MONASH University

On the Presence of Human Values in Software Development Artefacts: An Evaluation of GitHub's Issue Discussions

Arif Nurwidyantoro
Doctor of Philosophy

Main Supervisor
Prof. Jon Whittle

Co-Supervisors
Dr. Waqar Hussain
Dr. Mojtaba Shahin
Prof. Michel Chaudron (TU Eindhoven)

A Thesis Submitted for the Degree of Doctor of Philosophy at
Monash University in 2022
Faculty of Information Technology

Abstract

As software becomes ubiquitous and integrated into our lives, any violation of human values in software, such as social injustice or privacy breaches, raises public attention. Media have highlighted a few cases where software with a history of breaching people's values struggles to receive users' acceptance and becomes an object of public scrutiny. To avoid these circumstances, software practitioners need to consider (i.e. *be attentive to the implication of*) human values during software development.

The abstract and subjective nature of human values makes it difficult to integrate them in software. Previous studies on human values in software primarily focused on specific development phases, such as requirements and design phases. This thesis addresses this gap by conducting three empirical studies and developing a human values dashboard:

First, an empirical study was conducted to understand if human values are evident in issue discussions. In software engineering research so far, issue discussions have been considered an artefact used in the late stages of software development. This empirical study has proved that human values are present in issue discussions. This study found 20 themes associated with values (or *value themes*) in the issue discussions studied and proposed a description for each in the software engineering context.

Second, a set of classification methods were developed to automate the identification of human values in issue discussions. These methods were developed to address the obvious time and effort challenges when manual approaches are applied to identify and classify values from large amounts of textual data. The experiments' results showed that, among the methods evaluated, Multilayer Perceptron offers the best performance with a Precision score of 0.582, a Recall score of 0.741, an F1 score of 0.650 and an MCC score of 0.445.

Third, a human values dashboard was developed to support practitioners in the consideration of human values in software development. An exploratory study that collected data through interviews with software practitioners was conducted to gather potential benefits and requirements for the dashboard. Then, a human values dashboard was developed using the requirements from the exploratory study. Finally, a feedback study with software practitioners was conducted to confirm the dashboard's usefulness and to gather feedback for improving it.

The main contributions of this thesis are:

1. providing empirical evidence that human values are present in issue discussions;
2. finding value themes and proposing descriptions for them in the software engineering context;
3. developing and evaluating classification techniques to automate the detection of values in issue discussions; and
4. developing and evaluating a dashboard as a tool to support the consideration of human values in software development.

Declaration

This thesis is an original work of my research and contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Signature: ________________________________

Print Name: Arif Nurwidyantoro

Date: July 2022

Publications during Enrolment

Publications as a result of this thesis:

1. **Arif Nurwidyantoro**, Mojtaba Shahin, Michel Chaudron, Waqar Hussain, Rifat Shams, Harsha Perera, Gillian Oliver, Jon Whittle, *Human Values in Software Development Artefacts: A Case Study on Issue Discussions in Three Android Applications*, Information and Software Technology Journal: Special Issue on Waste and Value in Software Engineering, Vol. 141, January 2022
2. **Arif Nurwidyantoro**, Mojtaba Shahin, Michel Chaudron, Waqar Hussain, Harsha Perera, Rifat Shams, Jon Whittle, *Towards a Human Values Dashboard for Software Development*, Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM), October 2021

Other publications related to this thesis:

1. Waqar Hussain, Mojtaba Shahin, Rashina Hoda, Jon Whittle, Harsha Perera, **Arif Nurwidyantoro**, Rifat Shams, Gillian Oliver, *How Can Human Values Be Addressed in Agile Methods? A Case Study on SAFe*, accepted to be published in IEEE Transactions on Software Engineering, 2021
2. Rifat Shams, Mojtaba Shahin, Gillian Oliver, Waqar Hussain, Harsha Perera, **Arif Nurwidyantoro**, Jon Whittle, *Measuring Bangladeshi female farmers' values for agriculture mobile applications development*, in Proceedings of the 54th Annual Hawaii International Conference on System Sciences, 2021
3. Waqar Hussain, Harsha Perera, Jon Whittle, **Arif Nurwidyantoro**, Rashina Hoda, Rifat Shams, Gillian Oliver, *Human Values in Software Engineering: Contrasting Case Studies of Practice*, IEEE Transactions on Software Engineering, 2020
4. Rifat Shams, Waqar Hussain, Gillian Oliver, **Arif Nurwidyantoro**, Harsha Perera, Jon Whittle, *Society-oriented Applications Development: Investigating Users' Values from Bangladeshi Agriculture Mobile Applications*, in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society, 2020
5. Harsha Perera, Waqar Hussain, Jon Whittle, **Arif Nurwidyantoro**, Davoud Mougouei, Rifat Shams, Gillian Oliver, *A Study on the Prevalence of Human Values in Software Engineering Publications, 2015 – 2018*, in Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering, 2020

6. Harsha Perera, Gunter Mussbacher, Waqar Hussain, Rifat Shams, **Arif Nurwidyantoro**, Jon Whittle, *Continual Human Value Analysis in Software Development: A Goal Model based Approach*, in Proceedings of the IEEE 28th International Requirements Engineering Conference, 2020

7. Harsha Perera, Waqar Hussain, Davoud Mougouei, Rifat Shams, **Arif Nurwidyantoro**, Jon Whittle, *Towards Integrating Human Values into Software: Mapping Principles and Rights of GDPR to Values*, in Proceedings of the IEEE 27th International Requirements Engineering Conference, 2019

Acknowledgements

Praise to Allah the most glorified and the most high, who has showered His blessing and strength to me, so that I have been finally able to accomplish this thesis.

This research was supported by a Monash University Graduate Scholarship (MGS) and a Monash University Graduate Research Completion Award (GRCA). This thesis utilised a proofreader service (getproofed.com.au) to provide suggestions solely on the English expression and grammatical aspects of this thesis without affecting the substance and structure of the thesis.

First and foremost, I would like to sincerely express my gratitude to Professor Jon Whittle and Professor Michel Chaudron for the opportunity to pursue a PhD at Monash University. I would also like to appreciate them, along with Dr. Mojtaba Shahin and Dr. Waqar Hussain, for their guidance, advise, and continuous support throughout my PhD.

