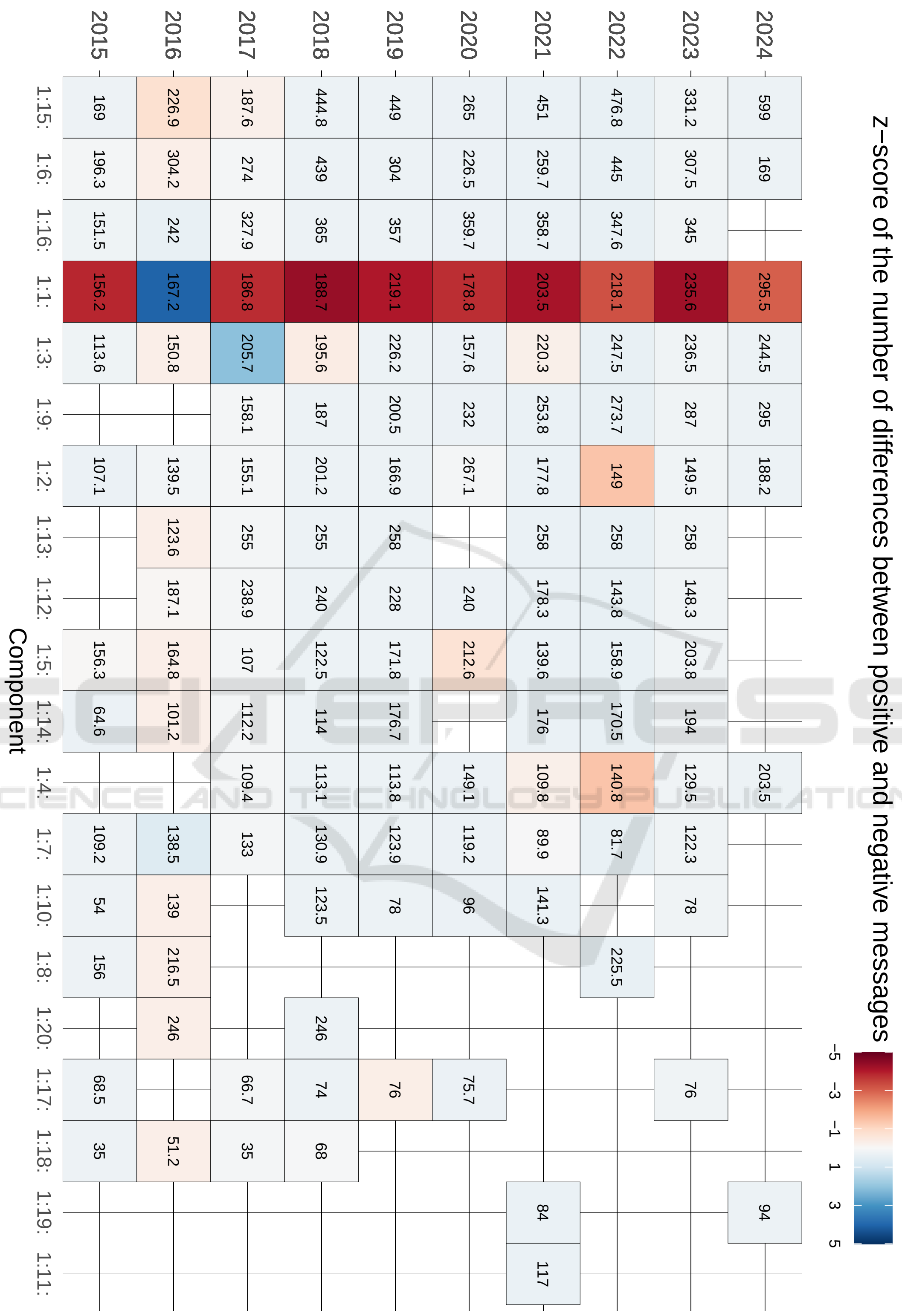


Figure 3: Heatmaps of the number of differences between positive and negative messages and the number of complexity in twenty components yearly.

