Tampere University

Niladri Saha

MINING OF MOTIVATORS AND DE-MOTIVATORS FOR SOFTWARE DEVELOPERS FROM GITHUB

AN EMPIRICAL STUDY

Faculty of Information
Technology and
Communication Sciences
M.Sc. Thesis
March 2022

ABSTRACT

Niladri Saha: Mining of Motivators and De-Motivators for Software Developers from GitHub
M.Sc. Thesis
Tampere University
Master's Degree Programme in Software, Web, and Cloud
March 2022
Examiners: Outi-Sievi Korte and Terhi Kilamo

In the present day, the software industry is experiencing a high rise of burnout cases amongst software engineers. Several companies are implementing various policies to motivate software engineers and prevent the rise of burnout cases and depression in the software industry. But still, the cases are rising worldwide. As software engineers are an essential part of the software industry, it is a concern for the software industry and its growth. Thus, it is necessary to find an approach to deal with this.

This thesis aims to find out measurable motivators and de-motivators that can be extracted from GitHub Repositories. It will also be useful for future research on the well-being of software developers in the context of open-source software development as well as for an organization.

An empirical study is done to find out the motivational and de-motivational factors from the extracted data of GitHub. First, a literature review is done to get the list of motivators and de-motivators. Then a quantitative approach is applied by extracting over 26566 Issues with comments from 35 repositories and over 6520 commits from 35 repositories. From the extracted data, mapping is done with the motivators and de-motivators. Once the mapping is completed, the calculation of the motivators and de-motivators is done using metrics that are based on the extracted data. After this, variations of those factors are observed with different sentiments and emotions. Then a survey is done with the developers associated with different GitHub repositories. Once the survey is completed, both the results from the data mining and survey are compared to verify whether these factors from GitHub correctly match with the developers' viewpoint.

Results show that Issue complexity, risk, and collaboration are the factors that can be measured from the GitHub repository. However, it is found that risk is not at all a de-motivator as both the survey and mining results show developers are more willing to work on risky issues. The collaboration score seems to be contradictory with survey results as the survey shows developers feel the positive effect on collaboration.
Keywords: Emotion, Motivation, Issue, Commit, Complexity, Risk, Sentiment, Collaboration

PREFACE

This thesis was done as part of research work under the supervision of assistant professor Outi-Sievi Korte and was funded by Ulla Tuominen Foundation, Finland and Tampere University, Finland. The idea of the thesis was stemmed from the rising number of burn-out cases arising in Finland amongst software engineers in the past few years. This thesis work can be considered as the starting point for future research works related to the well-being of software developers. The topic was unique and quite challenging in the initial days due to the lack of studies made on this domain. However, it was a great experience doing this thesis work due to the voluminous learning associated with it.

I would like to thank my supervisor Outi-Sievi Korte and examiner Terhi Kilamo for their useful suggestions to improve my thesis work. Apart from that, I would like to thank God for allowing me to study master's from Tampere University. Also, I am very grateful to my mother for her unconditional support and love during my stressful and lonely time in Finland.

Tampere, March 2020

Niladri Saha

Table of Contents

LIST OF SYMBOLS AND ABBREVIATIONS

AET	Affective Events Theory
MPS	Motivation Potential Score
AEM	Affective Events Model
AE	Affective Events
JCM	Job Characteristic Model
OSS	Open-Source Software
MOCC	Motivators, Outcomes, Characteristics, and Context
CI/CD	Continuous Integration/Continuous Deployment
BERT	Bidirectional Encoder Representations from Transformers
GloVe	Global Vectors for Word Representations
ELMo	Embeddings from Language Mode
API	Application Program Interface
CPU	Central Processing Unit
GPU	Graphics Processing Unit
VIST	Valence, Instrumentality, Self-Efficacy, Trust
3.x	Python version starting from 3.0

1. Introduction

The software industry is now one of the biggest industries. A survey by Statista [17] has predicted that in 2021 the revenue of the software industry will be close to USD 578,018 million. Worldwide the software industry employs around 55 million workforces [18]. Thus, it is one of the critical service sectors where millions of software engineers work hard to maintain the software's usability for the citizens worldwide. However, a software engineer's job is not the easiest one to do. Coding, testing, designing, or client meetings are part of every software developer's daily routine. In addition to this, in some cases, the constant pressure to prove their mettle, toxic work culture, bad managers, and long working hours lead to a burnout situation for software engineers. To avoid burnout and depression, software engineers need to be motivated towards their work. In a study on software development, Rasch et al. [10] claimed that a software developer's performance has a direct impact on any software development. On the other hand, motivation influences performance [70]. A software developer is also an employee of an organization or part of a software project. Hence, motivation is one of the important factors in performance from a software development perspective. Further, it is claimed that motivation is the desire to act to achieve a goal [19]. It is one of the most important aspects of defining and achieving our goals. As a result, a software developer's day-to-day tasks may be hampered by a lack of motivation. Even a lack of motivation makes it difficult for software engineers to achieve their objectives.

Therefore, in the present day, several companies try to introduce various methods for the well-being of the employees to motivate them to do their job with more flexibility. In addition to that, several policies such as work from home, paid maternity leaves, mental-health programs, performance-based salary hikes, working hour flexibilities are implemented in companies like American Express, Hyatt, Google to motivate employees in their workplace [21]. However, there are still some gaps in implementing such policies, so, many of the software engineers are suffering from high burn-out rates also some companies like Infosys, Accenture, and Wipro are suffering from high attrition rates [20]. From the employee reviews [68][69], it is observed that a de-motivating environment is the main reason behind such attrition in most cases.

Therefore, it is important to get an insight into the motivational and de-motivational factors for software engineers and try to build a framework for an organization or a team to have motivated developers. Several studies have been made in the last ten years on motivational aspects in software engineering. Among these methods, some proposed several ways to mine motivation. Even there are research approaches to mine motivation from the software repositories [6] or using software-based methods to get the factors of motivation [13]. However, they do not provide any guidance on how to derive the measurable motivators and de-motivators or any source that helps us measure the motivators and de-motivators for the software engineers. Thus, this thesis is done to address the gaps in the previous research works. The work is performed as an empirical study to get insight into the motivational factors directly from the mined data.

In this thesis, all the data are collected related to closed issues and their commits from GitHub. Then mapping is done with the motivators as well as de-motivators with the data collected from GitHub. In addition to this, this thesis also shows how to extract sentiment and emotion from all the issues and try to gain more insight to check how the emotion affects these motivators or de-motivators in an Open-Source Software Project context. The main aim of this thesis is to answer the following questions:

RQ1. What are the measurable motivators and de-motivators that can be derived from GitHub?
RQ2. How to derive the measurable motivators and de-motivators from GitHub?
RQ3. How are these measurable motivators or de-motivators influenced by emotion and sentiment?

The whole thesis is structured as follows: Chapter 2 gives the overview of different types of motivation models related to software engineering and related works that had been done in the past on motivation mining. Chapter 3 elaborates on Sentiment and Emotion and describes different emotion and sentiment analysis tools available in the market. Chapter 4 illustrates the Open-Source Project concepts and gives an overview of GitHub. Chapter 5 describes the methodologies used in this thesis. Chapter 6 illustrates the results and their related analysis. It also answers the all-research questions. Finally, Chapter 7 is the conclusion which summarizes the whole thesis and tries to give an idea about the future research related to motivators and de-motivators calculation.

2.Motivation and Motivation Models

This chapter elaborates on the concept of motivation. It also deals with the different types of motivation models that have applications in software engineering. All the models are described with their advantages and disadvantages because it helps in selecting the best model to build the framework for calculating motivators and de-motivators.

2.1 Definition of Motivation

There are several definitions provided for motivation in different industry perspectives. One of the definitions by McConnell [22] states that motivation is a soft factor. It is difficult to quantify, and it is also sometimes overshadowed by other trivial factors. However, it is easier to measure [22]. According to McConnell [22], every organization recognizes the importance of motivation, but no one tries to put it into practice properly. Many conventional management approaches have been observed to be ineffective, resulting in significant reductions in motivation and morale in exchange for tiny methodology improvements.
Motivation can be viewed as *intrinsic* or *extrinsic [24]*. In a nutshell, intrinsic motivation is a desire to do something interesting, whereas extrinsic motivation is a desire to do something because of some rewards associated with it. The inclusive leadership or management style is conducive to intrinsic motivation whereas an exclusive approach like coercive or authoritative action relies on extrinsic motivation [24]. Motivation, on the other hand, is derived from the word 'motive,' which refers to the needs, wishes, wants, or drives that are linked with persons [25]. It is the process that motivates people to achieve their objectives. The desire for money, success, recognition, job happiness, teamwork, and other psychological elements can all influence people's behaviour at work. It also states that the process of motivation consists of three distinct stages: First a need or driver to achieve, second a stimulus that needs to be aroused and finally when needs are satisfied, the satisfaction or attainment of a goal.

2.2 Intrinsic Model of Motivation

The Intrinsic Model of Motivation is built based on the intrinsic motivational factors of a human being. It is claimed that intrinsic motivation is a motivation that is driven by internal factors [26]. Ryan et al. [76] claim that intrinsic motivation is the doing of an activity for its inher-

ent satisfactions rather than for some separable consequences. In humans, intrinsic motivation is not only a form of motivation, but it is a pervasive and also important one [76]. From birth to the stage of adolescence, humans, in their healthy state, are curious, inquisitive. They are ready to explore unknown things. For these, they do not need any extraneous incentives as stated by Ryan et el. [76].

Intrinsic motivation is an important part of a person's well-being. It is more concerned with self-realization, growth of an individual, a satisfaction to achieve something by own merits. The model based on intrinsic motivation elaborates the framework for self-realization or growth of the person.

There are several limitations associated with intrinsic motivation. Some popular psychology studies state one such thing as the "overjustification effect" [26]. The reason is when someone is already intrinsically motivated to do a task that is associated with extrinsic motivation. As stated, if a person loved chemistry (which is an intrinsic motivation), and then the person enrolled in a class on chemistry at school due to love on the course and was focused on getting an excellent grade (which is an extrinsic factor) - the student's intrinsic motivation would diminish [26]. The intrinsic motivation model in such cases, cannot solve such kind of scenario.

Next elaboration of two Intrinsic Models of Motivations: Goal Setting Model and the Expectancy Model are discussed.

2.2.1 Goal Setting Model

Goal Setting Theory or model is an employee engagement tactic that deals with setting specific goals to improve the productivity of the employees in a workplace [63]. To improve employee performance and employee engagement at the workplace, goal-setting theory can help in several ways. Locke et al. [27] described the Goal Setting Theory i.e., Goal Setting Model based on five moderators. These moderators are given below as per the study by Locket et al. [27].

Goal Commitment

Goal commitment is important for people when the goal is difficult to achieve. It is because achieving a difficult goal requires very high effort from the people. If there is no commitment towards the goal, then attaining the goal may be out of sight. According to Locke et al. [27], two categories of factors facilitate goal commitment: a) factors that make goal commitment

important including the importance of the outcome when the goal is achieved and b) people's self-belief that they can attain the goal.

Goal Importance

Goal importance should be clear to the people. There are many ways to convince people about the goal. One of the approaches is publicly announcing with all its background and importance also with the outcome. Generally, in any software engineering project or any ICT company, the importance of the goal is set by a performance review.

Self-Efficacy

Self-Efficacy is another important moderator towards goal commitment. It is just like goal importance that induces goal commitment. For example, to achieve the goal of a technically sound team, it is important to inspire Software engineers to be empowered with the decision-making processes or niche skills. Inspiring leaders try to inspire or motivate their subordinates with motivational speeches or workshops that motivate employees to take the risk.

Feedback

Feedback is also an important part of the goal attainment process. To check their proper progress Software engineers can ask for a review of their goal plan from their supervisors. An effective goal plan needs multiple rounds of discussion between the supervisor and his or her subordinates. A weak review of the goal plan or ineffective goal plan can kill employee motivation or may hamper employees' progress in a workplace.

Task Complexity

Another moderator in goal attainment is task complexity. To complete a high-level task or a task with higher complexity, a high-level skill set is needed. Thus, in that case, employees who lack such skills need to upskill themselves. This case turns to self-efficacy as by self-efficacy an employee can reskill or upskill themselves in certain areas. It helps them further to solve complex tasks.

Fig. 1 shows the Goal Setting Model which explains how moderators with goal core and mechanisms help in achieving satisfaction which further increases the productivity of the employees.

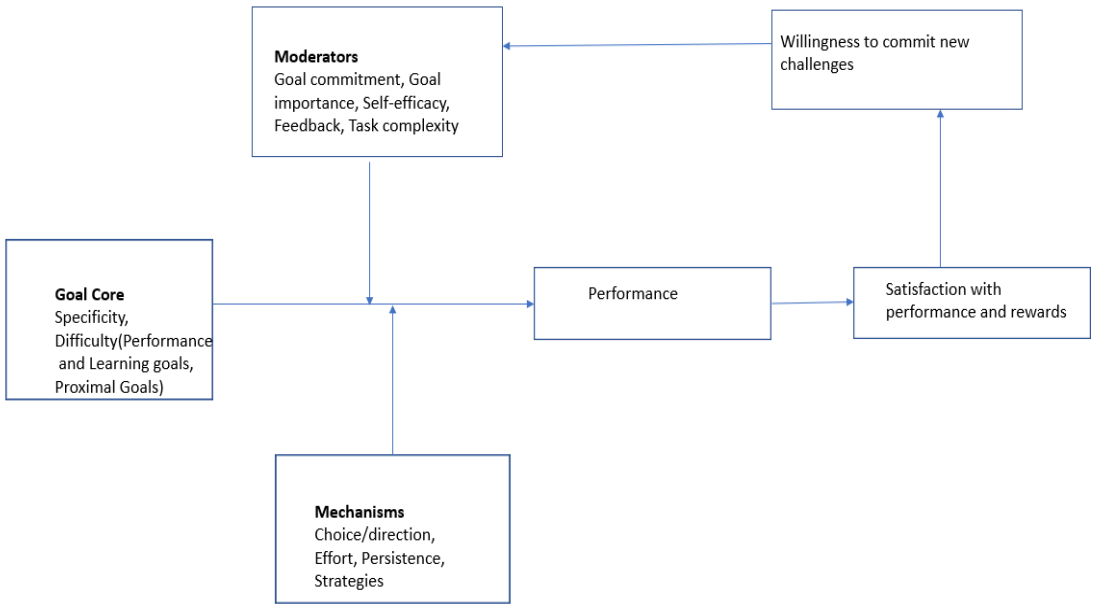

***Figure 1**: Elements of Goal-Setting Theory (Model) [27]*

Advantages of the Goal-Setting Model are as follows:

- Goal-Setting Theory can be used in workplaces to raise the incentives of the employees smoothly and quickly [28].
- Goal-Setting Model can raise the intrinsic or personal motivation of the employee towards work. However, proper feedback should be in place, which can guide the employee to attain its goal [28].

Disadvantages of the Goal-Setting Model are as follows:

- Conflict with the employees and its management in goal setting. There are many instances where conflict happens between the employee and the management in a goal setting. Sometimes unrealistic goals can hamper employee motivation which ultimately leads to a higher attrition rate.
- Understanding of the goal. In case the employee has no idea about achieving the goal or lacks the skill to attain it, then it will be a failure.
- In some circumstances, the Goal-Setting Model does not guarantee employee satisfaction. For example, although Amazon is chosen as one of the best places to work due to its most flexible work pattern, the satisfaction score is low [29]. It is again due to the long work hours and unrealistic goals for employees.

2.2.2 Expectancy Model

The Expectancy Model is built on the mental process of choosing a particular aim from the given options. Expectation Theory or Model was first proposed by Victor Vroom from Yale School of Management. He states, “This theory emphasizes the needs for organizations to relate rewards directly to performance and to ensure that the rewards provided are those rewards deserved and wanted by the recipients.” [31]. The Expectancy Model has served as a rich source for theoretical innovations in several domains like organizational behaviour, leadership, and compensation [74].

From, Vroom’s viewpoint [75], the expectancy model has three key elements:

- Expectancy: effort → performance (E→P)
- Instrumentality: performance → outcome (P→O)
- Valence: V(R) outcome → reward

Where variables E, P, O, R, V denote expectancy, performance, outcome, reward, and valence respectively.
Although in a few studies the outcome is also considered as a key element [74].

Expectancy
Expectancy is one’s effort that results in the attainment of desired performance goal P [31]. It has three moderators which are: Self-Efficacy, Goal Difficulty, and Perceived Control.

Instrumentality
Instrumentality is the belief that a person will receive a benefit or reward when the expectations are met [31]. The reward can be in the form of cash prizes, promotions, awards and so on. Performance incentives which are directly related to monthly pay-out for individual performance are one of the major elements in Instrumentality.

Valence
Valence is the value that the candidate places on the outcome which is based on their motivation to reach the goal. The valence can have three distinct scores [31].
0 = indifferent to the outcome
-1 = avoiding the outcome
+1 = welcomes the outcome

Now, in calculating the motivational force all three components including the valence score will be used.
Motivational Force = Expectancy X Instrumentality X Valence …….. (1)[33]

The advantages of the Expectancy Model are as follows:

- It is based on self-interest. So, it explains the formulas to the people who want to achieve maximum satisfaction and minimum dissatisfaction [32].
- It gives importance to rewards and pay-off.
- It centres around psychological extravagance where the final objective is achieved to attain maximum satisfaction and minimum pain.

The disadvantages of the Expectancy Model are as follows,

- Expectancy Model is an idealistic approach. In a real-life scenario, it is very hard to see people achieving satisfaction after attaining full rewards. The correlation is a hypothetical one [32].
- The application of the theory is limited in real-life scenarios. As reward cannot directly correlate to any performance in any organization. Other parameters also come in this picture like education, responsibility, designation, relationship with your manager.

Fig. 2 shows the Expectancy Model where it is shown that the motivational state is the outcome of the performance.