I also want to sincerely thank my research collaborators in the OVIS team: Associate Professor Gillian Oliver, Associate Professor Rashina Hoda, Harsha Perera, Rifat Ara Shams, and Dr. Davoud Mougouei (alumni), who offered a great support in this research work. They have been such great advisors and motivators.

Furthermore, I should also acknowledge all valuable feedback and insightful comments from my panel members, Professor John Grundy, Professor Burak Turhan, Associate Professor Ron Steinfeld, Dr. Xin Xia, and Dr. Li Li. I also acknowledge all valuable comments and feedback from my examiners, Dr. Kelly Blincoe and Dr. Christoph Treude. Their suggestions, encouragement, and support were very valuable to scope this thesis for timely completion.

I also would like to acknowledge the Faculty of IT, Monash University, for providing valuable workshops and seminars, and extending my scholarship. Furthermore, I want to express my gratitude to the Graduate Research Team of the Faculty of IT, for all suggestions and assistance throughout my candidature, especially during the pandemic situation.

Last but not least, I would like to thank my family, especially my parents, for their neverending spiritual and mental supports. My deepest gratitude goes to my lovely wife, Fadhilatul Muharam, who wonderfully always by my side through the difficult times. I also would like to thank my two wonderful little sons, Taqy and Arham, for their laugh and understanding throughout my PhD.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter describes the motivation, research objective, and research questions for this thesis. The latter sections of this chapter describe the contributions and the outline of this thesis.

1.1 Motivation

Human values are defined in social sciences as *'what people hold important in their [lives]'* [1]. These values, such as privacy or universalism, are important to consider in software development because people feel threatened if their values are compromised [1]. This consideration is not only for the steps of software development (e.g. from initial study to evaluation), but also the use of the software. Several recent incidents have demonstrated people's awareness of their values and how strongly people react to the violation of their values in software. For example, recent changes in WhatsApp's terms and privacy policy led millions of users to migrate to alternative messaging applications [2]. One of the reasons was the fear of another privacy breach by WhatsApp's parent company, Facebook, which will have access to users' data after the new policy begins [2]. In this case, users' trust in Facebook did not seem to have recovered after the infamous Cambridge Analytica case in 2015 [3]. Another similar case was demonstrated in a protest by digital and human rights groups about using facial recognition systems in justice systems that introduce bias against minorities [4]. The bias causes minorities more likely to be detected as offenders and, in turn, increases the fear of unfair treatment [4].

Software practitioners, such as developers or requirements engineers, need to consider (i.e. *be attentive to the implication of*) human values during software development to mitigate potential breaches and win over their users. Research in marketing science suggests that nontechnical factors, such as epistemic values (e.g. curiosity, novelty, and a desire for knowledge) [5], and costs or attractiveness [6] could influence users' decisions to choose and use an application. For instance, a privacy-aware user would choose an application that respects users' privacy. In this case, human values act as *'criteria or standards'* [1] for users when selecting and using an application. Considering human values in an application would offer competitive advantages over the millions of applications available in the market. These benefits and the importance of human values suggest that supporting their consideration during software development is necessary.

1.2 Research Objective and Questions

Challenges need to be addressed to pursue the objective of considering human values in software development. This thesis focused on the following two challenges. The first challenge was the abstract concept of human values that results in the lack of understanding and definition of human values in software engineering contexts [7]. For this reason, previous work on values in software [8–12] has chosen to rely solely on existing human values models from the social sciences (e.g. Schwartz's values [1]). Other studies in human-computer interaction (HCI) fields (e.g. [13, 14]) tend to consider a specific set of values (e.g. values with ethical implications). The second challenge concerned the consideration of human values in software development life cycles. Previous studies, such as those on value-based requirements engineering [15], value-sensitive design [13, 14], and value-sensitive software development [16], mainly focused on the early phases of software development (e.g. requirements and design). The later phases of development, such as implementation or release, have received limited attention. Moreover, current studies have not necessarily provided practical tools to support wider human values in software development.

More efforts are needed to understand human values in software engineering contexts and support their consideration in software development. To this end, this thesis used software development artefacts as a potential source for identifying and understanding values. Earlier literature has suggested that technological artefacts embody human values [17]. Studies in software engineering have investigated certain values, such as *privacy* [18, 19], *security* [20–23], *accessibility* [24], and *energy efficiency* [25, 26] in software development artefacts, e.g. issues or source code commits. Although these studies did not identify these concepts as human values, they demonstrated that these artefacts are a potential source for identifying and understanding values. Because there is no human values model defined specifically for software engineering yet, this thesis chose to start with using the definitions and characteristics of human values as described in Schwartz works [1, 27, 28] to find values in software development artefacts. Furthermore, to accommodate the possibilities of other values specific for software engineering, the study of values in this thesis is not limited to the values listed in the Schwartz model, but also values that are considered important by the software stakeholders. In this way, investigating software development artefacts allows us to understand human values in software engineering contexts to address the first challenge.