REFERENCES

Calefato, F., Lanubile, F., Maiorano, F., and Novielli, N. (2018). Sentiment polarity detection for software development. In *Proceedings of the 40th International Conference on Software Engineering*, pages 128–128.

Cataldo, M. and Herbsleb, J. D. (2013). Coordination breakdowns and their impact on development productivity and software failures. *IEEE Transactions on Software Engineering*, 39(3):343–360.

Chen, Z., Cao, Y., Yao, H., Lu, X., Peng, X., Mei, H., and Liu, X. (2021). Emoji-powered sentiment and emotion detection from software developers' communication data. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 30(2):1–48.

Edler, D., Holmgren, A., and Rosvall, M. (2024). The MapEquation software package. https://mapequation.org.

El Asri, I., Kerzazi, N., Uddin, G., Khomh, F., and Idrissi, M. J. (2019). An empirical study of sentiments in code reviews. *Information and Software Technology*, 114:37–54.

Fitness, J. (2000). Anger in the workplace: an emotion script approach to anger episodes between workers and their superiors, co-workers and subordinates. *Journal of Organizational Behavior: The International Journal of Industrial, Occupational and Organizational Psychology and Behavior*, 21(2):147–162.

Foerderer, J., Kude, T., Mithas, S., and Heinzl, A. (2016). How temporal work styles and product modularity influence software quality and job satisfaction. In *Proceedings of the 2016 ACM SIGMIS Conference on Computers and People Research*, volume 44, pages 105–112. ACM.

Fucci, G., Cassee, N., Zampetti, F., Novielli, N., Serebrenik, A., and Di Penta, M. (2021). Waiting around or job half-done? sentiment in self-admitted technical debt. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 403–414. IEEE.

Fukuda, H. and Leger, P. (2015). Why do developers not take advantage of the progress in modularity? In *Proceedings of the 8th International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ACM.

Garcia, D., Zanetti, M. S., and Schweitzer, F. (2013). The role of emotions in contributors activity: A case study on the gentoo community. In *2013 International conference on cloud and green computing*, pages 410–417. IEEE.

Girardi, D., Lanubile, F., Novielli, N., and Serebrenik, A. (2021). Emotions and perceived productivity of software developers at the workplace. *IEEE Transactions on Software Engineering*, 48(9):3326–3341.

Gomes, P. J. and Joglekar, N. R. (2008). Linking modularity with problem solving and coordination efforts. *Managerial and Decision Economics*, 29(5):443–457.

Graziotin, D., Fagerholm, F., Wang, X., and Abrahamsson, P. (2018). What happens when software developers are (un)happy. *Jnl of Systems and Software*, 140:32–47.

Guzman, E., Azócar, D., and Li, Y. (2014). Sentiment analysis of commit comments in github: an empirical study. In *Proceedings of the 11th working conference on mining software repositories*, pages 352–355.

Huq, S. F., Sadiq, A. Z., and Sakib, K. (2020). Is developer sentiment related to software bugs: An exploratory study on github commits. In *2020 IEEE 27th Intl Conf on Software Analysis, Evolution and Reengineering (SANER)*, pages 527–531.

Islam, M. R. and Zibran, M. F. (2018). Sentistrength-se: Exploiting domain specificity for improved sentiment analysis in software engineering text. *Journal of Systems and Software*, 145:125–146.

Jalote, P. and Kamma, D. (2019). Studying task processes for improving programmer productivity. *IEEE Transactions on Software Engineering*, 47(4):801–817.

Lau, K.-K. (2006). Software component models. In *Proceedings of the 28th international conference on Software engineering*, pages 1081–1082.

MacCormack, A., Rusnak, J., and Baldwin, C. Y. (2007). The impact of component modularity on design evolution: Evidence from the software industry. *SSRN Electronic Journal*.

Madampe, K., Hoda, R., and Singh, P. (2020). Towards understanding emotional response to requirements changes in agile teams. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: New Ideas and Emerging Results*, pages 37–40.

Mockus, A., Fielding, R. T., and Herbsleb, J. (2000). A case study of open source software development: the apache server. In *Proceedings of the 22nd international conference on Software engineering*, pages 263–272.