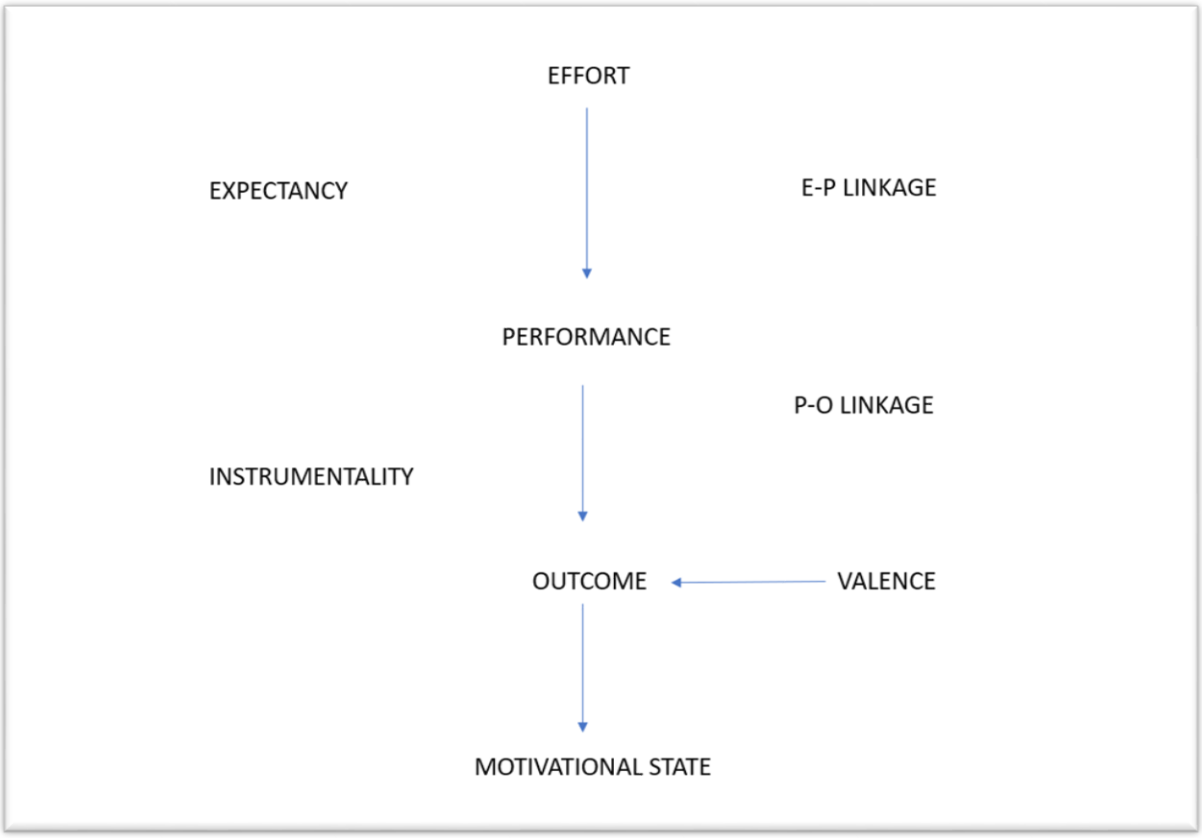

Figure 2: *Expectancy Theory Process [33].*

2.3 Extrinsic Motivation Model

The Extrinsic Motivation Model is built on the extrinsic motivational factors of the human being [34]. Extrinsic motivation is a construct that comes into play whenever there is an activity done to achieve a separable outcome [76]. The extrinsic motivational factors are motivation towards rewards, prizes, fame, money. For example, to buy a luxurious car you need a well-paying job or a well-established business. Thus, to get a good salary you need to motivate yourself to do the work but here the goal is to buy a car. In contrast to the Intrinsic Motivation Model, the extrinsic model does not explain how self-efficacy should be increased, how to achieve the rewards by the process of self-determination.

The Extrinsic Motivation Model works best for those people who are motivated towards monetary rewards, fame, public attention. The model is valid when extrinsic motivation is valid till the reward has its value. Once the reward loses its worth, the extrinsic motivation loses its value [34]. For example, consider a scenario, when companies like Google, Amazon, Nokia fail to achieve a desired profit in the markets and fail to provide the facilities to the employee and finally shuts down their operation. In that scenario, the extrinsic motivation to work in such a company will be diminished. Ryan et al. [76] depict a taxonomy of human motivation in Fig. 3 which can help to discriminate between intrinsic and extrinsic motivation.

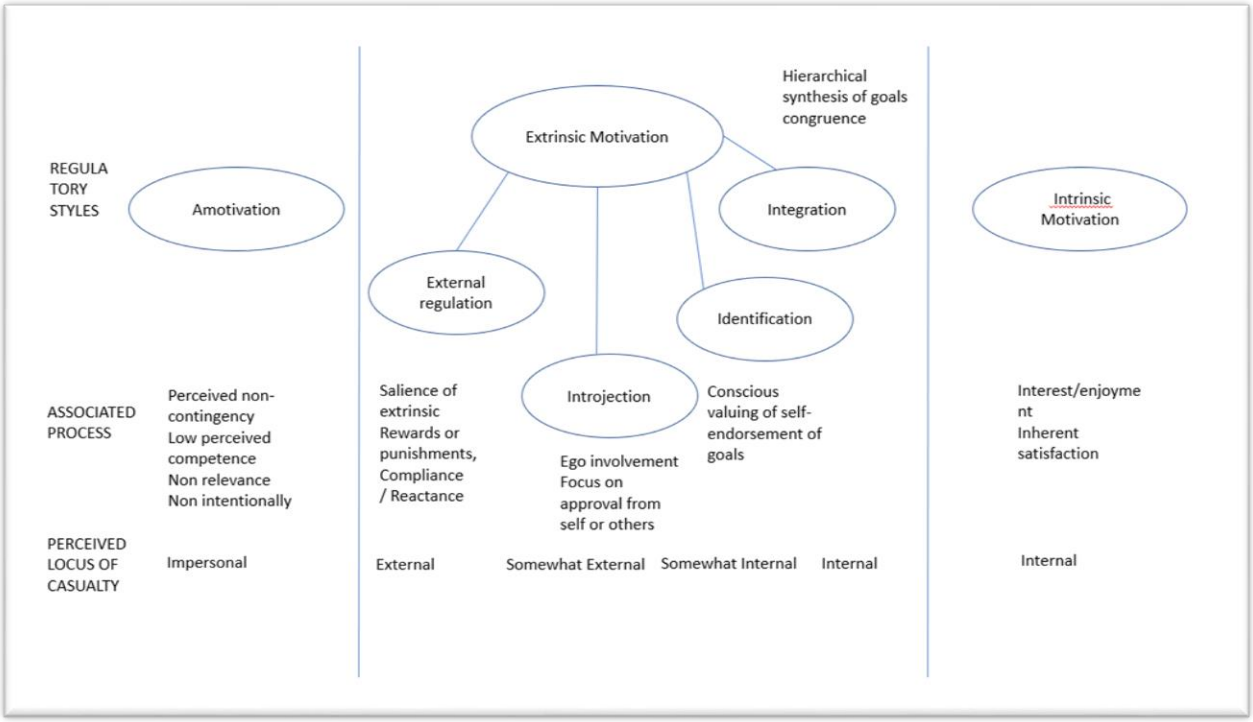

Figure 3: *A taxonomy of human motivation [76]*

There are several limitations associated with the Extrinsic Motivation Model. The Extrinsic Motivation Model cannot explain the cases where the extrinsic model loses its worth. It only outlines the idea of how to achieve the reward. However, sometimes to achieve an externally motivated reward, some intrinsic factors come into play which is not explained by an extrinsic motivation model. In the next section, two Extrinsic Motivation Models: The Job Characteristic Model and the Affective Events Model are elaborated.

2.3.1 Job Characteristic Model

Job Characteristic Model or JCM was first coined by Hackman and Oldham in 1976 [35]. This model has been popular among different management practices. It helps the management in different workplaces to identify certain job characteristics that can affect the outcome of a job [35]. JCM also provides recommendations on how to enrich jobs in an organization [64]. Hackman and Oldham mentioned five job characteristics that can help management practices in identifying the satisfaction rates amongst the employees. Five characteristics are given as follows,

Skill Variety
Skill variety indicates the variety of skills needed to do a certain job. It also indicates how a candidate can gather proper skills to complete a certain job. For example, there are two job responsibilities in a software firm- one is a manual tester, and one is an automation tester. The job of a manual tester is quite straightforward and needs a limited skill set. However, the job of an automation tester includes knowledge of manual testing as well as different automation testing framework. Thus, the second type of job requires more skill varieties and experience.

Task Identity
In task identity, one can find a complete picture of his or her job when the person can identify the type of the job. As an example, a software engineer is associated with the only requirements gathering part of a project and another software engineer is associated with all the end-to-end processes of the project- from the requirement, gathering to project live. So, the second software engineer knows the complete picture of the job and can identify his or her position in the project.

Task Significance
Task significance means when a job has some impact on the lives of society. For example, a heart-surgeon deals with complex cases of the hearts of several people in a town or city. His

or her life-saving procedures change many lives in the city or town. Similarly, a network engineer implements different complex network designs around a certain area in a city which helps many citizens seamlessly connect to the network and complete their work.

Autonomy

Autonomy means when your job has a certain degree of freedom in carrying out tasks like flexible work hours, work from home facilities, decision-making capabilities. In most corporate behemoths, this power lies with only supervisors or managerial positions. But now, it comes to ground-level employees like entry-level Software engineers, HR professionals, and many more.

Feedback

Feedback is important when it comes to job progress. It is always better to appraise an employee in his or her daily activities. It not only motivates the person but also guides the person in his or her duties. It also makes them aware of whether they are going in the proper direction or not. As per Luenendonk [35], “If they are told by their supervisors or managers that they are going a good job, they are likely to feel motivated to continue with how they are doing so far. In contrast, if they are told that they are not performing as expected, then they will respond accordingly and improve their performance.”

Now if all the factors are combined, the Motivation Potential Score will be found. The equation is like equation (2),

MPS = (Skill Variety + Task Identity + Task Significance)/3 X Autonomy X Feedback ……. (2)[35]

A high score of MPS means all five job core characteristics are high which means positive outcome [35]. If the score is low, then the scores of those characteristics can be found and offset by making it high. It means management should take care of low scoring characteristics so that in future they can have a high value. The model has been shown below in Fig. 4.

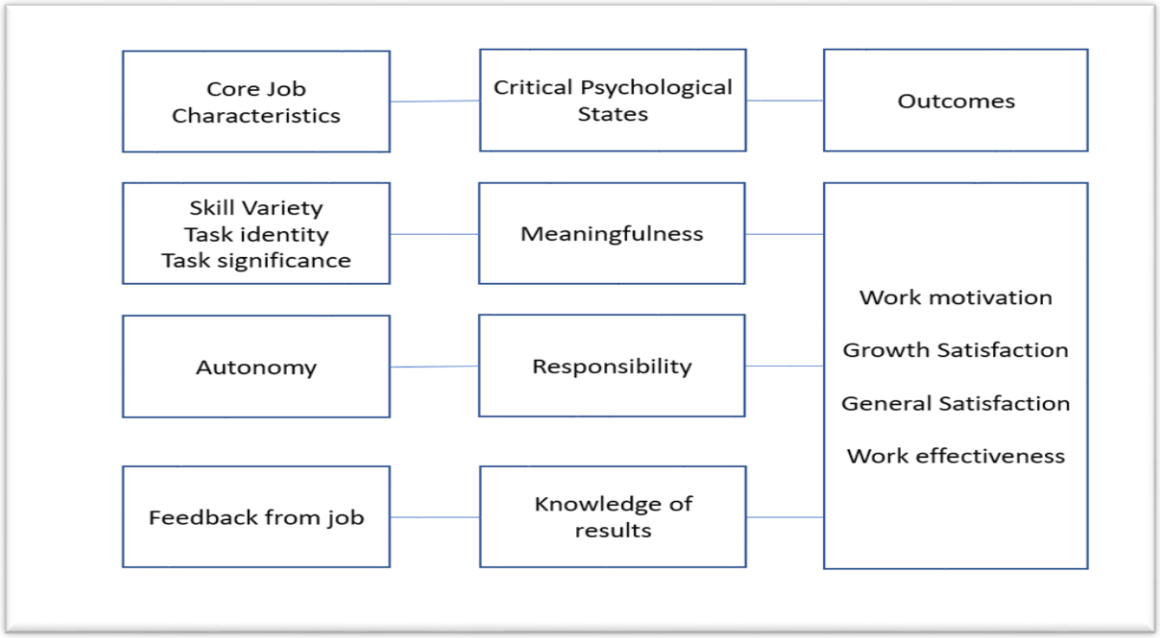

Figure 4: *Job Characteristics Model Overview Diagram [38]*

As per Luenendonk [35], the benefits of JCM are as follows,

- JCM is very easy to understand model which is why it receives popularity in several consultancy firms.
- JCM is a common framework in calculating Job Satisfaction.
- The job characteristics model helps an organization to improve employee performance and satisfaction by adjusting the job responsibilities.

The limitation of JCM are as follows,

- The job characteristics model was developed during the 1980s when organizations had well-defined job roles [36]. But now, the scenario is changed. In current days, an organization has several overlapping roles where this model does not find its feasibility.
- It is not easy to measure the motivation of each employee. It needs individual consultation which makes the whole process lengthy. Thus, in those cases, the job characteristics model does not fit well [37].

2.3.2 Affective Events Model

Affective Events Model was developed by Russell et al. [39]. They proposed the idea of this theory by saying “Affective Events Theory also adds time as an important when examining effect and satisfaction. Research on mood and emotion indicates that affect levels fluctuate over time and that the patterns of these fluctuations are predictable to a great extent”. The

theory proposes that patterns of affective reactions influence both overall feelings about one's job and discrete behaviours at the workplace. Therefore, this model is based on emotion and moods which have direct effects on job performance as well as job satisfaction. The fundamental idea of the theory is based on that affection fluctuates over time. It is not constant. Just like mood swings happen to us all the time during a single day. Similarly, the work environment is also a place where each day is not the same. Based on the workplace settings our emotions and mood change indirectly. The behaviour at work as proposed by Russell et al. [39] are of two types: Affect Driven Behaviour and Judgement Driven Behaviour. AE Model is supported by the Five-Factor where Conscientiousness, Agreeableness, Neuroticism, Openness to Experience, and Extraversion. These factors lead to more job satisfaction [1].

Affect Driven Behaviour

Affect Driven Behaviour is an almost instantaneous reaction to an event [40]. Here the response is almost instant just after the event. For example, during a heated argument with your colleague due to some misunderstanding in a work issue, you stop talking with the person.

Judgement Driven Behaviour

In judgement driven behaviour, the reaction is not instantaneous rather it goes through cognitive review. For the same example that is mentioned above, when the heated argument happened instead of stopping talking, you think about the matter that what went wrong. You are more interested in finding the root cause of the disagreement and based on that you take your next action. So, the process is time-consuming. The AE Model is shown in Fig. 5.

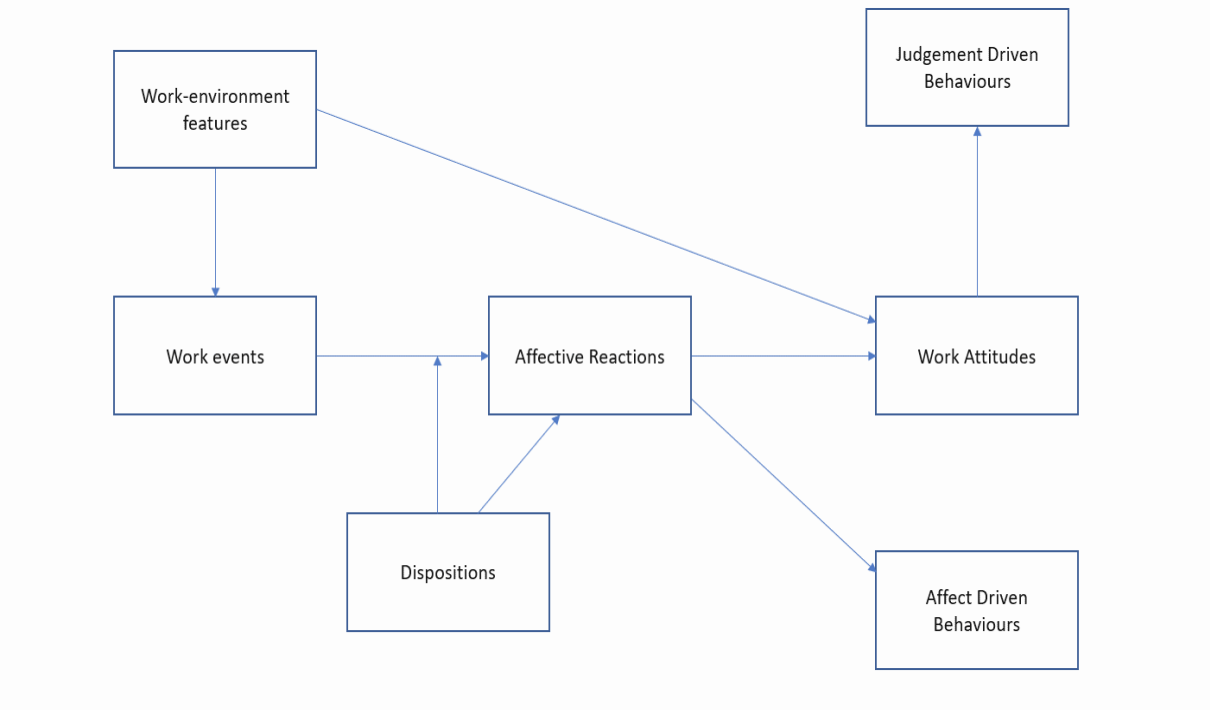

Figure 5: *Affective Events Model: Macro Structure [39]*

The advantages of this AE Model are as follows,

- AE Model is easy to understand and has direct applications in real-life scenarios.
- AE Model gives a satisfactory overview of workplace conflicts and their influence on job satisfaction.
- AE Model is also helpful for employees to act wisely in conflicting situations in the workplace. The same applies to managers too.

The disadvantages of the AE Model are as follows,

- AE Model only deals with job satisfaction. Motivation is a secondary thing here.
- Sometimes job satisfaction and motivation do not go hand in hand [30]. Thus, the Affective Events Theory is not applicable for those cases.
- Motivation is the outcome of an emotion or mood. Thus, Affective Events Theory fails to explain the motivation behind a task properly as it only covers the variety of moods or emotions in different environment settings.

2.4 Hybrid Model of Motivation

Sharp et al. [14] proposed a new model of motivation for software engineers which takes all the factors from the Extrinsic Model and Intrinsic Model. In addition to this, it also considers individual personality, environment Factors, and software engineers' characteristics. This model has paved the way for thinking about motivation in the software engineering context from a different perspective. First, it follows the systematic literature review by Beecham et al. [3] and lists all the motivators and categorizes them into intrinsic and extrinsic motivation factors that are relevant to the software engineering domain (refer Fig. 6).