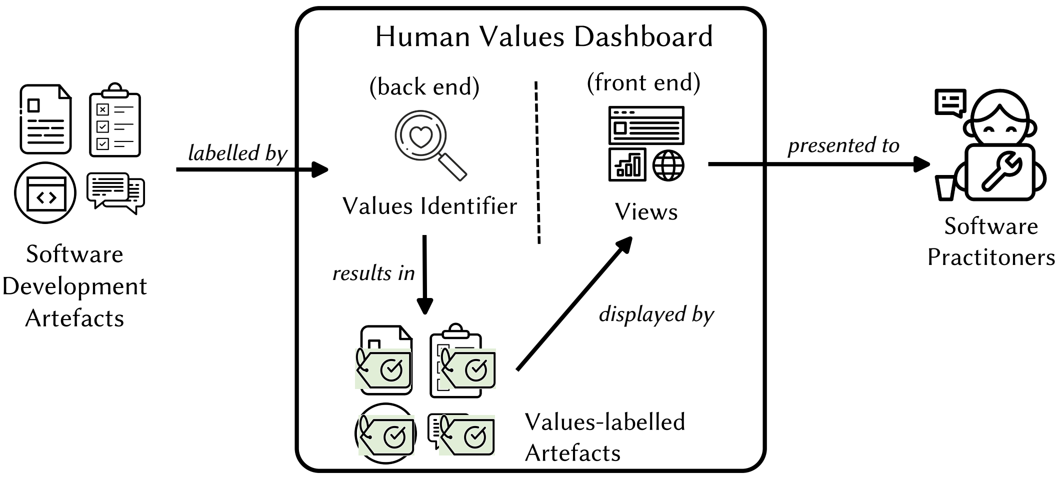


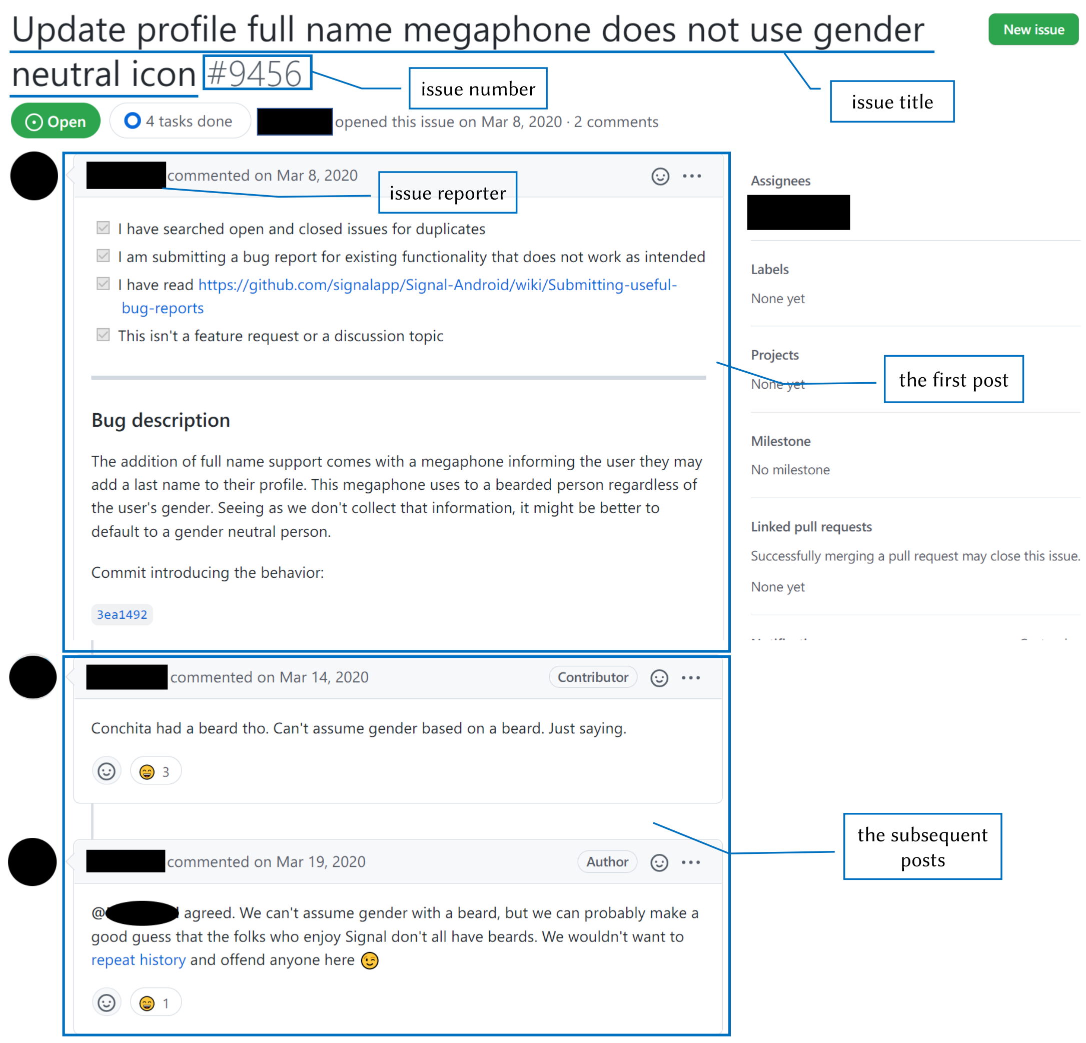
FIGURE 1.1: Proposed human values dashboard

For the second challenge, this thesis proposes a human values dashboard as a tool to address the limited support of considering human values in software development. Accordingly, this dashboard uses software development artefacts because these artefacts capture the software development process. Figure 1.1 illustrates how the proposed human values dashboard works. The dashboard consists of a back end and a front end. The back end of the dashboard detects and labels values in software development artefacts. Then, the dashboard's front end provides different views of the labelled artefacts to help software practitioners take values into account during software development. Because there is a limited understanding of such a dashboard, this thesis first investigated what artefacts would be suitable for it. For instance, these artefacts should be available in multiple stages of software development, including the latter stages, and have *human values present* (i.e. mentioned or expressed). Then, this thesis investigated how the dashboard could utilise these artefacts and support the consideration of human values in software development.

Among various software development artefacts, this thesis focused on GitHub Issues. GitHub is one of the most popular software repositories and is commonly used in software engineering research [29]. GitHub Issues is an issue tracking system that allows discussions (i.e. *issue discussions*[1]) [30] where project stakeholders can communicate their concerns, problems, and feedback during software development [31, 32]. Issue discussions as the object of this thesis is different with GitHub Discussions that was introduced later by GitHub *'for asking questions or discussing topics outside of specific Issues or Pull Requests'* [33].

Figure 1.2 shows an example of a post in GitHub Issues. An issue starts with a post with a title and a description from a reporter. GitHub allocates an issue number for each post. The first post is followed by subsequent posts by other contributors or the reporter. Because of this discussion format, there is virtually no limitation in terms of what contributors can discuss in an issue. For this reason, previous studies have used issues to study how software development teams address them [32, 34], manage requirements [35], and make software design decisions [36, 37]. In this case, in an issue discussion, project contributors may express concerns and feedback, if the values are lacking in the software, or appreciate the presence of values in the software. Thus, the presence of human values in an issue discussion becomes an indicator whether human values are present or lacking in the software. For this thesis, issue discussions allow for the understanding of human values in software development contexts and the development of a tool to support software development. This thesis first evaluated issue discussions to understand if human values were present. Then, to reduce the manual analysis efforts, a number of classification methods were developed to automate the detection of values. Next, an exploratory study was conducted to investigate how the dashboard could support software practitioners in considering human values during software development. A human values dashboard was then developed based on participants' needs in the exploratory study. Finally, an interview study with software practitioners was conducted to confirm the dashboard's usefulness and gather feedback for improving the dashboard.