Mockus, A., Fielding, R. T., and Herbsleb, J. D. (2002). Two case studies of open source software development: Apache and mozilla. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(3):309–346.

Murgia, A., Ortu, M., Tourani, P., Adams, B., and Demeyer, S. (2018). An exploratory qualitative and quantitative analysis of emotions in issue report comments of open source systems. *Empirical Software Engineering*, 23:521–564.

Murgia, A., Tourani, P., Adams, B., and Ortu, M. (2014). Do developers feel emotions? an exploratory analysis of emotions in software artifacts. In *Proceedings of the 11th working conference on mining software repositories*, pages 262–271.

Murić, G., Abeliuk, A., Lerman, K., and Ferrara, E. (2019). Collaboration drives individual productivity. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–24.

Ortu, M., Adams, B., Destefanis, G., Tourani, P., Marchesi, M., and Tonelli, R. (2015). Are bullies more productive? empirical study of affectiveness vs. issue fixing time. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 303–313. IEEE.

Ortu, M., Destefanis, G., Counsell, S., Swift, S., Tonelli, R., and Marchesi, M. (2016). Arsonists or firefighters? affectiveness in agile software development. In *Agile Processes, in Software Engineering, and Extreme Programming: 17th International Conference, XP 2016, Edinburgh, UK, May 24-27, 2016, Proceedings 17*, pages 144–155. Springer International Publishing.

Philipp Ranzhin (2015). I ruin developer's lives with my code reviews and i'm sorry. https://habr.com/en/articles/440736/, Last accessed on 09/2019.

Pletea, D., Vasilescu, B., and Serebrenik, A. (2014). Security and emotion: sentiment analysis of security discussions on github. In *Proceedings of the 11th working conference on mining software repositories*, pages 348–351.

Richter, M., Eck, J., Straube, T., Miltner, W. H., and Weiss, T. (2010). Do words hurt? brain activation during the processing of pain-related words. *Pain*, 148(2):198–205.

Robinson, W. N., Deng, T., and Qi, Z. (2016). Developer behavior and sentiment from data mining open source repositories. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 3729–3738. IEEE.

Sage Sharp (2015). Closing a door. https://sage.thesharps.us/2015/10/05/closing-a-door/, Last accessed on 09/2023.

Segalotto, M., Bolzan, W., and Farias, K. (2023). Effects of modularization on developers' cognitive effort in code comprehension tasks: A controlled experiment. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering*, volume 63, pages 206–215. ACM.

Spadini, D., Aniche, M., and Bacchelli, A. (2018). *PyDriller: Python Framework for Mining Software Repositories*.

Sun, H., Ha, W., Teh, P.-L., and Huang, J. (2016). A case study on implementing modularity in software development. *Journal of Computer Information Systems*, 57(2):130–138.

Sun, H., Ha, W., Xie, M., and Huang, J. (2014). Modularity's impact on the quality and productivity of embedded software development: a case study in a hong kong company. *Total Quality Management & Business Excellence*, 26(11-12):1188–1201.

Tourani, P., Jiang, Y., and Adams, B. (2014). Monitoring sentiment in open source mailing lists: exploratory study on the apache ecosystem. In *Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, CASCON '14, page 34–44. ACM.

V., M. and C. Palvia, P. (2007). Retention and quality in open source software projects. *Americas Conference on Information Systems*.

Valdez, A., Oktaba, H., Gómez, H., and Vizcaíno, A. (2020). Sentiment analysis in jira software repositories. In *2020 8th International Conference in Software Engineering Research and Innovation (CONISOFT)*, pages 254–259. IEEE.

Weide, B. and Gibson, D. S. (1997). Behavioral relationships between software components.

Wu, J., Huang, X., and Wang, B. (2023). Social-technical network effects in open source software communities: understanding the impacts of dependency networks on project success. *Information Technology & People*, 36(2):895–915.

Yadollahi, A., Shahraki, A. G., and Zaiane, O. R. (2017). Current state of text sentiment analysis from opinion to emotion mining. *ACM Computing Surveys (CSUR)*, 50(2):1–33.