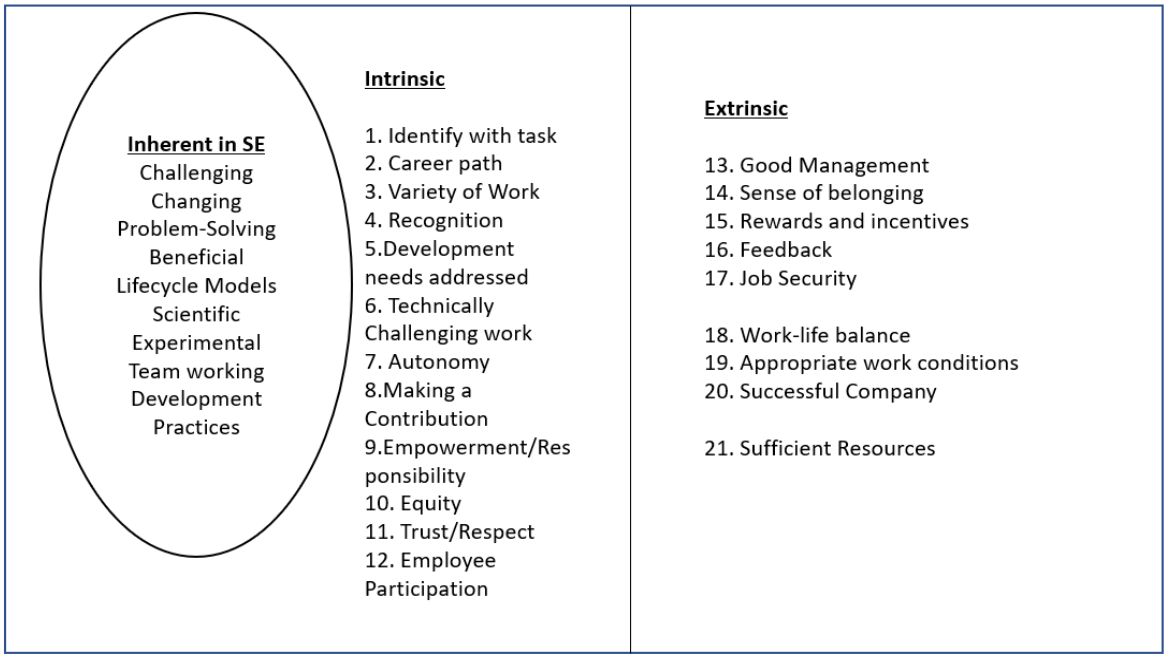

***Figure 6**: Intrinsic and Extrinsic motivators [14]*

Secondly, the MOCC Model briefs about software engineers' characteristics and then shows how these characteristics change concerning the environment and individual personality. Later it is enhanced by categorizing some intrinsic motivational factors into two parts: Inherent in software engineering and organization-specific or job-specific. The further enhancement is done by adding factors like locus of control, job, and goal clarity as intrinsic motivators [14]. Similarly, self-esteem, communication skills are considered as an individual personality and the model was re-generated like in Fig.7.

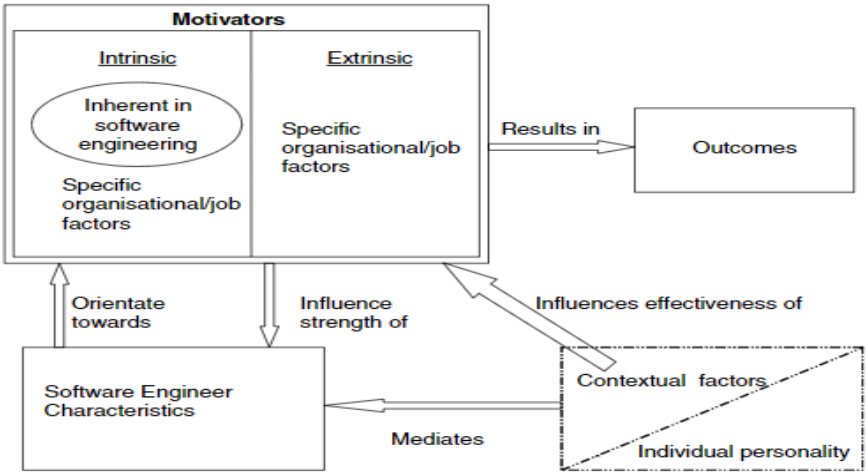

Figure 7: *Enhanced MOCC Model [14]*

In this model, context and individual personality are given an important place. It tries to put an overview of how it affects motivation whether intrinsic or extrinsic which further leads to an outcome.

The advantages of the MOCC Model are given as follows,

- MOCC Model is easy to understand and provides a coherent framework for future studies in motivation analysis of software engineers.

- MOCC Model contains all the components of intrinsic as well as extrinsic motivation. In addition to this, it also considers individual personality and contextual factors in deriving the outcome of motivation.

- It is more concerned about motivation and its outcome not about only job satisfaction. Thus, it may help companies to find out the motivated and demotivated Software engineers.

The disadvantages of the MOCC Model are,

- MOCC Model is based on a systematic literature review. It does not have any practical implementation.
- MOCC Model does not consider the internalized extrinsic motivational factors.

- Most of the portion of the model is built from an organization or management perspective. Thus, there are missing points on personal motivation.

In this thesis, all the models had been studied but MOCC Model was chosen. It is because MOCC Model consists of the factors related to both intrinsic as well as extrinsic motivation. In addition to this, MOCC Model includes the internal factors associated with the person as well as the context. In software engineering, internal factors associated with a software engineer or developer matter and also the context which is the job role or workplace setting [14]. Thus, MOCC Model is a good candidate to start the thesis with.

2.5 Related Works on Motivation Mining

There are several studies performed in the past on motivation mining using different approaches. Some of them are not related to motivation mining but they provided a great basis for this thesis work. The important ones are mentioned below,

Hertel et al. [6] proposed an approach to measure the motivational factors for the software developers working in an OSS project. In this approach, the measurable motives are derived via survey-based results and two social sciences models: VIST and Extended Klandermans. However, it did not consider all the factors related to intrinsic motivation.

Ortu et al. [62] did a study on the JIRA issues and mined emotion from the issue comments. They showed how emotion affects issue fixing time. Here, in this study, it was shown happy developers fixed issues in less time but when they had negative emotions, the fixing time could be longer. This is indeed useful if we apply this for motivation mining too. In addition to issue fixing time, other data points can also be reviewed as part of calculating motivational factors.

Another approach by Da Silva et al. [5] is to find out a motivation strategy for software engineers using a set of motivational factors defined by Beecham et al. [3] with the expectancy theory. It calculated the average dissatisfaction and average motivation force score for each motivator by surveying. It is more about finding the impact of a set of motivators in a team of software developers. However, it did not provide any significant guidance on how to measure those motivational factors.

Graziotin et al. [8] studied to find what were the factors that created unhappiness amongst the software developers. Also, the research was done to support happy developers are

highly productive. Although it is more about finding factors that generate negative emotion, the factors which were listed for unhappiness can be considered as de-motivators too which again matches with the de-motivator list provided by Beecham et al. [3]. However, Graziotin et al. [8] did not provide a way to measure these factors that create unhappiness or the factors that create happiness.

Shahri et al. [13] did an empirical study to find out the motivation using a software-based approach. Here, they collected all the details from the participants using an online persona-based survey to get the persona pattern and their preferences to a certain motivational setting. It was like an experiment where a certain goal was published in an employee portal. Employees were asked to take different online quizzes or events to earn virtual badges, scores to accomplish the goal of getting a promotion. The approach was interesting as it deals with different parameters like Points, Virtual Badges to weigh the motivation of a candidate but is not helpful in mining measurable motivators and de-motivators.

Beecham et al. [3] pointed out several motivators and de-motivators that can be used for the study of the motivation of software engineers. It has also stated which motivators or de-motivators can be considered as intrinsic factors as well as extrinsic factors. These factors are helpful to get an overview of the motivation of a Software Engineer. However, it does not mention any technique to how to calculate them or how to get them directly or indirectly from GitHub or any online platform that hosts Open-Source projects.

Murgia et al. [7] and Ortu et al. [9] implemented an approach to store the issue details from an Issue Tracking System like Jira for further extraction of details related to the issue in emotion mining. But it does not mention any further approach to how to utilize those data in motivation mining. The paper mentioned issue fixing time as one of the factors that affect emotion [9]. However, it did not show how it can be measured.

Sinha et al [23] implemented an approach to find the sentiment of developers from the GitHub commit logs. The technique was quite straightforward to understand but it did not mention why the developers' sentiment changes with the day of the week or with the number of files changed.

The studies which are mentioned above have provided several approaches to mine motivation, related emotion, and sentiment from software repositories. Some of these methods are the basis of the thesis work. However, they don't have any proper technique to measure the motivators or de-motivators and study their variation with sentiment and emotion.

3. Sentiment and Emotion

In this chapter, the general concepts of sentiment and emotion are discussed. Sentiment and emotion both played an important role in this thesis work. It was needed to observe the variation of the sentiment and emotion with different motivators and de-motivators. Novielli et al. [72] mentioned that sentiment and emotion have the following significance in software engineering,

- Identifying sentiment and emotion based on the analysis of developers' communication and textual feedback help in measuring the emotional states of different stakeholders.
- The sentiment and emotion analysis also helps in understanding the antecedents and the impact of different affective states for an individual developer or a group of developers.
- The whole concept of emotion and sentiment helps in providing recommendations to the developers about other developers' sentiment and emotional traits.

Novielli et al. [72] also state that affective computing which is "computing that relates to, arises from, or influences emotion" is now an established discipline. Affective computing is a multidisciplinary field that investigates how technology enables human affect recognition. It also helps to embed emotional intelligence in software systems to support emotional awareness in individuals and groups. Thus, both sentiment and emotion analysis are important in understanding the traits of software engineers working in a software project.

3.1 Concept of Sentiment

The sentiment is a judgement about a particular topic or idea. The sentiment is a quite popular term in social media. For example, on Twitter or Facebook, a video or a post receives a myriad number of comments. From the comments, we can judge whether the video or the post creates a positive impact on the netizens or not by going through the sentiment expressed through their comments. Similarly, in software engineering, when a team of software engineers work on an industry-based project or in an open-source project, they comment on the tasks of other software engineers. Based on the comments or opinion, a software engineer can understand whether his or her tasks are going in a proper direction or not.

As per Cambridge Dictionary, Sentiment is a thought, an opinion, or idea based on a feeling about a situation or way of thinking of a situation[41]. As per Collin's Dictionary, Sentiment can be defined in three contexts [42]. In variable noun context, a Sentiment is an attitude of

people based on their thoughts and feelings. In countable noun context, it is an idea or a feeling that someone expresses in his or her feelings. In an uncountable noun context, the definition is like this "**Sentiment** is feelings such as pity or love, especially for things in the past, and maybe considered exaggerated and foolish." [42]

3.2 Concept of Emotion

Emotion has significance in any software engineer's personality. There are various definitions exist for Emotion [50][51][52] in different contexts. Emotions are intuitively well-understood ideas, but defining them properly is difficult [39]. Emotion is a reaction to an event, or it is a range of reactions that are interconnected with each other. Intuition tells us that there are various types of emotion, each of which deals with a unique phenomenal experience and has diverse repercussions for individuals and organizations [39]. In software engineering, emotion has a great significance as based on emotion, the motivational outcome varies a lot. In the software engineering industry, each day is new learning for a software engineer. Therefore, there will be a range of emotions in a single moment that can affect an engineer's actions in the workplace. It is, therefore, a need to understand what emotion means. Next, the different definitions of emotion are discussed. Although there is no concrete definition to define emotion, it is useful enough to get an overview of it.

Shaver et al. [53] mentioned five different types of emotion that are evident in human nature. Gordon et al. [65] mentioned seven types of emotion frequently occurring in software developers. After studying both kinds of literature, those emotions are described below.

Fear

Fear is one of the most common emotions among human beings. It is aroused automatically in the central nervous system when the person feels threatened, alone dealing with difficult situations. It is also observed when something bad happens without any explanation and knowledge, where the total situation is out of control of the person, the fear becomes evident. The outcome of this emotion is crying, yelling, thinking of doing something wrong to avoid the situation, or thinking of committing suicide or killing someone to save himself or herself.

Sadness

When something life-threatening event is coming, a person is aroused with fear. But sadness comes when the life-threatening or any fearful event is over. It means it deals with the result of such an event. When someone dies, or when someone already faced brutal treatments or faced life-threatening incidents, he or she becomes sad with those happen-

ings. The person may possess a negative outlook towards life. The person may avoid social contact to hide sadness. Such kind of emotion is quite prevalent amongst rape victims, refugees. In the software engineering context, it may happen with those employees when they are unable to deliver a product within the deadline and it directly affects their employment. The engineer is sad as he lost his job and sees no sign of getting the next one. However, the victim can come out of sadness by self-controlling of the emotion and other preventive measures like therapy, social contact and so on.

Anger

Anger comes when something that the person deserves, goes to another person. Or the person is deprived of the deserving thing. It can also happen when the person thinks someone trying to create pain in his or her life or trying to create a bad impact on his or her life. Expressions of this emotion will be like showing teeth, yelling, cursing. But it can be self-controlled by the proper use of anger management principles.

Joy

Where sadness is associated with negative outcomes, joy is associated with positive outcomes. Positive outcomes mean success in a business or being successful at the job or task. From a software engineering perspective, it means when a software engineer successfully delivers a product fix an issue or implements a new feature successfully. Joy makes people more outgoing and cheering as opposite to sadness where people become more introverted and withdrawn.

Love

Love has a similar characteristic to joy. The difference is joy is not associated with any outcome. Love is more about liking certain things in life. A person has a love for dogs or any kind of pet. It is since a person likes certain characteristics of a pet that attracts him. Similarly, a human being can love another human being. In the software engineering context, love can be explained as a software engineer's love for coding or learning new technologies. Or the engineers' love for working in different countries for an employer to meet more like-minded software engineers.

Happiness

Happiness comes when you feel satisfied with your life or achievement [66]. Perfect happiness comes when you achieve all the desired goals, and all your needs are fulfilled. From a software engineering perspective, this can be achieved when a developer is satisfied with the performance of a project. Or, more specifically, when the developer gives a spectacular performance in his or her work and achieves the result he or she wants.

Disgust

According to Merriam-Webster dictionary, disgust is a transitive verb that means to provoke or to loathe [67]. In other words, an emotional repulsion or aversion. If we consider disgust in a software engineering context, it can be a software developers' aversion to work in a toxic work environment or repulsion towards certain kinds of work. This version can be due to career concern or lack of risk-taking nature. In addition to this, other factors also play a crucial role.

3.3 Sentiment Analysis Tools

Sentiment Analysis is a process of extracting sentiment from a text, video or audio by using different techniques like natural language tokens, natural language processing. In this section, an overview of different sentiment analysis tools is given. Most of them are available in the market and used in this thesis work.

SentiStrength

SentiStrength is an automatic sentiment analysis tool that can analyse texts up to 16000 English words per second [43]. It can also analyse sentiment on texts written in Arabic, Spanish, Italian, and Finnish. It has human-level accuracy in scoring the sentiment from a small web text [43]. The SentiStrength has two types of sentiment strengths: negative and positive. Apart from that, the scores from SentiStrength can be expressed in different scales. In the binary scale, it gives a score of either 1 or -1. 1 means positive sentiment and -1 means negative sentiment. SentiStrength also gives a score on a trinary scale. In that case, the score will be either 1 or 0 or -1. Here, 1 denotes positive sentiment, -1 denotes negative sentiment, and 0 indicates neutral sentiment. In addition to this, it has also a singular scale score where the score value ranges from -4 to +4.

The advantages of SentiStrength are as follows,

- SentiStrength is easy to use, and it is easily available from the web.
- SentiStrength comes with a dictionary list of 16000 words with the score. We can customize that score to fit our context in sentiment analysis.
- Different types of scoring scales which we can use based on our context

The disadvantages of SentiStrength are as follows,

- SentiStrength is not available for Linux platforms. It is only available for Windows OS.

- The scoring of the words can be customized so that it can score sentiments as per the context. For example, a sentiment that is negative in a corporate workplace may have some positive sentiment in the entertainment industry. Thus, if we have to analyse sentiments from these industries, we can modify the scores in the dictionary or add our own words with scores to work as per the context. It makes it quite restricted as the whole process depends on the presence of words nothing on the overall context.

TextBlob

TextBlob is a python library for processing textual data [44]. Here, textual data means social media texts or comments, text format data from several online documents and so on. It provides an API interface for Natural Language Processing tasks which includes parts-of-speech tagging, sentiment analysis, classification, and so on. In TextBlob, scoring ranges from -1 to 1. 1 indicates highly positive sentiment or polarity whereas -1 indicates highly negative sentiment or polarity. 0 means neutral sentiment.

TextBlob has a set of extensive features like noun-phrase extraction, part-of-speech tagging, sentiment analysis, tokenization, spelling correction, and classification of sentiments using Naïve Bayes, Decision tree algorithms [44].

The advantages of TextBlob are as follows,

- TextBlob is easy to understand and use for programmers.

- TextBlob is supported by both Python 2 and Python 3.

- Sentiment scoring is better than SentiStrength as wordlist is not limited.

The disadvantages of TextBlob are as follows,

- No neural network integrated into it. As a result, the prediction is not always good.

- The whole analysis process using TextBlob is time-consuming.

- Wordvector (refer to **Appendix B**) integration is missing [45].

Vader

Vader (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media [46]. The scoring of Vader is done for each word by summing the valence score, and then adjusted according to the rule, later normalized between -1 and +1 [46]. That's why it is called **Compound Scoring**. Here, -1 means extreme negative sentiment whereas +1 means extreme positive sentiment. 0 denotes neutral sentiment. It is also possible to set threshold scores for sentiments. The threshold scores for different types of sentiment are given below [46].

Negative Sentiment: Compound Score < -0.5

Neutral Sentiment: Compound Score between -0.5 and +0.5

Positive Sentiment: Compound Score > 0.5

The advantages of Vader are as follows,

- Vader is easy to understand and implement.
- With NLTK, it can work with longer texts.
- The variation score range makes it easier to classify the sentiment of the texts.

The disadvantages of Vader are as follows,

- For longer texts, it is slow to produce results of the sentiment score.
- It is based on social-media texts. Thus, it may not give proper results if we apply that in the texts created in the context of Software Engineering Team projects.
- As researchers introduce their own set of threshold values for sentiment classification in Vader, later it may create confusion when the same text creates different sentiments in different contexts.

Flair

Flair is a simple natural language module or library that is developed by Zalando Research Group in collaboration with Humboldt University, Germany [48]. PyTorch is the base of the Flair as Pytorch is one of the best deep learning frameworks that can handle complex tasks.

Flair has several pre-trained models to accomplish the tasks like name entity recognition, part-of-speech tagging, text classification, and custom training models [47].

An important part of Flair works on Contextual String Embedding. Next, it is described.

Contextual String Embedding

In NLP, Context is very important. To predict the next word or character based on the previous character or word is the basis of sequence modelling [47]. Contextual String Embeddings is done using the concept of a trained character language model that is again based on sequence modelling. Here's the overview diagram of contextual string embedding in Fig. 8.