[1]This thesis will use the terms *'issue discussions'* and *'issues'* interchangeably.



Update profile full name megaphone does not use gender neutral icon #9456

New issue

Open   4 tasks done   [redacted] opened this issue on Mar 8, 2020 · 2 comments

[redacted] [redacted] commented on Mar 8, 2020

- I have searched open and closed issues for duplicates
- I am submitting a bug report for existing functionality that does not work as intended
- I have read https://github.com/signalapp/Signal-Android/wiki/Submitting-useful-bug-reports
- This isn't a feature request or a discussion topic

Bug description

The addition of full name support comes with a megaphone informing the user they may add a last name to their profile. This megaphone uses to a bearded person regardless of the user's gender. Seeing as we don't collect that information, it might be better to default to a gender neutral person.

Commit introducing the behavior:

3ea1492

[redacted] [redacted] commented on Mar 14, 2020   Contributor

Conchita had a beard tho. Can't assume gender based on a beard. Just saying.

😄 3

[redacted] [redacted] commented on Mar 19, 2020   Author

@[redacted] agreed. We can't assume gender with a beard, but we can probably make a good guess that the folks who enjoy Signal don't all have beards. We wouldn't want to repeat history and offend anyone here 😉

😄 1

Assignees

[redacted]

Labels

None yet

Projects

None yet

Milestone

No milestone

Linked pull requests

Successfully merging a pull request may close this issue.

None yet

FIGURE 1.2: Example of an issue in GitHub
(Boxes in the blue outline indicate the components of a GitHub issue)

Research objective: To understand how human values are present in issue discussions and how a human values dashboard utilising this presence could support the consideration of human values in software development.

To realise the objective of this thesis, three high-level research questions, along with their sub-questions, were defined. Table 1.1 presents these research questions and their corresponding chapters. The following discussion describes these research questions.

RQ1 To what extent are human values present in issue discussions?

Before developing the dashboard, it was necessary to understand how human values are present (i.e. mentioned or expressed) in issue discussions. The first stage of this thesis investigated this by conducting an empirical study of three open-source Android applications: Signal Android, K-9 Mail, and Firefox Focus. This study focused on finding what values are evident in issues and their prevalence across projects. The empirical study used the following sub-questions:

RQ1.1 What values are discovered in issue discussions?
RQ1.2 Does the presence of values differ across projects? If so, how?

RQ2 To what extent can the detection of human values be automated?
Manual analysis of values in issue discussions requires considerable effort. Therefore, automating the human values analysis in issue discussions is essential for the analysis to be practical for software practitioners. This thesis investigated whether this analysis could be automated using classification methods. This study evaluated some well-known classification methods, namely, support vector machines, random forest, logistic regression, and multi-layer perceptron and assessed which approach offers the best performance. The evaluation also involved three additional experimental parameters: the unit of classification, resampling technique, and feature set. The unit of classification parameter represents the scope of the classification, i.e. the whole issue or a single post in issues. The resampling technique parameter represents a set of techniques used to handle the imbalanced nature of the dataset obtained from the previous stage. Meanwhile, the feature set parameter represents the feature extracted from the dataset for classification. To understand the experimental parameters' influence on the detection, the sub-questions for this stage were defined as follows:

RQ2.1 How does a different unit of classification affect the performance of the detection of human values?
RQ2.2 How does the use of resampling techniques affect the performance of the detection of human values?
RQ2.3 How does the use of different feature sets affect the performance of the detection of human values?
RQ2.4 To what extent is the performance of detecting human values influenced by the chosen classification methods?

RQ3 How can the automated detection of human values be useful in software development?
The automated detection of human values requires visualisations and interfaces to be practical to software practitioners. The proposed human values dashboard aims to

provide these functionalities. To start, it was necessary to understand whether software practitioners consider human values important. It was also essential to gather insights from practitioners on the potential benefits of the dashboard for software development, which artefacts they think are suitable for the dashboard, and what they need from a dashboard to support their software development activities. To obtain such insights, an exploratory study was conducted with the following sub-questions:

RQ3.1 What are the perceptions of practitioners towards human values in software development?

RQ3.2 Who will benefit from and what are the potential benefits of a human values dashboard?

RQ3.3 Are software development artefacts suitable for identifying values for the dashboard? If so, which artefacts?

RQ3.4 What is needed for a human values dashboard to be helpful in software development?

A human values dashboard was developed based on the findings in the exploratory study. This dashboard uses an automated detection method evaluated in **RQ2**. After the human values dashboard was created, it was necessary to confirm whether it is helpful to practitioners and how they perceive the automated detection results. It was also necessary to obtain feedback from practitioners to improve the dashboard. To obtain such insights, a feedback study was conducted with the following sub-questions:

RQ3.5 To what extent do practitioners find the human values dashboard useful?

RQ3.6 How do practitioners perceive the performance of the automated human values detection?

1.3 Thesis Contributions

The key contributions of this thesis can be summarised as the following:

1. This thesis presents the design and execution of an empirical study to evaluate the presence of human values in issue discussions (**RQ1**). As part of this study, the thesis also:
   - provides criteria to evaluate the presence of human values in issue discussions,

- provides contextualised software engineering descriptions of human values from issue discussions, and
- provides a publicly available ground truth for the development of an automated human values detection system.

2. This thesis presents the evaluation of classification methods to automatically detect the presence of human values in issue discussions (**RQ2**).

3. This thesis presents a human values dashboard as a tool to support the integration of human values in software development (**RQ3**). As part of the development of this tool, the thesis also:

   - captures software practitioners' perceptions on human values in software development,
   - identifies high-level requirements for the development of a human values dashboard,
   - compiles the dashboard's potential benefits in various software development roles and phases,
   - evaluates the dashboard from the software practitioners' point of view,
   - confirms that the dashboard could be helpful in software development, and
   - obtains feedback for the improvement of the dashboard.