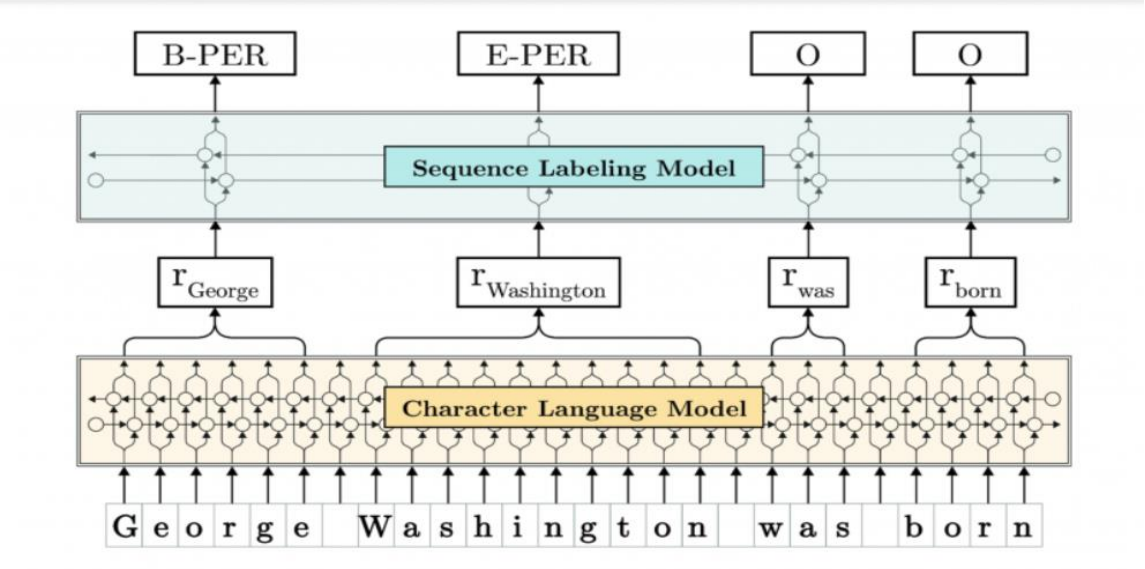

***Figure 8**: Contextual String Embedding [47]*

Advantages of Flair are given as follows,

- Flair has all the strong and robust word embeddings models integrated like GloVe (refer to **Appendix B**), BERT (refer to **Appendix B**), Elmo, and so on [47].
- Flair has also embedded Contextual String Embedding which makes it capable of predicting the next set of characters or words based on the context.
- Flair supports many languages. Thus, it is not only used for only English language but also in other languages. Flair is now supporting English, German, Dutch, Arabic, Finnish, Italian, and French languages [49].

Disadvantages of Flair are given as follows,

- Flair has no scale of neutral sentiment. So, it can predict only Positive sentiment and Negative sentiment.
- For a large set of text classification, Flair takes a huge time. For each text classification, it loads the pre-trained model for each run.
- As Flair takes huge time for text classification on large datasets, it is not suitable for computers with normal CPUs. GPU will be a better option.

There is another sentiment analysis tool developed by Calefato et al. [4] at Politecnico Di Bari. This tool is Senti4SD. It is built on the base of SentiStrength but by enhancing it by leveraging the benefits of the neural networks [4]. The dataset is from StackOverFlow, which consists of 3.8 million questions, 5.9 million answers, and 11.6 million comments [4]. After labelling the data with gold labels, it is trained by the support vector machine supervised algorithm [4] (refer to **Appendix B**) to get the sentiment score and its classification for the dataset. However, it has a codebase that is old and does not support a higher version of Python like 3. x. If the whole classifier is running with downgraded versions, other functionalities like classifiers from Sklearn do not work. It has the docker version which has the accessibility issue as the results generated by Senti4SD are only available in the Docker environment, you cannot share it outside. To avoid this, there is a way around which is to create a shared directory in a Linux environment and add that path during the docker environment set-up. This leads to an overhead as Senti4SD creates two files for each type of sentiment. When the input file size is several megabytes or when a file contains more than 10,000 data records, then it involves extra manual effort. More importantly, the calculation of sentiment should be done independently, and it cannot be integrated with any Python or R script due to the complexities involved. Due to this, it was removed from this sentiment analysis study.

3.4 Emotion Analysis Tools

Several tools in the market can help us to extract emotion from textual data. These tools help extract emotion from social media texts, comments from open-source software projects, texts from online forums and so on. Text2emotion and EMTk are such tools for emotion analysis from social media or any texts from different online platforms.

Text2emotion

Text2emotion is a python library that helps in extracting emotion from the texts. It is developed by Gupta et el. [54]. This library right now supports five different emotions: Fear,

Sad, Happy, Angry, and Surprise [55]. Text2emotion follows the following steps to bring out the final emotion from a text [55].

Text Pre-processing

In this phase, all the texts are passed through data cleaning to make them suitable for any kind of emotion analysis [55]. In this stage, the following actions are observed,

- Removal of unwanted text part from the message.
- Perform natural language processing on the texts.
- Bring out the well-processed texts from the pre-processed texts.

Emotion Investigation

This phase is crucial as it finds all types of emotion in the well-processed texts. The following actions are observed in this phase,

- Find appropriate words that signify emotions or feelings.
- Check the emotion category for each word.
- Storing the count of the emotions relevant to the words found.

Emotion Analysis

This is the final stage where emotion will be produced for a particular text message. The following events are part of this,

- The output will be in the form of a dictionary.
- The emotion and its corresponding values will be in key-value pair.
- Higher the score of a particular category of emotion, we can deduce that is the emotion belongs to that message.

Advantages of Text2emotion are as follows,

- Text2emotion is easy to use. For a python novice, this can be a great tool to analyse emotions.
- The emotion categories and the scores are easy to understand.
- The Python library has great online support as the developers are easily reachable.

- Good for analysing software engineering related texts.

Disadvantages of Text2emotion are as follows,

- Although Text2emotion is easy to use, for critical issues it has no support documents.
- In the case of emotion categories with the same highest emotion score, the analysis seems to be confusing for some users who are new to this tool. For example, a sentence can produce two categories of emotion like sad and happy with the same highest emotion scores. In these cases, any novice developers will be in a dilemma which is the actual emotion in the sentence.
- Text2emotion does not have fair popularity amongst the Python developer community as it is quite new to the open-source market. Therefore, support from the python developer community is sometimes missing. Only the developers who developed this library can help with this.

EMTk

EMTk is an emotion mining tool kit for training custom sentiment and emotion from the text [73]. The tool kit has two modules,

- Emotion mining module which is responsible for training custom emotion classifiers from the text. It is mainly based on the 5K posts from the *StackOverflow* forum [73]. Thus, it is suitable for emotion classification in software engineering texts.
- Emotion-polarity module which is responsible for the classification of sentiments on technical corpora from the developer's communication channel [73].

EMTk is beneficial in the following scenarios,

- If we want to assess the polarity of sentiment without training our own classification model.
- If we want to assess the emotion expressed in the technical texts in the software development domain without training our own classification model.

EMTk is not helpful when,

- We want to assess the emotion expressed in the issue comments as it is mostly trained for commit comments [73].
- We want to assess the emotion on both issue and commit comments and the dataset is large. It is because the codebase in GitHub has an old version of Python which does not work properly in all environments. Dockerized version can be used

in place of that, but a lot of manual tasks are involved in the case of the large data set. As the files that are generated by the classification model are stored in the docker environment, they can't be directly used in the host environment. We need to copy to and from a shared path. Thus, it makes the process quite time-consuming. Also, the classification model for emotion works on commit comments only. Thus, efficiency may not be good.

After comparing both the libraries, Text2emotion was chosen due to its ease of binding with the python code. EMTk is not helpful in this regard because you cannot embed the EMTk functionalities with a python code at your convenience. EMTk should be executed independently for just emotion analysis for the commit texts or issue comments. Thus, there will be overhead when you have to deal with a large dataset and at the same time, you need to integrate with your codebase. Considering the dockerized version, you cannot call it from your python code easily. There are indirect ways, for example, making two docker images of the EMTk and the python codebase respectively then running the single docker file. However, in this case, too, the optimization of the codebase is needed to call the EMTk methods from the python script. In addition to this, the codebase of the EMTk is not supported by the higher Python versions like 3. x or above.

4.Overview of Open-Source Software Development

This chapter contains an overview of the Open-Source Software Development Process. Open-Source Software Development is a process by which a Software's source code is publicly available. In this Software, all the source codes are available with an open-source license to study, change, and improve its design [56]. Mozilla Firefox and Libre Office are some of the most popular Open-Source Software Products [56].

4.1 Open-Source Software Development Lifecycle

Open-Source Software Development is quite a lengthy process like commercial software development. It is due to the fact there is no proper rigid method followed in open-source software development. Here, the typical waterfall model does not work. Even for each type of work, the process will be different. That is why an overview of the Open-Source Software Development lifecycle as stated by Wu et al. [57] is given in Fig.9.

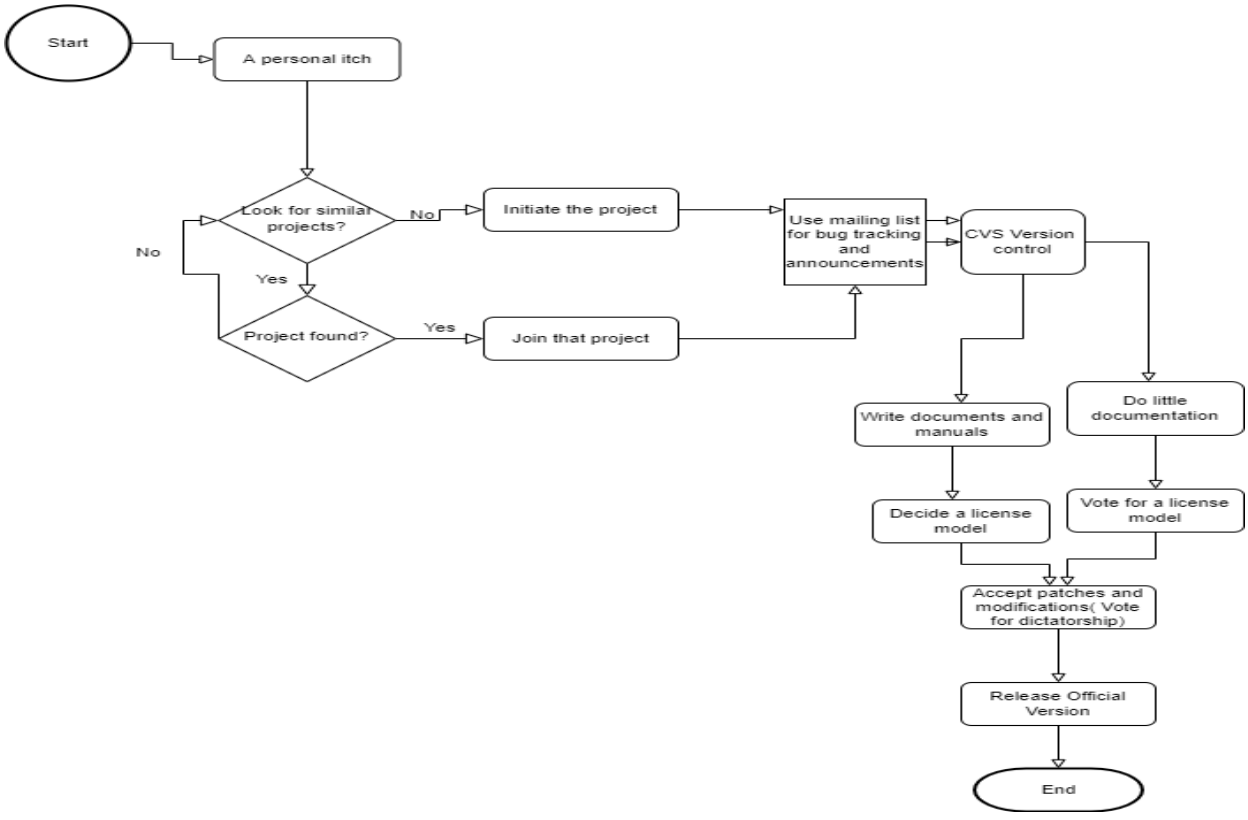

Figure 9: *Open-Source software development lifecycle [57]*

Whereas Haddad et al. [59] described the lifecycle of developing a feature in Open-Source Software Development. This can help us to get an overall idea of how OSS Feature works. During any new feature development, a new feature request is raised first. This request is then visible to all the community members of the project. Based on the type of feature, it is prioritized after the discussion with all members part of the Software Project. For a particular feature, a proposal is done who will lead the project [59] and the release date is set. After a thorough discussion in architecture designing, it must go through several developments, testing. Generally, during development, several bugs may appear which can be tracked differently by bug tracking tools like Jira, GitHub issue tracker and so on. In the present day, most of the development and testing are associated with continuous feedback. That is why CI/ CD is quite appropriate for it. Generally, in some OSS projects, a weekly build or nightly build is made based on the automation build [59]. But this is applicable for only larger projects, for smaller projects CI/CD feedbacks are generally instant or made within an hour or two. Once, all the reviews are done and all the tests are passed, it is passed to the next round of discussion between project contributors and maintainers. Once the feature looks good and stable, it is released publicly.

However, the lifecycle model that is mentioned by Wu et al. [57] seems to be fair in terms of understanding the whole process in Open-Source Software Development. The process that is mentioned by Wu et al. [57] is quite different from Haddad et al. [59]. According to Wu et al. [57], any developer who wants to upgrade their skill set or is interested in doing some interesting programming looks for a project in any open-source repository. If he or she does not find that, the person himself or herself start building a project aligned to his or her interest. If he or she does find a project that sparks interest in the person, he or she joins the project. After joining, the person tries to communicate with the collaborators using the mailing list and proposed a bug that she or he wants to work in. Once he or she starts working on a bug or development, the person involves in the documentation related to the bug or development changes. After the development is developed and tested properly, the developer or the team applies for a model that needs a license. Once the license is provided and official patches are accepted, the person with the team or person himself or herself decides the release date for its official launch. Thus, the lifecycle model here is quite generic and simple to understand. It is not specific to any part of the software; however, it is a BlackBox concept that a non-technical user also understands.

4.2 GitHub Repository

GitHub Inc is a provider of internet hosting for software development and version control using Git [58]. It has the features of source code management and distributed version control functionalities that Git provides [58]. It has several features like bug tracking, integrated Kanban Board, Integrated CI/CD facilities with the options for deployment in cloud vendors like AWS, Azure, Google Cloud, Alibaba Cloud, Heroku, RedHat Openshift, and so on. GitHub also provides integration facilities with code quality inspection platforms like Sonar Qube, Embold. It also provides features like security vulnerability detection like detecting .env files that contains important application credentials and deleting those pull requests which contain the configuration or environment credentials files.

Apart from the above features, it also provides audit logs that can help any team to check the history of unauthorized or risky activities that are associated with an open-source software project. Now, GitHub is also available from mobile supported by iOS or Android. Despite the above-advanced features, GitHub has some serious negative attributes,

- It suffers from security vulnerabilities. Although several features are there to protect an open-source project repository, there are cases where some projects fail to leverage such benefits and, in those cases, GitHub cannot protect such projects from security breaches. Generally, these cases happen when a team does not have much knowledge about code vulnerabilities or software security.
- Generally, if someone is contributing to open-source projects, GitHub has free services that developers can avail themselves of. But some projects although hosted in GitHub are not open-source projects, in those cases, to avail the advanced services like cloud deployment, GitHub unlimited API access, or advanced auditing, unlimited code scanning comes with a huge price tag. For larger teams, this cost can go up based on the requirements.

4.3 Different Types of Issues in GitHub

In GitHub, there are different types of issues associated with a project. These issue types are identified by Labels. Labels indicate different issue categories that can help other team members to identify the type of the issue. Hence, the team members can be able to know what to do with the issue. GitHub online documentation states the following labels which are used quite frequently in any project [60].

LABEL	DEFINITION
Bug	Bug indicates an unexpected behaviour in the software. Or an unexpected error that is affecting the software.
Documentation	Documentation label indicates a need for improvements or additional details in the form of documents.
Duplicate	Duplicate indicates similar types of issues or pull requests.
Enhancement	Enhancement indicates an improvement of an existing functionality of a software
Feature	The feature is also frequently observed in many software projects hosted in GitHub. There is a thin line difference between Enhancement and Feature. Enhancement denotes an improved functionality that will be added to the existing software. Whereas Feature denotes a completely new part in the software project, it is related to the existing software but not an improvement to the existing functionality. It needs to be developed independently. However, it should be useful to the software.
Good First Issue	Good First Issue indicates a good issue for the first-time collaborators.
Help Wanted	Help wanted indicates that the owner or maintainer of the project needs some help on an issue.
Invalid	The invalid label denotes the issue of a pull request or a discussion that is not relevant to the project.
Question	The question indicates there are more clarification or information needed for an issue or pull request.
Won't Fix	WontFix signifies that the issue or pull request will not continue as it may not have any chance to work in future.

Table 1: *Different labels in GitHub and their definitions*

4.4 PyGitHub

PyGitHub is a python library that uses GitHub API version 3 [61]. With this library, we can connect to the GitHub repository and fetch details about it like issue number, its label, priority, users, collaborators, comments, commits, and so on. Even with this library, we can set the label of an issue, create an issue. In addition to this, we can raise a pull request related to an issue using a python script- we just need to import this library.

A small code snippet of PyGitHub initiation in Fig.10,

```
from github import Github

# using an access token
g = Github("access_token")

# Github Enterprise with custom hostname
g = Github(base_url="https://{hostname}/api/v3", login_or_token="access_token")
```

Figure 10: *Initiation of PyGitHub in a Python program [61]*

Next step discusses the connection to GitHub repositories using PyGitHub in Fig.11.

```
for repo in g.get_user().get_repos():
    print(repo.name)
    repo.edit(has_wiki=False)
    # to see all the available attributes and methods
    print(dir(repo))
```

Figure 11: *Connecting repositories using PyGitHub [61]*

The advantages of the PyGitHub library are as follows:

- Easy to install for any novice developer. Also, the syntax is easy to understand.
- With GitHub generated token, it is easy to call.
- Direct interaction with GitHub repositories makes it a viable choice when access to GitHub is restricted.

The disadvantages of the PyGitHub library are as follows:

- Using a single user token, the API data fetch is limited in PyGitHub. To get more data, premium membership is needed which is costly.

- Developers should be fluent with the all-important methods of PyGitHub otherwise some crucial utilities will be hard to explore.

- Lots of pre-processing is needed before utilizing the results from PyGitHub as the data format is not always acceptable in some project environments.

PyGitHub library is one of the building blocks of this thesis work as it helped in extracting issue and commit details for an open-source project from GitHub.

5. Research Methodologies

This chapter elaborates on the methodologies that are implemented to carry out the thesis work. The basis of the thesis is based on Murgia et al. [7], Ortu et al. [9][62], Rath et al. [11], and Sinha et al. [23]. In addition to that, the approach to find out motivational or de-motivational factors from different categories were taken from Beecham et al. [3] and Sharp et al. [14]. The process flow that was undertaken in this thesis work, is shown in Fig. 12.

5.1 Overview of the Mining Process and its Components

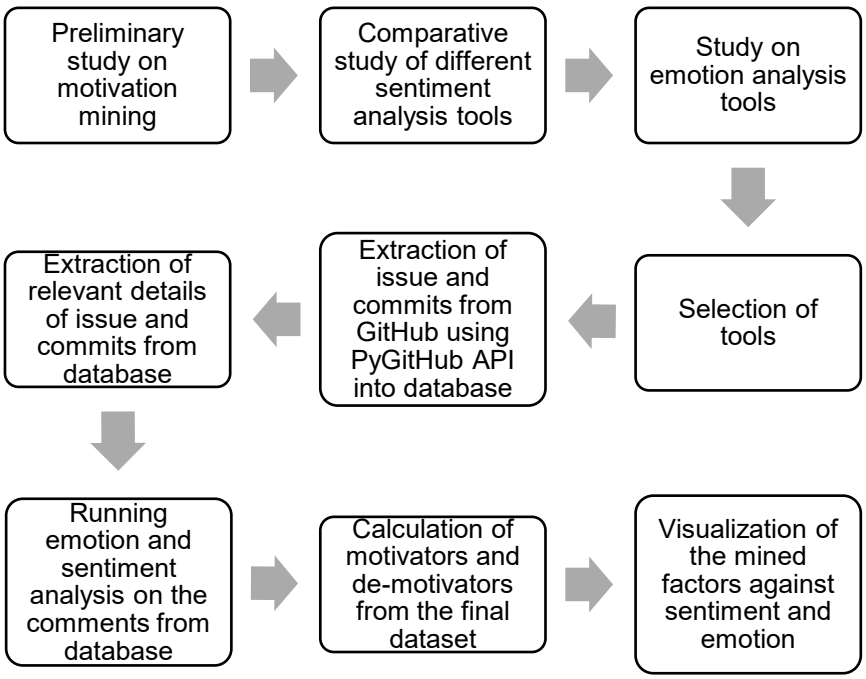

Figure 12: *Motivation mining process*

Fig. 12 describes the process of connecting to GitHub and then fetching data using PyGitHub API. The fetched data is then stored in the database. Next, different scripts are executed to generate relevant extracts for the issue and commits in separate CSV files. From the CSV files, metrics related to motivators or de-motivators are calculated. Finally, the factors are visualized using the Power BI analytics tool. Power BI analytics tool helps to get insights from the mined data.

In the next sections, the whole process is elaborated in a step-by-step manner.

5.2 Preliminary Study: Tools Comparison for Sentiment Analysis

Sentiment analysis is one of the most important parts of this thesis work apart from emotion analysis and data mining (refer **Appendix B**). The entire motivation calculation depends on this because whenever developers commit or comment on an issue, they convey their sentiment and emotion through this. In section 3.3, the different sentiment analysis tools which are mentioned, are quite useful in analysing sentiments from social-media texts or online resources. The same tools were used in this thesis paper to observe their performance in the Software Engineering context. In this case, a basic comparison study was done on four repositories with a small test data set with 20 to 30 comments. The repositories were**: Apache Airflow, Apache Sharding, Apache Skywalking, Apache Traffic Version**. At first, a set-up of the token was initiated from GitHub and passed to the PyGitHub module. It fetched all the issues and their related comments for all four repositories. Then, the results were saved in a CSV file and then all four libraries were executed on it. Finally, the results were stored in the same CSV file with a different column.

For the Apache Airflow project, the following results were observed for 29 Issue comments in Table 2.

Sentiment Analysis	Positive	Negative	Neutral
Flair	17(58.6%)	12(41.37%)	0(0%)
Vader	23(79.31%)	2(6.89%)	4(13.79%)
TextBlob	23(79.31%)	4(13.79%)	2(6.89%)
SentiStrength	23(79.31%)	6(20.68%)	0(0%)
Manual	23(79.31%)	3(10.34%)	3(10.34%)

Table 2: *Sentiment analysis on Apache Airflow issue comments*

For the Apache Sharding project, the following were the results for 24 Issue comments in Table 3.

Sentiment Analysis	Positive	Negative	Neutral
Flair	16(66.66%)	8(33.33%)	0(0%)
Vader	23(95.83%)	1(4.16%)	0(0%)
TextBlob	23(95.83%)	1(4.16%)	0(0%)
SentiStrength	23(95.83%)	1(4.16%)	0(0%)
Manual	23(95.83%)	1(4.16%)	0(0%)

Table 3: *Sentiment analysis on Apache Sharding issue comments*

For the Apache TrafficVersion project, the following details were found for 30 Issue comments in Table 4.

Sentiment Analysis	Positive	Negative	Neutral
Flair	0(0%)	30(100%)	0(0%)
Vader	23(76.66%)	7(23.33%)	0(0%)
TextBlob	20(66.66%)	10(33.33%)	0(0%)
SentiStrength	6(20%)	24(80%)	0(0%)
Manual	21(70%)	5(16.66%)	4(13.33%)

Table 4: *Sentiment analysis on Apache TrafficVersion issue comments*

For the Apache Skywalking project, the scores for 21 Issue comments were observed in Table 5.

Sentiment Analysis	Positive	Negative	Neutral
Flair	3(14.28%)	21(85.71%)	0(0%)
Vader	12(57.14%)	8(38.09%)	1(4.76%)
TextBlob	10(47.61%)	10(47.61%)	1(4.76%)
SentiStrength	14(66.66%)	7(33.33%)	0(0%)
Manual	11(52.38%)	9(42.85%)	1(4.76%)

Table 5: *Sentiment analysis on Apache Skywalking issue comments*

From the above results, it is seen that the predictions from TextBlob, Vader, SentiStrength are close to manual review. However, SentiStrength was removed from the list because it

has a very restricted set of words that decides the tone of a sentence. A presence of "not, no or does not" creates a negative sentiment although the sentence is either neutral or positive. Similarly, the presence of just some happy phrases like "Good!", "Great!" makes a sentence positive although it has negative connotations. Thus, options were limited to only three tools Vader, TextBlob, Flair. Flair does not have a neutral sentiment option. However, it is necessary to find out how much sentiment score is varied with the emotion. That is why it was necessary to run the study on a much larger dataset of issue comments to find out which tool was aligning well with the emotion. Generally, it is observed that negative emotions like Fear, Sad, Angry associated with negative sentiment. To achieve this, 3.5k issue comments from 35 repositories were extracted using PyGitHub and then the three tools were executed on it along with the emotion extraction tool Text2emotion. After the process was over, saving the scores from each library in a CSV file was done. Then, the average of sentiments from each library for each emotion had been used. The results are shown in Tables 6 to 8.

Emotion	Flair Sentiment Average
Fear	0.942259626
Happy	0.920903065
Sad	0.897981414
Surprise	0.916365338
Angry	0.960987829

Table 6: *Average of sentiment score per emotion for Flair*

Emotion	TextBlob Sentiment Average
Fear	0.077481882
Happy	0.136221241
Sad	0.066983793
Surprise	0.125168405
Angry	-0.002980055

Table 7: *Average of sentiment score per emotion for TextBlob*

Emotion	Vader Sentiment Average
Fear	0.372623572
Happy	0.250343136
Sad	0.196813376
Surprise	0.204250204
Angry	-0.351539573

Table 8: *Average of sentiment score per emotion for Vader*

From the results, it is observed that Flair is giving quite contradicting sentiment for negative emotions like Fear, Angry, and Sad. The sentiment score is on the highly positive side. Thus,

it was set aside from the consideration. Due to this, there were only two options Vader and TextBlob. For Vader, although the sentiment score for Fear and Sad is not very highly positive, TextBlob still fits well with all the emotion categories. In TextBlob, Angry is well fitted with a negative sentiment average. Fear and Sad although is on the positive side in sentiment average, still is much lower than Vader. In addition to this, using Flair has disadvantages related to CPU. Thus, after applying it on a large dataset of 26K Issue comments and over 6K commit comments took more than 24 hours. Hence, it was not a feasible solution in the given CPU based environment. Therefore, it was decided to go with TextBlob for the sentiment scoring along with Text2Emotion to get the overall tone of the developer or commenter during commenting on an issue. TextBlob has another advantage – it has its classifier like Naïve Bayes which can help us to build our sentiment classification model [8]. However, the focus was more on the scoring of the sentiments which is why the TextBlob module was chosen to determine the sentiment scores of several texts.

5.3 Extraction of Issue Details

After the selection of the Sentiment Analysis tool, the next part was the extraction of issue details for all the 35 repositories. This concept was taken from Rath et al. [11] where they created a dataset from Git Commits and Jira Issues for seven open-source software projects. Using PyGitHub, it was connected to the below 35 repositories in Table 9

apache/spamassassin
apache/groovy-website
apache/cordova-plugin-globalization
apache/openwhisk-runtime-nodejs
apache/dubbo-samples
apache/apisix-website
apache/maven
apache/couchdb-fauxton
apache/submarine
apache/cordova-plugin-camera
apache/cordova-ios
apache/cordova-android
apache/mynewt-core
apache/kylin
apache/fineract
apache/drill
apache/bookkeeper
apache/iceberg
apache/iotdb
apache/couchdb
apache/dolphinscheduler
apache/cloudstack
apache/trafficcontrol
apache/geode
apache/skywalking

apache/tvm
apache/arrow
apache/pulsar
apache/shardingsphere
apache/echarts
apache/superset
apache/airflow
bitcoin/bitcoin
facebook/react-native
microsoft/vscode

Table 9: *List of repositories for the study*

and collected 26566 comments from all the closed issues associated with these repositories. After that, storing the data in MYSQL tables Issue and Issue_comment was performed. While fetching the issue details as well as comments, the comments that were made by the GitHub bot were ignored. It is because the thesis work is centered around human motivation and emotion. The Issue table has the following columns in Table 10.

Column Name	Description
Issue id	It is the issue number associated with an issue also the primary key of the table.
Repo_name	This is the repository name with the issue associated with it.
Commentid	This is the id associated with each comment for that issue. This is also the primary key of the table and is referred to by the Issue_comment table.
Title	Summary of the issue
Startdate	Start date of the issue i.e., when the issue is created.
Enddate	End date of the issue i.e., when the issue is closed.
Days_needed	It is a derived column i.e.; it is created after subtracting the start date from an end date to get the total time needed to close the issue.
Issuetype	It is the label of the issue. Generally, most of the time, it is observed that any issue in GitHub is tagged with a label e.g., Bug, Feature.
Assignee	Assignee denotes to whom the issue is assigned.

Table 10: *Issue table structure*

Table 11 contains the details of Issue_comment table.

Column Name	Description
Commentid	It is the id associated with each comment. It is the primary key as well as the foreign key of the Issue_comment table.
Comment	The actual text string is associated with an issue.

User	It is the commenter who is commenting on the issue.
Reactions	It is the string of emoticons like thumbs-up, thumbs-down, heart, and angry face that is associated with the issue comments.

Table 11: *Issue_comment table structure*

5.4 Extraction of Commit Details

Once the extraction of the issues and their related comments were extracted into Issue and Issue_comments tables, the next step was for commit details from all the 35 repositories mentioned above. Those commits which are in the “Success” state were considered. It is because the interest was more towards past commits which were complete. Even the PyGitHub module generally shows details of successful commits. There are a total of 6520 commits from all the repositories. The reason behind extracting commit was to find the emotion in the commit comments from the commit author. Also, it was done to see whether emotion or sentiment has any variation in commits or not. It was stemmed from the idea by Sinha et al. [23]. All 6520 commits from all the closed issues from the 35 repositories were stored in the “commit” table. Here are the details of the columns in the “commit” table in Table 12.

Column Name	Description
Commitid	This is the hash value of the commit, which is unique for every commit. It is the primary key of the table. Every issue has at least some commits associated with it in GitHub.
Comments	This column stores the comments associated with any commit. This can be null as in several cases author does not put any comments. There are cases where a bot in a repository makes automated comments which were not considered. It is because the thesis work is interested in human emotion and related sentiment.
Author	This column value indicates the person who is pushing the commit as well as commenting on the commit.
Date	The date column indicates the date on which the commit is made.
Files_list	This column describes the list of files that are changed during a specific commit.
No_line_changes	This column signifies the total number of line changes for a commit.
Issue	This is the issue number that has a reference to issueid in the Issue table. It denotes the commit associated with an issue.
Repo	This column signifies the repository name which contains the commit.

Table 12: *Commit table structure*

Here is the data model of the whole database structure in Fig. 13.

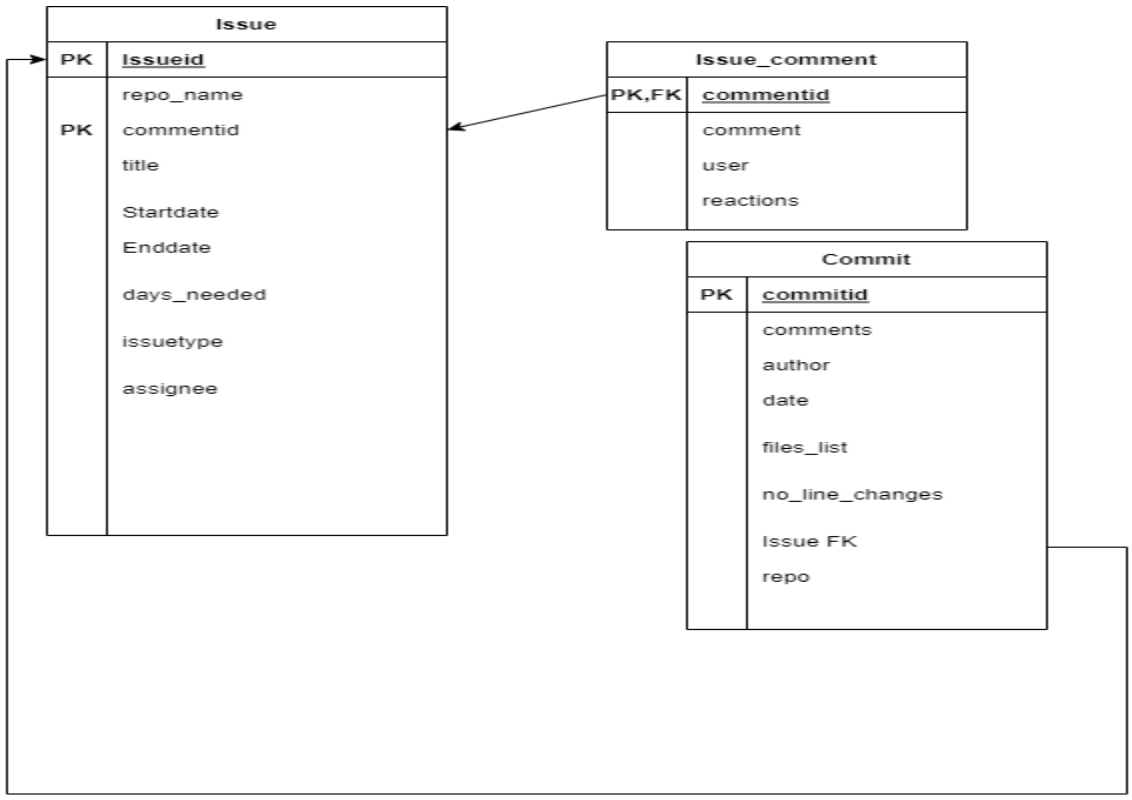

Figure 13: *The data model of the extraction process in Issue, Issue_comments, and Commit tables (Primary keys are shown in bold with letters PK and Foreign keys are shown with letters FK)*

In the next section, mapping of the motivators from the literature review with the extracted details from both Issue, Issue_comments, and Commit tables is shown.

5.5 Mapping of Factors and Metric Calculation

Once the extraction of issues and commits was done in the MYSQL tables, the next task was to map those data with the motivators and de-motivators that are listed by Beecham et al. [3]. Beecham et al. [3] listed out 22 motivators and 15 de-motivators that are part of the lives of Software engineers. In the table below, a listing of the motivators is shown that could be mapped with the data that was found from GitHub. The details in Table 13 were found after analyzing the stored data in the Issue and Commit tables. From Beecham et al. [3], technically challenging work was taken and mapped with the values from the “Files_list”, “no_line_changes” columns in the Commit table. As per the definitions for technically challenging work [3][14], it fits well with the complexity of the issue. In this case, “Files_list” and “no_line_changes” can be the probable factors to calculate the complexity of an issue. On the other hand, employee participation can be used as a contribution to a project [3]. In case

of issue, collaborators can be found from GitHub from the values of “Assignee”, “Commit Author”, and “User” fields in Issue, Issue_comment, and Commit table respectively. Similarly, the risk, which is a de-motivational factor, can be of different types in a different context. Here, in this thesis work, the easiest one to calculate was from the “days_needed” field from the Issue table in the context of project deadline [2].

Motivators/De-Motivators from Literature	Related datapoint from Issues	Related data point from Commits	Extrinsic/ Intrinsic
Technically challenging work/ The complexity of the task [3]	—	Files_list, no_line_changes	Intrinsic
Employee participation/ Involvement Or working with others [3]	Assignee, User	Author	Intrinsic
Risk [3] or Risk related to project delivery time [3]	Days_needed	---	Extrinsic

Table 13: *Mapping of motivators and de-motivators*

Next is the metric calculation process which shows how to derive the measurable motivators and de-motivators. The following assumptions were made to calculate each factor,

Complexity

Software complexity can be analyzed and measured by methods and models of task complexity [15]. However, Claes et al. [12] and Yahya et al. [16] described Lines of Code as one of the approaches to calculate the complexity of a Software in terms of lines of code. LOC or Lines of Code is one of the traditional and easiest methods to determine the size of the complexity of a Software.

Assumption

The assumption was made here concerning an issue. It means the complexity was calculated at the issue level. As there are the lines of code changes for an issue in its commits, it would measure the code complexity of the issue. It would give the size of the complexity after multiplication with several file changes in the commit for the issue. Thus, the size of the complexity was measured from GitHub using the below formula,

$C_i = n_L * n_F$ ..... *(3)*

Where C_i= Complexity of an issue at Code level

n_L = Number of lines of changes in the issue

n_F = Number of file changes in the issue

Risk

According to Raphael [2], the timeline or delivery time is one of the major risks in any Software Development project that is following an agile framework. This delay may occur due to less availability of resources, improper planning of the processes, too much customer involvement.

Assumption

Most of the project repositories that were used to extract data had Kanban like boards to track the releases in the project. Kanban is a very popular framework in agile-based projects. Therefore, it was assumed that issue completion time could be considered as a risk at the micro-level i.e., at the issue level. As most of the projects involve here uses the Kanban approach and from the literature review it was found that risk was one of the de-motivational factors in project delivery, issue completion time can be taken as risk. Hence, from GitHub "Risk" was calculated like below,

R_i = *High when* $C_t > Avg\ (C_t)$……. *(4)*

= *Low when* $C_t < Avg\ (C_t)$……. *(5)*

Where R_i = Risk of an issue, C_t = Completion time of an issue

Thus, Risk is based on the average completion time of an issue. If it is greater than the average completion time, there is a higher risk associated with it.

Collaboration

Beecham et al. [3] stated that Collaboration is one of the motivational factors that affect software engineers when they are working in a team. Based on the team-mates, a software engineer's motivation varies.