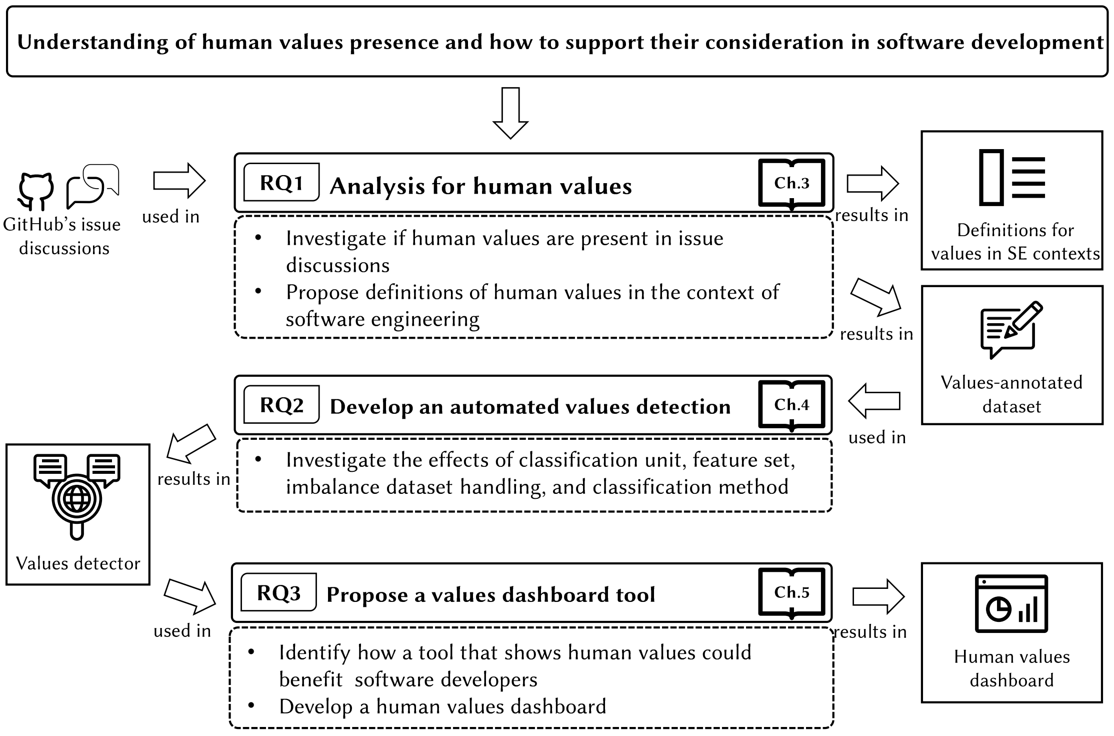
1.4 Thesis Outline

The thesis consists of six chapters. Some of these chapters are derived from publications. Figure 1.3 shows the overview and organisation of the chapters that present the research question of this thesis. The thesis is organised as follows:

Chapter 1 describes the motivation, the research questions, the contributions, and the organisation of this thesis.

Chapter 2 provides the research background and introduces definitions for the terms used in the thesis.

Chapter 3 addresses research question **RQ1** by conducting an empirical study to investigate the presence of human values in issue discussions in three Android applications. First, it provides criteria for identifying values. Second, it lists the themes



Understanding of human values presence and how to support their consideration in software development
GitHub's issue discussions
used in
RQ1 Analysis for human values
Ch.3
Investigate if human values are present in issue discussions
Propose definitions of human values in the context of software engineering
results in
Definitions for values in SE contexts
results in
Values-annotated dataset
used in
RQ2 Develop an automated values detection
Ch.4
Investigate the effects of classification unit, feature set, imbalance dataset handling, and classification method
results in
Values detector
used in
RQ3 Propose a values dashboard tool
Ch.5
Identify how a tool that shows human values could benefit software developers
Develop a human values dashboard
results in
Human values dashboard

Figure 1.3: Thesis overview and organisation

discovered and their descriptions in issue discussions. Third, it presents the prevalence of these themes in the applications under study. This chapter is derived from the publication:

🗋 **Arif Nurwidyantoro**, Mojtaba Shahin, Michel Chaudron, Waqar Hussain, Rifat Ara Shams, Harsha Perera, Gillian Oliver, Jon Whittle, *Human values in software development artefacts: A case study on issue discussions in three Android applications*, Information and Software Technology Journal: Special Issue on Waste and Value in Software Engineering, Vol.141, January 2022. [Impact Factor(2021): **2.73**, SJR rating(2020): **Q2**]

Chapter 4 addresses research question **RQ2** by evaluating a number of classification methods and considering the parameters for the unit of classification, resampling techniques, feature sets, and classification methods.

Chapter 5 addresses research question **RQ3** by developing a human values dashboard to support the consideration of human values in software development. First, it describes an exploratory study conducted to gather requirements and how the dashboard could be helpful in software development. Second, it presents the analysis and implementation of the dashboard. Third, it describes a study performed to gather feedback to improve the dashboard. The exploration stage of this chapter is derived from the publication:

- **Arif Nurwidyantoro**, Mojtaba Shahin, Michel Chaudron, Waqar Hussain, Harsha Perera,Rifat Ara Shams, Jon Whittle, *Towards a Human Values Dashboard for Software Development: An Exploratory Study*, Proceedings of the 15th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, October 2021. [Core rating: **A**, Acceptance Rate: **19.4% (24/124)**]

Chapter 6 concludes the thesis and provides potential directions for future works.

Table 1.1: Overview of the research questions, sub-questions, and their corresponding chapters

Research Questions	Sub-Questions	Chapter #
RQ1 To what extent are human values present in issue discussions?	**RQ1.1** What values are discovered in issue discussions?	Chapter 3
	RQ1.2 Does the presence of values differ across projects? If so, how?	
RQ2 To what extent can the detection of human values be automated?	**RQ2.1** How does a different unit of classification affect the performance of the detection of human values?	Chapter 4
	RQ2.2 How does the use of resampling techniques affect the performance of the detection of human values?	
	RQ2.3 How does the use of different feature sets affect the performance of the detection of human values?	
	RQ2.4 To what extent is the performance of detecting human values influenced by the chosen classification methods?	
RQ3 How can the automated detection of human values be useful in software development?	**RQ3.1** What are the perceptions of practitioners towards human values in software development?	Chapter 5
	RQ3.2 Who will benefit from and what are the potential benefits of a human values dashboard?	
	RQ3.3 Are software development artefacts suitable for identifying values for the dashboard? If so, which artefacts?	
	RQ3.4 What is needed for a human values dashboard to be helpful in software development?	
	RQ3.5 To what extent do practitioners find the human values dashboard useful?	
	RQ3.6 How do practitioners perceive the performance of the automated human values detection?	