Assumption

Here, it was assumed those people as collaborators who were not only involved in developing the code but also commenting on the issue to guide the developers or other team members who were working with the assignee. A noticeable thing was observed in most of the cases in GitHub that some issues had assignee value blank, but others were working on the issue as a commenter only or developer-only or both. Even there were cases, where the assignee was there others were working as a developer by committing code changes or commenting to guide others. Thus, those people are actual collaborators who are in any of the roles of Assignee, Commenter or Commit Author. So, the Collaboration was calculated from GitHub like equation (6),

$C_S = C_L - M_L$ ……. (6)

Where C_S = Collaboration Score, C_L = Number of members in the Collaborators List, M_L = Number of members in the Mentions' list only. Mentions' list indicates the list of mentions of the developers in a comment. If the person is mentioned only but not working in any of the roles mentioned above, then we can subtract them from the collaborators' list.
Generally, by this equation (6), Strong collaboration is indicated by positive value and weak collaboration is indicated by a negative value. If there are more than two issues and if we see all have a positive score, in that case, we will consider higher positive value as stronger collaboration. Whereas lower positive value means weaker collaboration.

5.6 Survey Design

Whatever insights were collected using the data mining process, it was needed to verify it from the Developers who were part of Open-Source projects. Thus, it was decided to have a survey with the developers who worked on an issue as a developer as well as a commenter. Using this condition from 35 GitHub repositories, 139 Developers were found. To complete the process smoothly, the below steps were followed,

- Listing all the developers that were found via mining process and condition given above when Developer was also commenter i.e., the developer was committing the issue as well as commenting on the issue discussion. To get such developers' details, there is a table created with the name "user" in MYSQL which stores all the collaborators' usernames and corresponding email addresses.
- After that, a list of survey questionnaires was created related to the thesis work (refer to **Appendix A**)
- Once the list of questions was compiled, it was sent to the developers in Microsoft form format.
- Once all the responses were recorded, a qualitative inference had been made to prove the insights that were gathered from data mining.

6.Results and Analysis

After mining 26566 Issue comments with Issue details and 6520 related commits, a total of 1069 records were found where both the developer i.e., commit author and commenter were the same. So, the presented analysis is from the developer perspective viz. how do the results vary for developers for different kinds of motivators or de-motivators. From this thesis work, two motivators Complexity and Collaboration, and one de-motivator Risk were extracted. For simplification purposes, the issue types are categorized into five main labels or categories: Bug, Documentation, Enhancement, Feature, and Others. In the category or label “Others”, all types of issues that do not belong to any of the four categories mentioned, are considered. The calculation results for the motivator metrics are given in Tables 14 and 15.

	Issue_category	Average	Min	Max	Count
1	Others	6454	1	602616	556
2	Bug	1100	1	86520	144
3	Documentation	24840	1	3237570	288
4	Enhancement	3642	1	157200	77
5	Feature	204995	128450	281540	4

Table 14: *Average, Minimum, Maximum complexity per issue category with count*

	Issue_category	Average	Min	Max	Count
1	Others	257	0	8911	556
2	Bug	187	1	624	144
3	Documentation	179	0	919	288
4	Enhancement	216	1	711	77
5	Feature	288	137	438	4

Table 15: *Average, Minimum, Maximum risk related to completion time per issue category with count*

Once all the results were calculated using the metrics in equations (3),(4),(5), and (6), the dataset was broken into three complexity categories for each Issue category: High, Low, non-Complex. Similarly for risk categories: High, Low. In each motivator, categorization was done by comparing the value of each calculated factor and the average value of the factor in

the dataset for each issue category. For the risk factor, the unit of completion time is in hours.

First, it was started with the motivator Complexity. The following pie-charts give a detailed insight into when the emotion was extracted from commit comments and issue comments.

Emotion vs Complexity

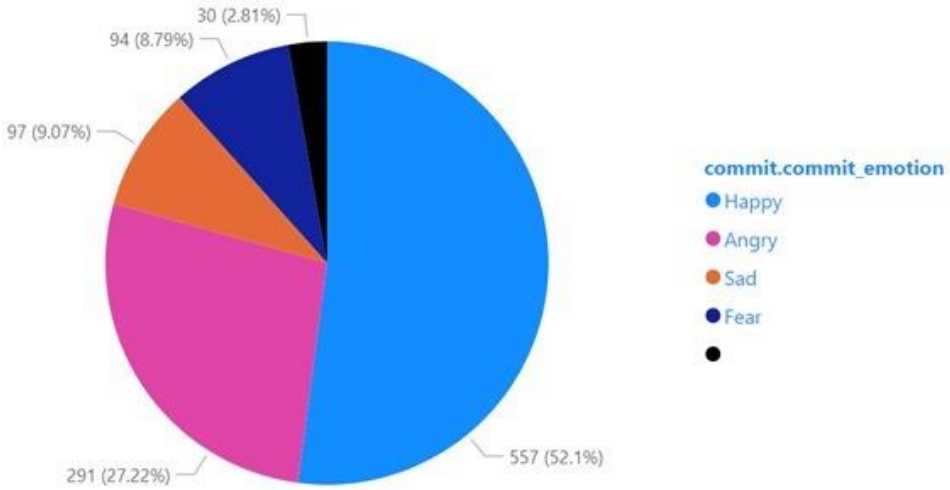

Figure 14: *Overall emotion for all issues via commit comments*

In Fig. 14, you can see that 52.1% of developers are showing happy emotion, followed by 27.22% of developers are showing angry emotion. To get into deep, the dataset was broken into three partitions- highly complex issues, low complex issues, and not complex Issues. Here are the emotion percentage for each scenario in Fig. 15 to Fig 17.

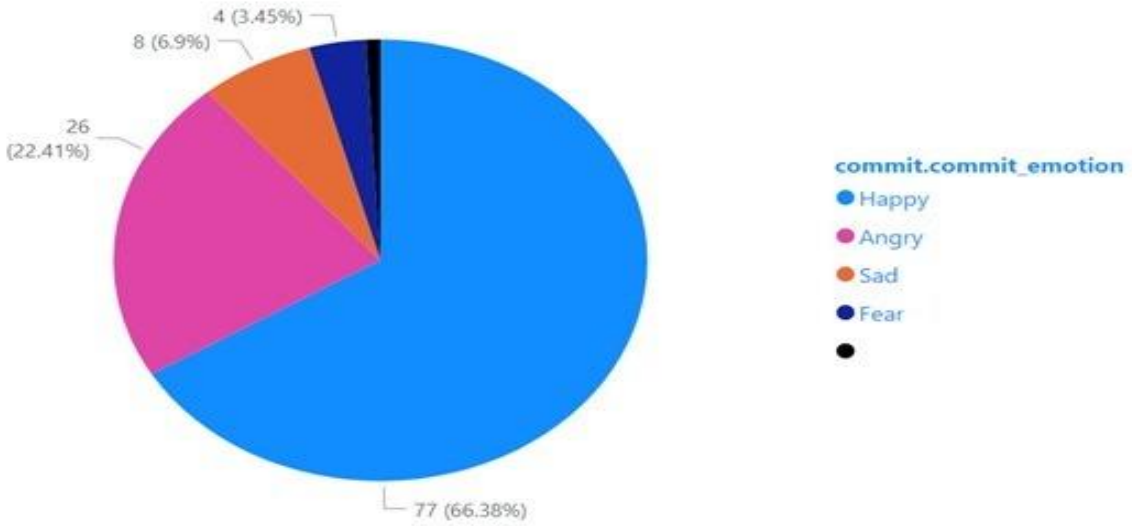

Figure 15: *Emotion percentage via commit comments for highly complex issues*

The above pie chart (Fig. 15) shows that the percentage of happy developers solving highly complex issues are quite big which is 66.38%. The next emotion is anger which is 22.41% of the developers.

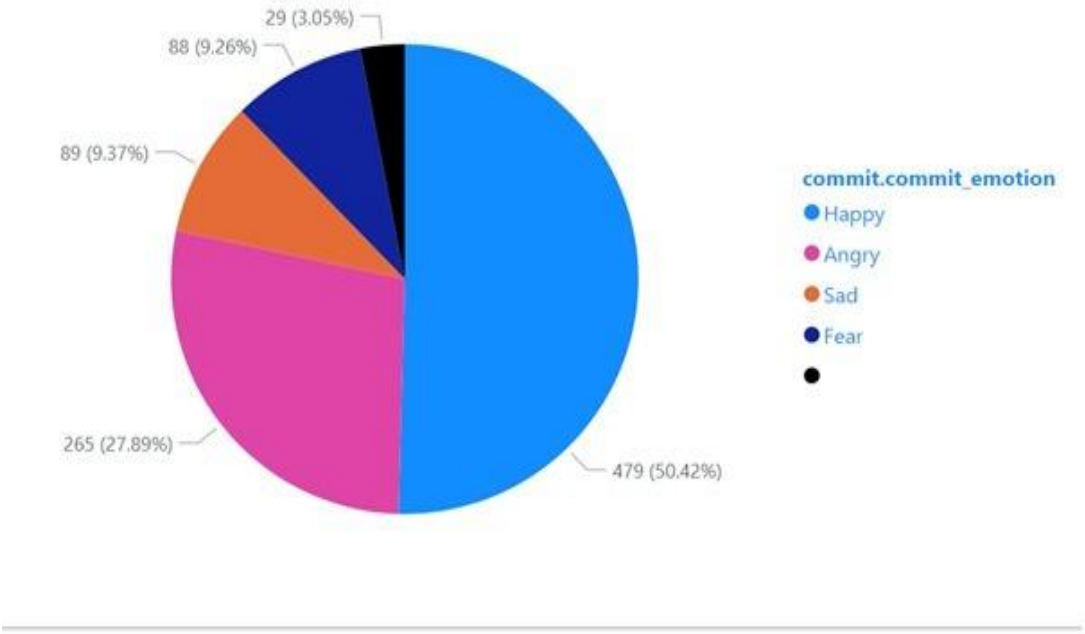

Figure 16: *Emotion percentage via commit comments for low-complex issues*

In Fig. 16, the happy developer percentage slashes down to 50.42% but the angry developers' percentage gets an increment to 27.89%. There were few comments where no emotion was detected as they were empty. This "no emotion" is indicated in Fig.16 by the black portion. Next, for not complex issues, pie-chart is Fig.17, the happy developers' percentage is lower than developers with fear.

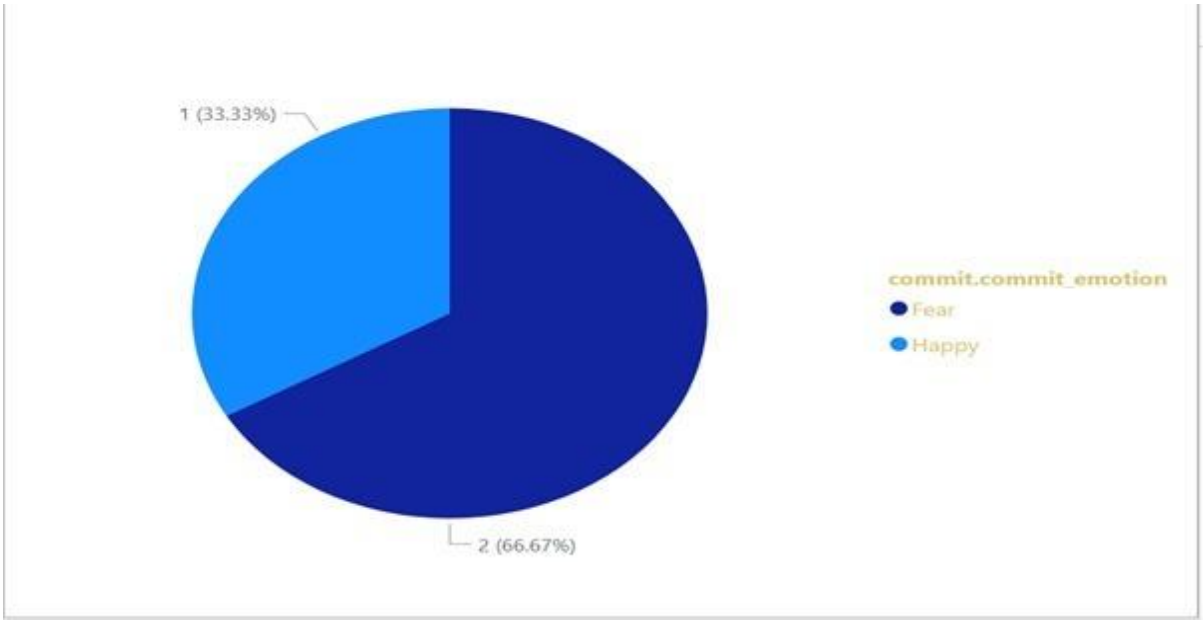

Figure 17: *Emotion percentage via commit comments for non-complex Issues*

In general, when the developer comments on an issue discussion, the overall emotion is like below in Fig. 18.

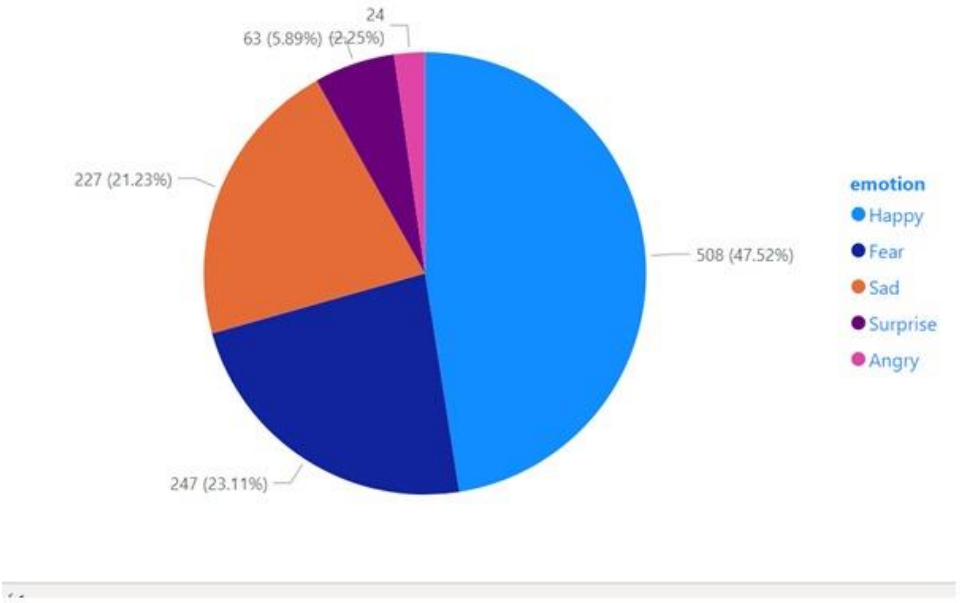

Figure 18: *Emotion percentage via issue comments for all issues*

In Fig.18, we can see 47.52% emotion is happy for developers when they involve in issue discussion. But, in detail, results give more insights. For highly complex issues,

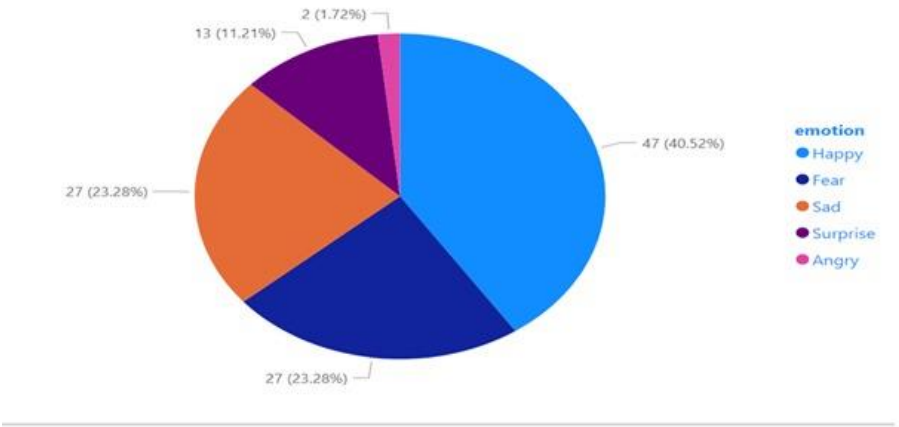

Figure 19: *Emotion percentage via issue comments for highly complex issues*

the happy emotion is dominant over other emotions (Fig. 19). However, in low-complex cases (Fig. 20), the developers with happy emotions are much higher than the highly complex issues.

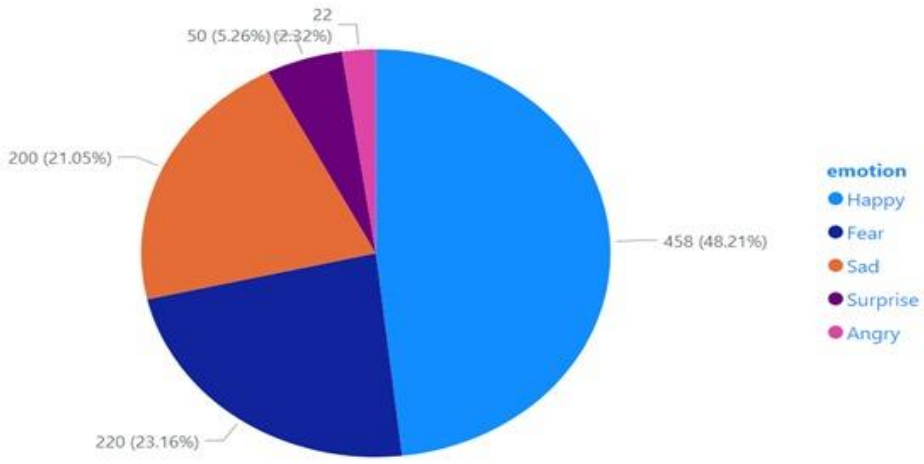

Figure 20: *Emotion percentage via issue comments for low-complex issues*

Emotion vs Risk (related to Deadline)

When emotion was studied with respect to risk, the following scenarios were considered.

Scenario 1: When the developers' emotion was taken from issue comments

49.43% of developers are happy with high-risk Issues (Fig. 21). Whereas 46.89% of developers are happy with low-risk issues (Fig. 22).