Chapter 2

Background

This chapter introduces the theoretical background and previous works related to this thesis. This chapter is organised as follows: Section 2.1 defines the definition of human values and their models in social science. Section 2.2 describes recent studies of human values in software engineering. Section 2.3 describes earlier studies of human values in the software development life cycle. Section 2.4 describes previous studies that investigated human values in software development artefacts. Section 2.5 presents earlier work on human values detection in both text documents and software development artefacts. Lastly, Section 2.6 describes studies on dashboards for software development.

2.1 Human Values

Human values have been studied in social science for decades. Cheng and Fleischmann summarised the definition of human values as *'guiding principles of what people consider important in life'* [38]. In addition to this definition, social scientists have also proposed several characteristics of human values. First, human values are associated with *'desirable end states or modes of conduct'* [27, 39]. Second, human values *'serve as guiding principles by providing criteria or standards of preference in selecting actions or policies'* [1, 27, 39–41]. Third, the types of those values (e.g. Figure 2.1) *'transcend specific situations'* [27, 42] of any *'individual, group, or society'* [39, 42]. However, the relative importance and meaning of those values could be different for their holders [1, 28, 39, 43]. For example, Hanel et al. found that people in Brazil value *travelling* as a *pleasure*, while people in the United Kingdom value it as a *freedom* [43]. Fourth, human values are *infused with feeling* such that a threat or a consideration of human values could respectively upset or excite the holder of those values [1].

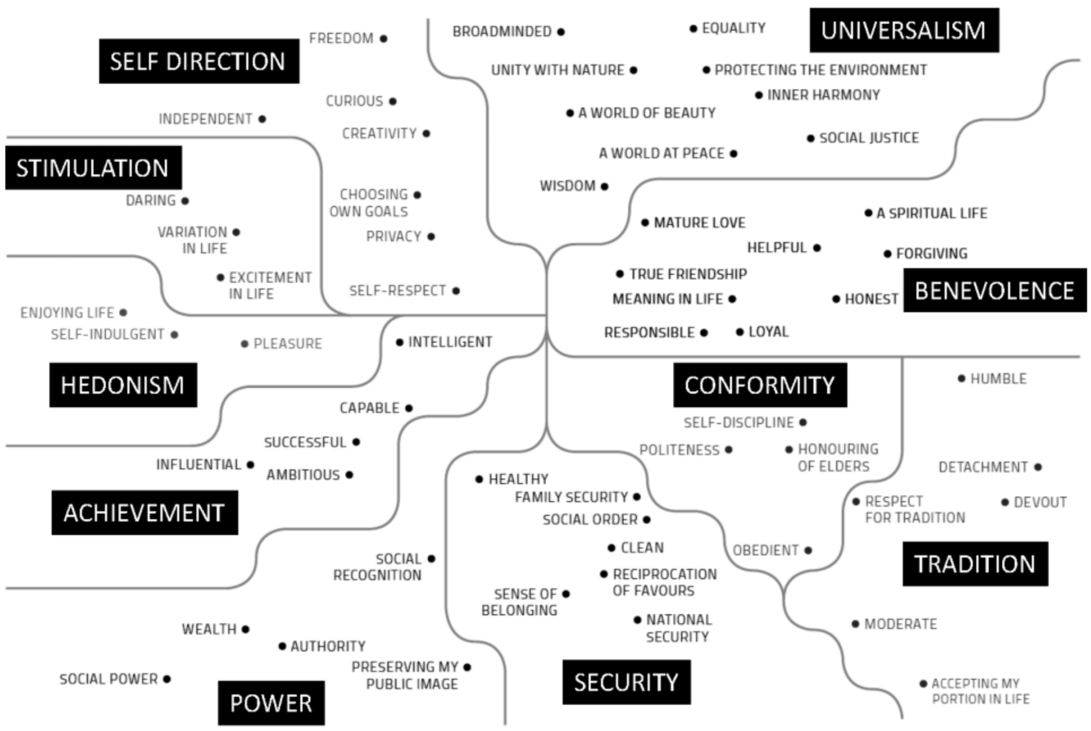


FIGURE 2.1: The Schwartz models of basic human values [1] taken from [44]

Social science studies have also proposed several human values models. In these models, the bigger concept of human values is divided into smaller categories. For example, Rokeach proposed a model called the Rokeach Value Survey that distinguishes terminal values (or ultimate goals such as *pleasure, social recognition, or a sense of accomplishment*) and instrumental values (or standards that guide behaviour such as being *ambitious, courageous, or forgiving*) [39]. Another model proposed by Schwartz [1, 45] not only categorises the values but also structures the relations between the value categories. In the earlier version of the model, which is also known as Schwartz's theory of basic values, human values are categorised into 10 types (i.e. *basic values*), such as *benevolence*, *stimulation*, and *universalism*, along with their *'exemplary specific values that primarily represent each value type'* [27] or *value items* [1], such as *privacy*, *freedom*, and *equality*. In this model, human values are arranged in a circular manner such that supporting values are adjacent to each other, while values that are conflicting are diametrically opposite each other [1]. For example, pursuing freedom could introduce conflict to the value of respecting tradition. Figure 2.1 shows the Schwartz original theory of basic values consisting of 10 values (highlighted in black) and each of their corresponding value items. In the newer version of the model (Figure 2.2), the definition of values has been refined, resulting in 19 basic values arranged similarly. Table 2.1 shows the 19 values in the refined Schwartz model along with their definition.

TABLE 2.1: List of 19 values in Schwartz's Refined Theory of Human Values [28]

No.	Value	Conceptual Definition
1	Self-direction—thought	Freedom to cultivate one's own ideas and abilities
2	Self-direction—action	Freedom to determine one's own actions
3	Stimulation	Excitement, novelty, and change
4	Hedonism	Pleasure and sensuous gratification
5	Achievement	Success according to social standards
6	Power—dominance	Power through exercising control over people
7	Power—resources	Power through control of material and social resources
8	Face	Maintaining one's public image and avoiding humiliation
9	Security—personal	Safety in one's immediate environment
10	Security—societal	Safety and stability in the wider society
11	Tradition	Maintaining and preserving cultural, family or religious traditions
12	Conformity—rules	Compliance with rules, laws, and formal obligations
13	Conformity—interpersonal	Avoidance of upsetting or harming other people
14	Humility	Recognizing one's insignificance in the larger scheme of things
15	Universalism—nature	Preservation of the natural environment
16	Universalism—concern	Commitment to equality, justice and protection for all people
17	Universalism—tolerance	Acceptance and understanding of those who are different from oneself
18	Benevolence—caring	Devotion to the welfare of in-group members
19	Benevolence—dependability	Being a reliable and trustworthy member of the in-group

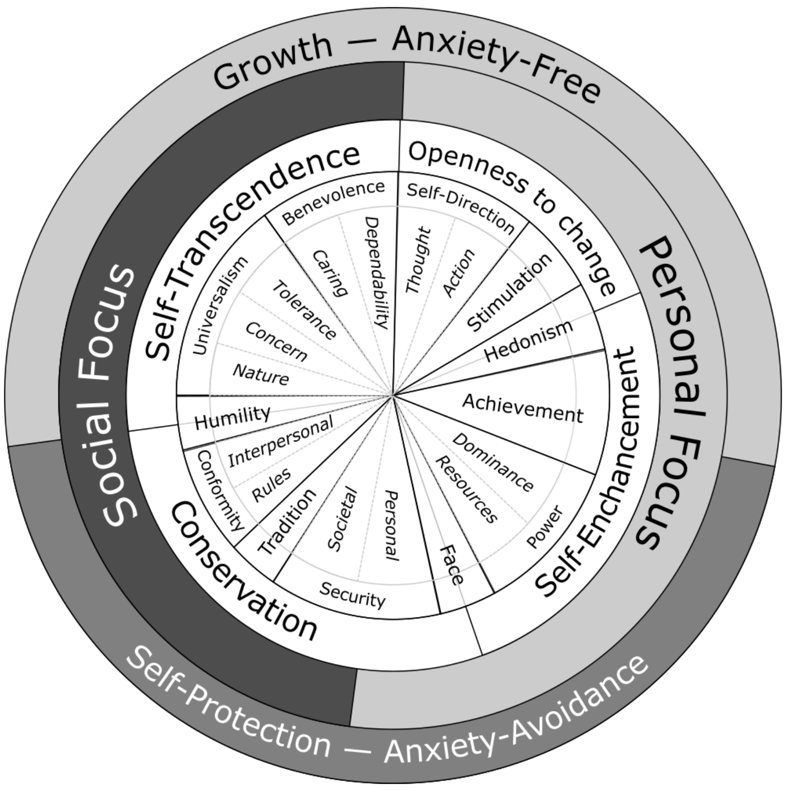


Figure 2.2: The Schwartz's refined model of human values taken from [28]

This thesis used Schwartz's model as its primary source to illustrate human values. It used Schwartz's model for the following reasons: First, Schwartz's model has been well studied and universally validated [28]. Second, this model covers most of the values from the other existing human values models [38]. Third, this model has been widely used in both social science and other disciplines such as computer science [46], data mining [47], and software engineering [8].

2.2 Human Values in Software Engineering

Previous work has proposed the consideration of concepts similar to human values in software engineering. For example, Boehm and Huang have proposed value-based software engineering [48], which were focused on monetary values, such as software costs and development budgets [8, 48]. Another work has focused on human-centric issues in software engineering, such as personality, technical proficiency, gender, age, and culture [49]. Although some of the human-centric issues overlapped with human values, they were not derived from human values concepts in the social sciences.

Other recent studies that work on human values have used definitions from social science. These studies include proposing frameworks or techniques to integrate human values in software. For example, Thew and Sutcliffe proposed value-based requirements engineering as a method to analyse values, motivations, and emotions during requirements engineering [15]. This method proposed taxonomies that are derived from social sciences. Furthermore, this method also provides a list of scenarios, questions, and process advice to be implemented during requirements elicitation. Perera et al. proposed a framework to consider and evaluate values during software development called Continuous Value Assessment (CVA). This framework also considers values from social sciences for their values brainstorming and evaluation steps [11]. Another technique was proposed by Winter et al. called Values Q-Sort. This technique used Schwartz values as a basis to investigate the values of software practitioners [50]. Unlike these works that proposes new frameworks or techniques, Hussain et al. in one of their findings suggested modifying existing techniques in software development to incorporate values (e.g. 'values stories' instead of user stories). Their work also adopted values as defined in social sciences.

The definitions of human values from social sciences were also used in recent studies to analyse their presence in software engineering. For example, Perera et al. used the Schwartz model to analyse values in software engineering publications [10]. Similarly, Shams et al. adopted Schwartz values in the analysis of user reviews in mobile applications [12]. Studies in the field of human-computer interaction (HCI), such as Value Sensitive Design (VSD) [14, 52], have also adopted definitions of human values from social science, with an emphasis on those that have ethical implications.

Although those recent works used human values from social sciences, Mougouei et al. [7] argued that these definitions of human values need to be contextualised for software engineering to be practical. To address this gap, this thesis attempted to propose contextualised software engineering definitions for some of those human values found in software development artefacts. Some values, such as *privacy* and *security*, arguably have a corresponding concept in non-functional requirements (NFRs). In software development, NFRs are associated with quality properties that can influence the degree of satisfaction of a software [53–55]. Regarding this, there is a disagreement in the software engineering community on whether human values and NFRs are distinct. One could argue that human values are types of NFRs. Alternatively, it could be argued that human values are distinct from NFRs but can benefit from utilising existing NFRs frameworks. This rationale comes from the argument that '*human values are* *associated with the moral concerns of users instead of using system perspective in NFRs*' [55]. This thesis supports the argument that human values can be described in a much broader sense than NFRs. However, ultimately, most human values (e.g. the Schwartz model) have not been considered explicitly in the software engineering literature [10].

2.3 Human Values in the Software Development Life Cycle

There are recent studies in software engineering that have attempted to integrate values into software development. Some studies have focused on the consideration of human values in the early phases of software development, such as the requirements phase (e.g. value-based requirements engineering [15], continual values assessment [11]) or design phase (e.g. value-sensitive design (VSD) [14, 52]). The frameworks presented in these studies involve capturing users' values and evaluating the implications of those values during the development of an application. Unlike these previous studies, other works have attempted to introduce the consideration of human values into the software development life cycle. For instance, a framework called values-first software engineering was proposed to gather users' feedback on human values using prototyping [8]. Another recent study identified intervention points and proposed modifications to an agile software development framework to incorporate values in software development techniques [51]. The methods or frameworks proposed in these studies attempted to support human values in software development. However, these works either focused on early software development phases or did not necessarily have the tool support for integrating human values in software development. This thesis attempts to address this gap by proposing a tool to support considering human values in the latter stages of software development.