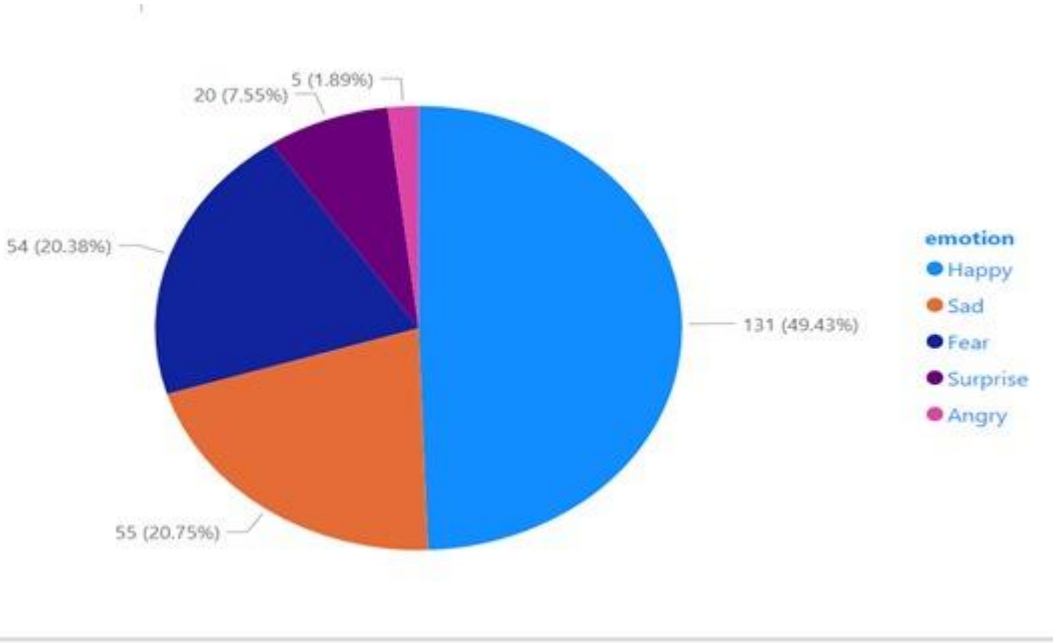

Figure 21: *Emotion percentage via issue comments for highly risk issues*

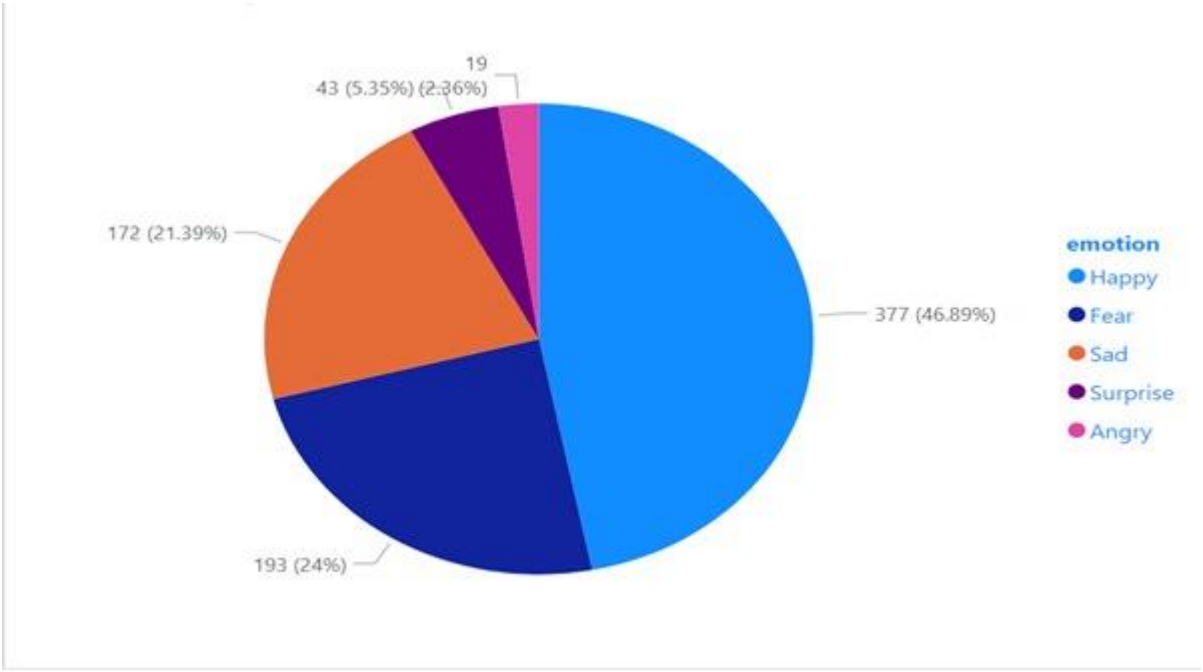

Figure 22: *Emotion percentage via issue comments for low-risk issues*

Scenario 2: When developers' emotion is taken from commit comments

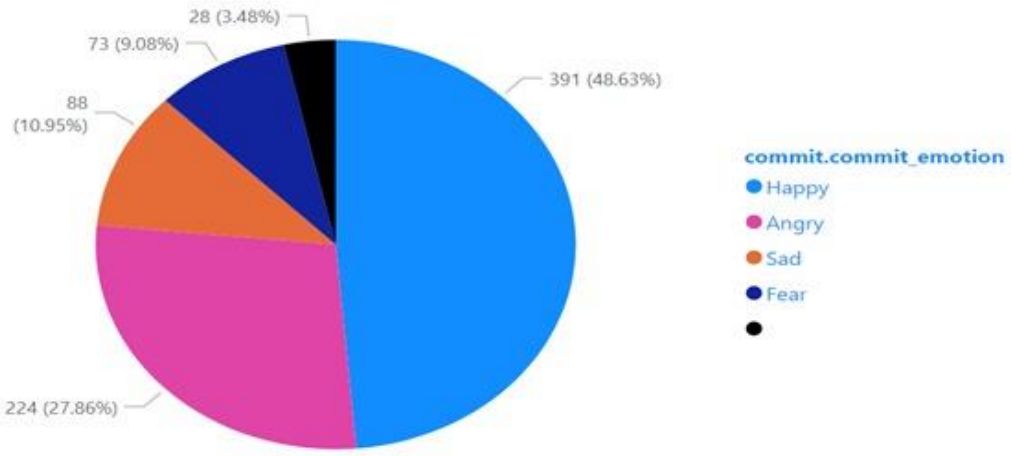

Figure 23: *Emotion percentage via commit comments for low-risk issues*

In this case, 48.63% are happy with low-risk Issues (Fig. 23) but 62.64% are happy with high-risk Issues (Fig.24). One observation is to note that few comments are not associated with any emotion because they are blank. It is because developers sometimes do not post any comments during the commit. This is indicated in Fig.23 and Fig.24 with black colour.

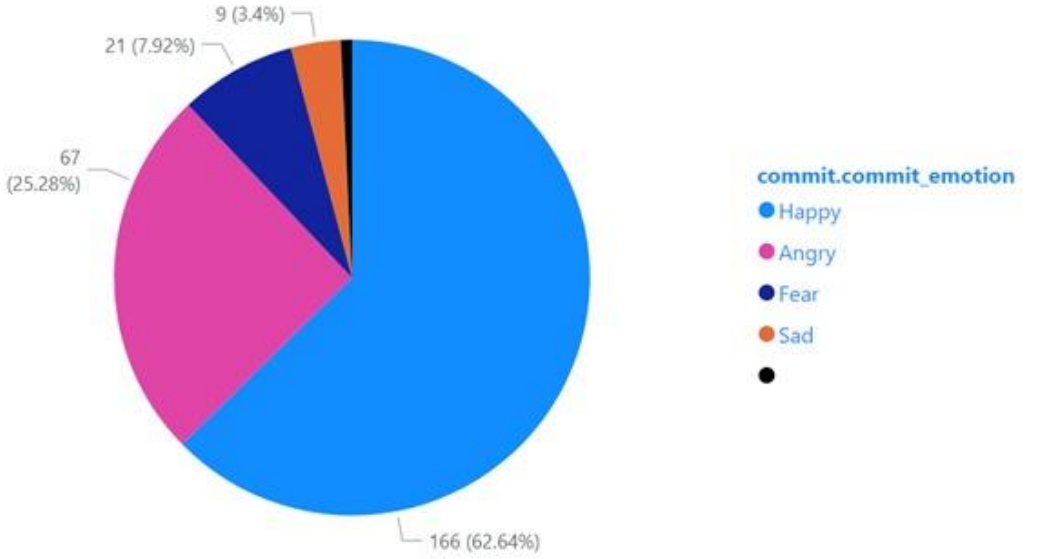

Figure 24: *Emotion percentage via commit comments for high-risk Issues*

Emotion and Sentiment vs Collaboration Score

Collaboration score is also an important aspect of motivation as discussed by Beecham et al. [3]. The following graphs show how emotion and sentiment vary with collaboration scores. From Fig.25, it is visible that emotions sad, fear, and surprise have a high value of median and average collaboration score. Whereas happy emotion has a low degree of median and average collaboration score. However, Fig. 26, which is plotted for sentiment against collaboration score has a different insight.

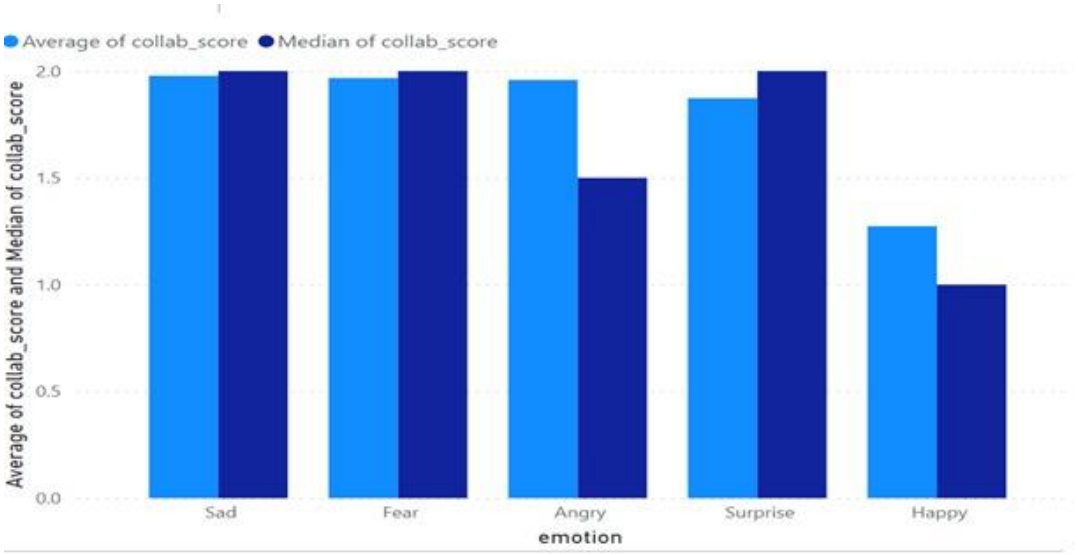

Figure 25: *Average and Median collaboration score for each emotion from issue comments*

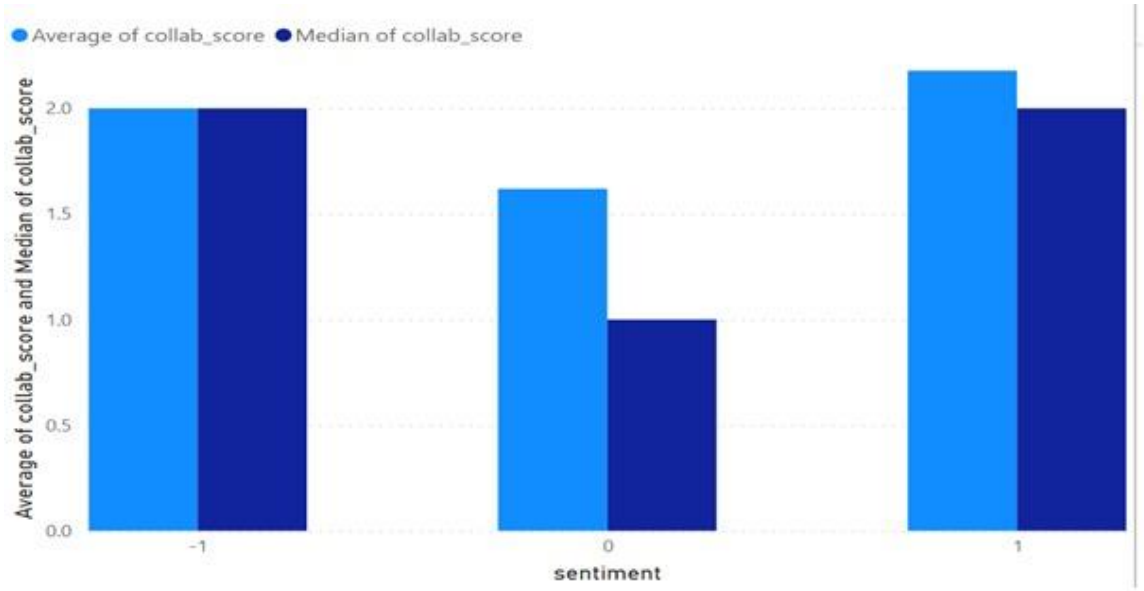

Figure 26: *Average and Median collaboration score for each sentiment from issue comments*

Fig.26 shows that when there is a negative sentiment, the average and median collaboration scores are high which is around a value of 2. But for neutral sentiment, the collaboration score is much lower. However, for the positive sentiment, the average collaboration score is higher than for the negative sentiment. Whereas the median score in the case of positive sentiment is lower than for the negative sentiment. When the sentiment and emotion are extracted from commit comments, the below charts (Fig.27,28) are displayed.

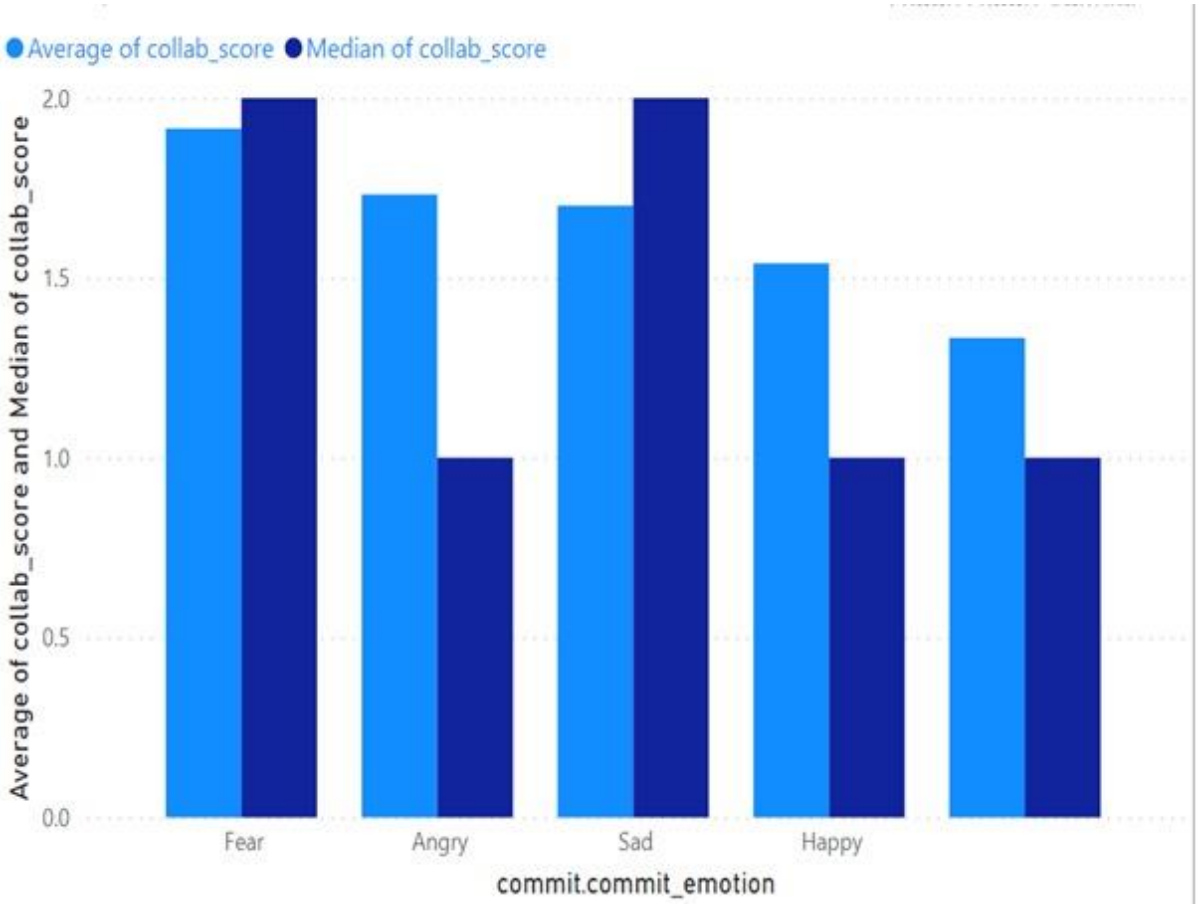

Figure 27: *Average and Median collaboration score for each emotion from commit comments*

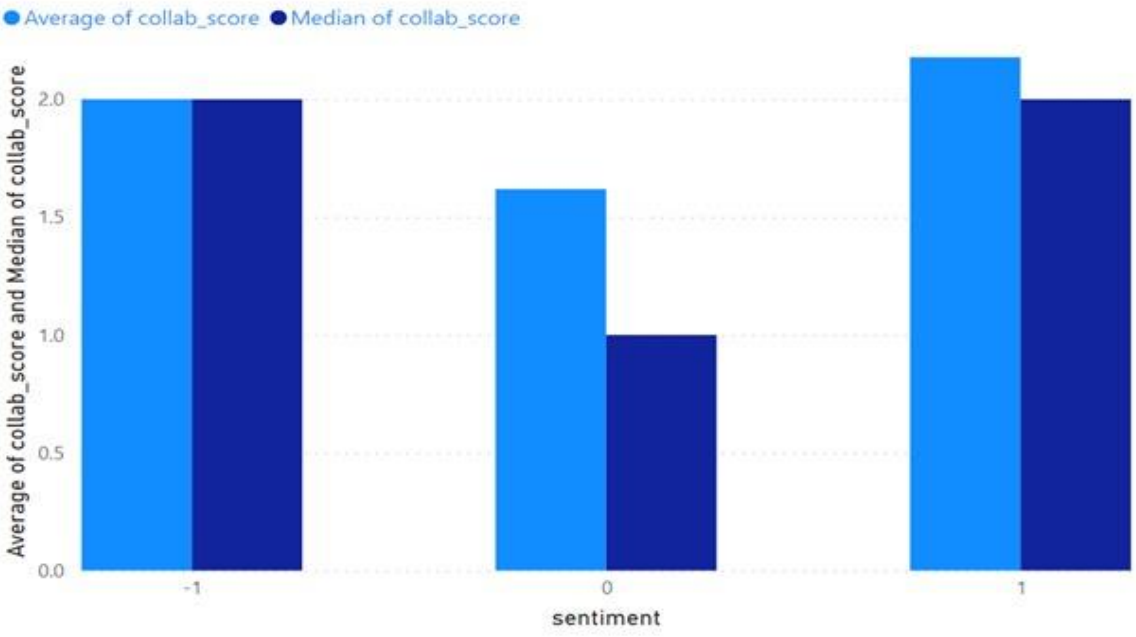

Figure 28: *Average and Median collaboration score for each sentiment from commit comments*

In the next section, the results are shown after performing a survey with six active developers from four different open-source projects hosted in GitHub. In the survey, direct questions related to any emotion were not asked. Instead of that, questions regarding satisfaction and accomplishment were framed. It is because happiness, joy, or sadness may have a different meaning for different developers. In the survey, it was assumed that positively motivated or satisfied developers are synonymous with happy developers.

For the survey part, the following questions were asked, and the below responses were received (Tables 16 to 18). The survey was done with only six developers from China, Brazil, Poland, Germany, and India.

Responses on years of experience in programming, open-source projects, and industry

Parameters	Less than 1 year	1-3 Years	3-5 Years	More than 5 Years
Programming Experience	0(0%)	1(17%)	2(33%)	3(50%)
Open-Source Software Project Experience	2(33%)	2(33%)	2(33%)	0(0%)
Industry experience	0(0%)	1(17%)	2(33%)	3(50%)

Table 16: *Survey responses on industry, programming, and OSS projects*

Responses on employment in software industry

Parameters	Yes	No
Employed in Software Industry?	5(83%)	1(17%)
Full-Time employment?	5(83%)	0(0%) **

NB:** No response was given

Table 17: *Survey responses on employment in software industry*

Responses on factors related to motivation

Parameters	Yes	No
Satisfaction in solving complex tasks (number of files and line changes)	6(100%)	0(0%)
Positive effect on collaboration in issues	6(100%)	0(0%)
Accomplishment on solving issues where risk is there with a deadline	6(100%)	0(0%)

Table 18: *Survey response on factors related to motivation*

Based on the performed research methodologies and their related results, the answers to the research questions can be given as follows,

RQ1. What are the measurable motivators and de-motivators that can be derived from GitHub?

After studying several kinds of literature and articles related to motivation mining especially Beecham et al. [3] and Sharp et al. [14], there are 22 motivators and 15 de-motivators found. After analysing the stored data in the database tables Issue, Issue_comments, and commit, it is deduced that on an issue-level to find the motivational factors or de-motivational factors, we must think of the project-level parameters on a smaller scale viz. on an issue level. The different fields like “days_needed” in table Issue can help in measuring the risk factor related to issue completion time. Similarly, fields like “no_line_changes”,” File_list” in the commit table help in measuring the complexity of the issue. Then the fields “assignee” in the Issue table, “user” in the Issue_comment table, and “author” in the commit table can help us to get the strength of the collaboration. The same mapping is mentioned in section 5.5.

RQ2. How to derive the measurable motivators and de-motivators from GitHub?

In the section 5.5, the different metric calculation methods are mentioned to derive the measurable motivators and de-motivators. To measure the risk related to the issue deadline, we need to consider the “Days_needed” field of the Issue table as it signifies the completion time of an issue. Then, we need to categorize the type of issues in major categories and then calculate the average, maximum, and minimum of the “Days_needed” value for each category. To deduce whether an issue is risky or not, we need to run a check whether the completion time for the issue is more or less than the average completion time for that issue category. Now to derive the complexity of the issue we have “no_line_changes”, “File_list” fields with us. As per the line of codes concept, the number of lines gives the basic complex-

ity of the code. It is way inferior to cyclomatic complexity in terms of calculating the complexity of a code [71], but it is easier to measure from the GitHub data. To calculate the complexity of the overall issue, we need to multiply the value of “no_line_changes” and “File_list”. To derive the collaboration, we need to deal with the collaboration score. It is nothing but the count of collaborators in the issue minus the number of mentions only. The collaborators can be commenters, commit authors or assignees associated with the specific issue.

RQ3. How are these measurable motivators or de-motivators influenced by emotion and sentiment?

In the survey response, 50% of the developers have more than 5 years of programming experience. In addition to this, 50% of the developers have been working in the industry for more than 5 years. Among the surveyed developers, 100% of them have positively responded by saying they are motivated to work on those issues that have risks of crossing deadline or deadline has been crossed. Another observation is 100% of the developers are motivated or happy to work on those issues which have complexity related to a huge number of files and lines of changes. Even all the developers have responded that collaboration has a positive effect on their work. Moreover, they have responded that they feel motivated or happy when there is a good collaboration. Now, if we compare the result with the data mined from GitHub, we can see for highly complex issues and highly risky issues related to the deadline, developers show happy emotion in most cases (68.38% for highly complex issues, 49.38% for high-risk issues). In the case of Collaboration, the scenario is a bit different, for happy developers the collaboration score is lower than sad and angry developers.

There are cases of differences in the results between the survey and the data mining. The reason behind this might be the small dataset from the mining and the much smaller dataset in the survey. If the dataset was larger, the results could have taken a different turn. Another difference that is found from this study – risk may not be a de-motivator for all developers. In the systematic literature review by Beecham et al. [3], risk has been mentioned as a de-motivator which is contradicting the results. However, a larger dataset may clear this contradiction.

7.Conclusion

This thesis outlines a method for determining what motivates and demotivates software developers working on open-source projects. GitHub is chosen as the open-source repository for this thesis study in order to obtain a variety of data connected to software development and software developers. It is demonstrated that three factors can be obtained and assessed from GitHub using the above-mentioned research approaches. Risk, complexity, and collaboration score (aka collaboration) are the three elements.

The MOCC Model, PyGitHub API, TextBlob library, Text2emotion library, and MySQL database were used to build the entire framework for this thesis. These five elements were crucial in connecting GitHub to extracting data related to issues and their contributions for this empirical study, and they helped develop the framework to deduce the motivators and demotivators. The data is small at this point, but it may be ramped up to acquire more insights from GitHub. We might be able to identify more intriguing motivators or de-motivators from GitHub using the same approach.

In terms of attaining the actual goal, the entire thesis labor is minor. However, it serves as a starting point for determining a software developer's motivational variables. It can be useful not only for open-source software development initiatives, but also for international corporations where software engineers face a variety of challenges on a daily basis. This thesis work can serve as a framework for multinational corporations that care about their software engineers and want to keep them by establishing a variety of rules once they've identified the generic motivational factors that apply to all software engineers.

Bibliography

[1] “Affective Events Theory.", https://en.wikipedia.org/wiki/Affective_events_theory.

[2] https://www.gratasoftware.com/common-risks-agile-projects-deal/

[3] Beecham, S., Baddoo, N., Hall, T., Robinson, H., & Sharp, H. (2008). Motivation in Software Engineering: A systematic literature review. Information and Software Technology, 50(9–10), 860–878. https://doi.org/10.1016/j.infsof.2007.09.004.

[4] Calefato, F., Lanubile, F., Maiorano, F., & Novielli, N. (2018). Sentiment polarity detection for software development. Empirical Software Engineering, 23(3), 1352–1382. https://doi.org/10.1007/s10664-017-9546-9.

[5] França, A. C. C., & da Silva, F. Q. B. (2010). Designing motivation strategies for software engineering teams: An empirical study. Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering - CHASE ’10, 84–91. https://doi.org/10.1145/1833310.1833324.

[6] Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in Open Source projects: An Internet-based survey of contributors to the Linux kernel. Research Policy, 32(7), 1159–1177. https://doi.org/10.1016/S0048-7333(03)00047-7.

[7] Murgia, A., Tourani, P., Adams, B., & Ortu, M. (2014). Do developers feel emotions? An exploratory analysis of emotions in software artifacts. *Proceedings of the 11th Working Conference on Mining Software Repositories - MSR 2014*, 262–271. https://doi.org/10.1145/2597073.2597086.

[8] Graziotin, D., Wang, X., & Abrahamsson, P. (2013). Are happy developers more productive? In J. Heidrich, M. Oivo, A. Jedlitschka, & M. T. Baldassarre (Eds.), Product-Focused Software Process Improvement (Vol. 7983, pp. 50–64). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-39259-7_7.

[9] Ortu, M., Murgia, A., Destefanis, G., Tourani, P., Tonelli, R., Marchesi, M., & Adams, B. (2016). The emotional side of software developers in JIRA. *Proceedings of the 13th International Conference on Mining Software Repositories*, 480–483. https://doi.org/10.1145/2901739.2903505.

[10] Rasch, R. H., & Tosi, H. L. (1992). Factors affecting software developers’ performance: An integrated approach. *MIS Quarterly*, *16*(3), 395. https://doi.org/10.2307/249535.

[11] Rath, M., Rempel, P., & Mader, P. (2017). The ilmseven dataset. 2017 IEEE 25th Interna tional Requirements Engineering Conference (RE), 516–519. https://doi.org/10.1109/RE.2017.18

[12] Sandros, Claes and Sofia nystedt. “Software Complexity and Project Performance.” (1999).

[13] Shahri, A., Hosseini, M., Almaliki, M., Phalp, K., Taylor, J., & Ali, R. (2016). Engineering software-based motivation: A persona-based approach. 2016 IEEE Tenth International Conference on Research Challenges in Information Science (RCIS), 1–12. https://doi.org/10.1109/RCIS.2016.7549312.

[14] Sharp, H., Baddoo, N., Beecham, S., Hall, T., & Robinson, H. (2009). Models of motivation in software engineering. Information and Software Technology, 51(1), 219–233. https://doi.org/10.1016/j.infsof.2008.05.009

[15] Tran, De, Ghislain Lévesque, and Jean Guy Meunier. (2004). *Software Functional Complexity Measurement with the Task Complexity Approach*.

[16] Yahya Tashtoush, Mohammed Al-Maolegi, and Bassam Arkok. (2014). "The Correlation among Software Complexity Metrics with Case Study." *International Journal of Advanced Computer Research* 4 (2): 414. https://search.proquest.com/docview/1613205253.

[17] https://www.statista.com/outlook/tmo/software/worldwide#:~:text=Revenue%20in%20the%20Software%20market,US%24823%2C706m%20by%202026

[18] https://www.statista.com/statistics/1126677/it-employment-worldwide/

[19] https://www.psychologytoday.com/us/basics/motivation

[20] https://www.moneycontrol.com/news/business/cognizant-attrition-hits-record-high-of-31-company-to-make-100000-lateral-hires-this-year-7244591.html

[21] https://www.glassdoor.com/Award/Best-Places-to-Work-LST_KQ0,19.htm

[22] McConnell, S. (1998). Problem programmers. IEEE Software, 15(2), 128. https://doi.org/10.1109/52.663801.

[23] Sinha, V., Lazar, A., & Sharif, B. (2016). Analyzing developer sentiment in commit logs. Proceedings of the 13th International Conference on Mining Software Repositories, 520–523. https://doi.org/10.1145/2901739.2903501

[24] "Motivation in Project Management." . https://eight2late.word-press.com/2008/08/29/motivation-in-project-management/.

[25] https://www.managementstudyguide.com/what_is_motivation.htm

[26] https://www.masterclass.com/articles/what-is-intrinsic-motivation-understanding-the-definition-of-intrinsic-motivation-and-how-to-use-intrinsic-motivation#what-is-intrinsic-motiva-tion

[27] Locke, Edwin A., and Gary P. Latham. "Building a Practically Useful Theory of Goal Setting and Task Motivation." The American psychologist 57.9 (2002): 705-17. Cross-Ref. Web.

[28] https://www.managementstudyguide.com/goal-setting-theory-motivation.htm

[29] https://www.ambitionbox.com/reviews/amazon-reviews

[30] Montana, Patrick J., and Bruce H. Charnov. "Management." ISBN 978-0-7641-3931-4.4th edition (2008) Web.

[31] https://en.wikipedia.org/wiki/Expectancy_theory

[32] https://www.managementstudyguide.com/expectancy-theory-motivation.htm

[33] Isaac, Robert G., Wilfred J. Zerbe, and Douglas C. Pitt. "Leadership and Motivation: The Effective Application of Expectancy Theory." Journal of managerial issues 13.2 (2001): 212-26. ABI/INFORM Global (Corporate). Web.

[34] https://www.healthline.com/health/extrinsic-motivation

[35] https://www.cleverism.com/job-characteristics-model/

[36] https://www.ckju.net/en/dossier/job-characteristics-model-what-it-is-and-why-it-mat-ters-more-ever

[37] https://biznewske.com/hackman-and-oldham-job-characteristics-model/

[38] http://www.geocities.ws/frtzw906/hackmanoldham.htm

[39] Cropanzano, Russell. *Affective Events Theory: A Theoretical Discussion of the Structure, Cause and Consequences of Affective Experiences at Work WORK-FAMILY CROSSOVER: A META-ANALYTIC REVIEW View Project*., (2017). Web.

[40] http://psychology.iresearchnet.com/industrial-organizational-psychology/job-satisfaction/affective-events-theory/

[41] https://dictionary.cambridge.org/dictionary/english/sentiment

[42] https://www.collinsdictionary.com/dictionary/english/sentiment

[43] http://sentistrength.wlv.ac.uk/

[44] https://textblob.readthedocs.io/en/dev/

[45] https://www.softkraft.co/python-nlp-libraries-features-us-cases-pros-and-cons/

[46] https://github.com/cjhutto/vaderSentiment

[47] https://www.analyticsvidhya.com/blog/2019/02/flair-nlp-library-python/

[48] https://github.com/flairNLP/flair

[49] https://github.com/flairNLP/flair/releases

[50] https://www.merriam-webster.com/dictionary/emotion

[51] https://www.britannica.com/science/emotion

[52] https://dictionary.apa.org/emotion

[53] Shaver, P., Schwartz, J., Kirson, D., & O'Connor, C. (1987). Emotion knowledge: Further exploration of a prototype approach. Journal of Personality and Social Psychology, 52(6), 1061–1086. https://doi.org/10.1037/0022-3514.52.6.1061.

[54] https://shivamsharma26.github.io/text2emotion/

[55] https://pypi.org/project/text2emotion/

[56] https://en.wikipedia.org/wiki/Open-source_software_development

[57] Ming-Wei Wu & Ying-Dar Lin. (2001). Open source software development: An overview. *Computer*, *34*(6), 33–38. https://doi.org/10.1109/2.928619

[58] https://en.wikipedia.org/wiki/GitHub

[59] Haddad, Ibrahim and Brian Warner. (2011). *» The Linux Foundation Understanding the Open-Source Development Model*.

[60] https://docs.github.com/en/issues/using-labels-and-milestones-to-track-work/managing-labels

[61] https://pygithub.readthedocs.io/en/latest/introduction.html

[62] Ortu, Marco, Bram Adams, Giuseppe Destefanis, Parastou Tourani, Michele Marchesi, and Roberto Tonelli. (2015). *Are Bullies More Productive? Empirical Study of Effectiveness Vs. Issue Fixing Time*. Piscataway: The Institute of Electrical and Electronics Engineers, Inc. (IEEE).

[63] https://www.indeed.com/career-advice/career-development/goal-setting-theory

[64] https://www.ckju.net/en/dossier/job-characteristics-model-what-it-is-and-why-it-matters-more-ever

[65] Sánchez-Gordón, M., & Colomo-Palacios, R. (2019). Taking the emotional pulse of software engineering—A systematic literature review of empirical studies. *Information and Software Technology*, *115*, 23–43. https://doi.org/10.1016/j.infsof.2019.08.002.

[66] http://happinessinternational.org/what-is-happiness/

[67] https://www.merriam-webster.com/dictionary/disgust

[68] https://www.ambitionbox.com/reviews/wipro-reviews

[69] https://www.ambitionbox.com/reviews/accenture-reviews

[70] Hasan, Jahid. Employee Performance Motivation a Systematic Literature Review with Linear Regression Analysis.

[71] https://www.geeksforgeeks.org/cyclomatic-complexity/

[72] Novielli, N., & Serebrenik, A. (2019). Sentiment and Emotion in Software Engineering. IEEE Software, 36(5), 6–23. https://doi.org/10.1109/MS.2019.2924013

[73] https://collab-uniba.github.io/EMTk/

[74] Van Eerde, W. & Thierry, H. (1996). Vroom's Expectancy Models and Work-Related Criteria. *Journal of Applied Psychology, 81* (5), 575-586.

[75] Vroom, V. H. (1964). Work and motivation. Wiley.

[76] Ryan, R. M., & Deci, E. L. (2000). Intrinsic and extrinsic motivations: Classic definitions and new directions. Contemporary Educational Psychology, 25(1), 54–67. https://doi.org/10.1006/ceps.1999.1020

[77] https://dzone.com/articles/introduction-to-word-vectors

[78] https://www.geeksforgeeks.org/explanation-of-bert-model-nlp/

[79] https://nlp.stanford.edu/projects/glove/

[80] https://www.geeksforgeeks.org/supervised-unsupervised-learning/

APPENDIX A: Survey Questionnaire

- How many years of experience do you have in programming? (E.g. School projects, Open-Source projects, Industry experience)
- How many years of experience do you have in Open-Source Projects?
- How many Open-Source projects are you currently involved in?
- How many years of experience do you have in industrial software development projects?
- Are you currently employed in the software industry?
- Are you employed full-time or part-time?
- Does your work include programming?
- Do you typically feel a sense of accomplishment after resolving an issue that took longer than expected (issue was resolved past deadline)?
- Do you consider other developers collaborating on an issue (by commenting on the issue or contributing to the code commit) has a positive effect on solving the issue?
- Do you typically feel a sense of accomplishment when you have completed a complicated issue (requiring changes in many code lines and/or in numerous files)?
- Do you feel a particular sense of accomplishment after completing more difficult tasks?

APPENDIX B: Definitions

Data mining

Data mining which is also known as knowledge discovery in data or KDD, is the technique to uncover patterns and other valuable information from the large data volume. It gives insights into different associations in a large data set amongst different variables, also useful as input for the different machine learning algorithms.

Word vector

Word vectors are a huge step forward in terms of analysing relationships between words, phrases, and documents. As a result, it can progress technology by giving machines a lot more information about words than was previously available with traditional word representations [77]. Technology such as speech recognition and machine translation are made possible by the word vector [77].

BERT

BERT is the abbreviated form of Bidirectional Encoder Representations from Transformers. It is a natural language processing model which was proposed by the researchers at Google Research in 2018 [78]. BERT has a state-of-the-art accuracy on various NLP and NLU tasks such as:

- Understanding of general language
- Stanford Q/A dataset for the SQuAD v1.1 and v2.0 Processes
- Situation With Adversarial Generations or in other words generative adversarial network

GloVe

GloVe is the abbreviated form of Global Vectors for Word Representation. It is an unsupervised learning algorithm for obtaining the vector representation of the words [79]. Training is performed on the aggregated version of global word co-occurrence statistics from a corpus, resulting in the showcasing of various interesting linear structures or substructures of the word vector space [79].

Supervised Learning

As the name implies, supervised learning takes place in the presence of a supervisor or teacher [80]. Basically, in supervised learning, we teach or train the machine aka machine learning model using data that are well labelled. This means some data is already tagged with the correct answer or tag. After that, the machine is provided with a new set of data so that the supervised learning algorithm analyses the training data and produces a correct outcome from labelled data which is nothing but the prediction [80].