2.4 Human Values in Software Development Artefacts

Earlier literature suggested that technological artefacts embody human values [17]. In software engineering, this idea has been supported by previous works that investigated human values in software development artefacts. These studies mostly considered human values that are well known in software engineering, such as *security*, *privacy*, and *energy efficiency*. For example, some studies investigated the notion of security in source code [22, 23] and issue discussions [20, 56]. Privacy has received much attention in previous studies investigating source code and configuration file artefacts [18, 19, 24, 57, 58], and project documentation [59]. Other studies have used source codes [26] and their commits [25] to understand and support energy efficiency in software development. Although related, these prior studies do not specifically discuss these concepts as human values. However, because the studies show that the concepts are found in the artefacts, they support the idea of investigating software artefacts (e.g. issue discussions) for human values.

2.5 Automated Human Values Detection

This section first describes earlier works on automated detection of human values in text documents, and then presents previous studies on automated detection of human values in software development artefacts.

Text documents – Earlier studies attempted to detect human values in text documents, such as net neutrality debate [60, 61] and nuclear power debate documents [62, 63]. These studies demonstrated the need for incremental efforts to improve the performance of the detection. A series of studies have been done to automate the detection of values in net neutrality debate documents. Initially, Ishita et al. used k-nearest neighbour method for a multilabel classification of values, resulting in an F1 score of 0.48 as their top performer [60]. Following this, several classification methods were used to improve the performance of the detection, namely support vector machine (SVM) [64, 65], naive bayes [65], latent value model (LVM) [61], and simulated annealing [66, 67]. These studies implemented a multilabel classification of values as a series of binary classifications for each value. The best performance was an average F1 score of 0.74 for the simulated annealing approach [67]. Another study attempted to automate the detection of human values in nuclear power debate documents written in the Japanese language. These studies started with a study that reported a varied performance of precision (0.05 to 0.38) and recall (0.10 to 0.34) for the detection of specific values [62]. A follow-up study reported that a deep learning–based method, namely fastText, could reach an F1 score of 0.85 in larger datasets [63].

Software artefacts – Earlier studies on the detection of human values in software artefacts primarily focused on specific concepts that are well known in software engineering, such as security (e.g. [20, 56, 58, 68–72]), privacy (e.g.[19, 57, 58, 68]), or energy efficiency (e.g. [25, 73]). These studies used a variety of approaches to detect these concepts. Some of the studies used a keyword-based approach by specifying terms related to the values that they wanted to detect (e.g. security [56] or energy efficiency [25]). Some others extracted topics related to specific values in the artefacts by using topic modelling approaches [20, 58, 70]. Other studies investigated source code artefacts using static analysis to detect violations of security [57] or privacy [19]. Machine learning approaches, such as naive bayes, logistic regression, and deep learning have also been used to detect these specific values in natural language-based artefacts ([69, 71, 72]), such as issue discussions or requirements documents. Some of the works, especially those that used the keyword-based ([25, 56]) and topic modelling approaches ([20, 58, 70]), did not report the performance of their approach. Others reported a high performance of their approach. For instance, in [71], the convolutional neural network was used to detect security in issue discussions and requirements with an accuracy of 0.96. Another study reported the best F1 score of 0.90 in identifying security discussions in microservices systems [69]. Even though these studies did not specifically discuss these concepts as human values, they demonstrated the possibility to automatically identify human values in software development artefacts.

Besides those previous studies that focused on some specific concepts related to human values, a recent work by Li et al. has considered broader human values by focusing on their violation in Android applications [74]. This work developed rule-based approaches to detect software defects that corresponded to the violation of values in Android APIs. The approaches proposed in this work resulted in an average accuracy of 0.93 and an average recall of 0.84 across all APIs (i.e. animation, media, mtp, nfc, telephony, and hardware) and all values (i.e. hedonism, self-direction, universalism, security, and conformity)

2.6 Dashboards for Software Development

Dashboards are generally used in organisations to monitor progress [75] and support decision-making [76]. In software development, recent studies have demonstrated the use of a dashboard to make decisions [77, 78] and promote awareness about a software project to the development team [79, 80]. To support decision-making, earlier works incorporated software quality indicators. For example, López et al. provided indicators, such as code quality and non-blocking codes, to support decision-makers in agile software development [81]. Similarly, Thiruvathukal et al. developed a dashboard that presents the size, issue density, issue spoilage, and productivity of a software development project [82]. Other dashboard studies focused on increasing the awareness of the development team. For instance, Leite et al. proposed a dashboard to make developers aware of unusual events in repositories [83]. Another study used a dashboard to visualise concerns in the context of software evolution [84]. In practise, software development teams generally used dashboards to monitor the development activities of a project [85–88]. However, these earlier studies focused mainly on technical aspects of software development. This thesis bridges this gap by proposing a human values dashboard for software development. Similar to previous work, the proposed value dashboard uses artefacts available in software repositories (e.g. [85–88]) and provides indicators to support software development activities (e.g. [82, 89]). However, unlike previous studies, the dashboard discussed in this thesis offers different indicators (i.e. human values perspective) in software development.

Chapter 3

Investigating Human Values in Issue Discussions

Although human values have been extensively investigated in the human-computer interaction field, little effort has been made to study and consider human values in software development. GitHub Issues is a potential artefact to investigate the presence of human values in the latter stages of software development. In this artefact, users and developers share and communicate their concerns for the software. This chapter presents a qualitative study to investigate to what extent human values are present in issue discussions (**RQ1**). To this end, an empirical study was conducted to analyse the issue discussions of three Android applications, namely Signal, K-9 Mail, and Firefox Focus. This study identified 20 value themes and proposed descriptions for each theme in software engineering contexts. Moreover, this study found that the functionalities and statement of values could contribute to the presence of values discussion in an application. The findings suggest that human values are indeed present in issues for which automated tools can be developed.

3.1 Introduction

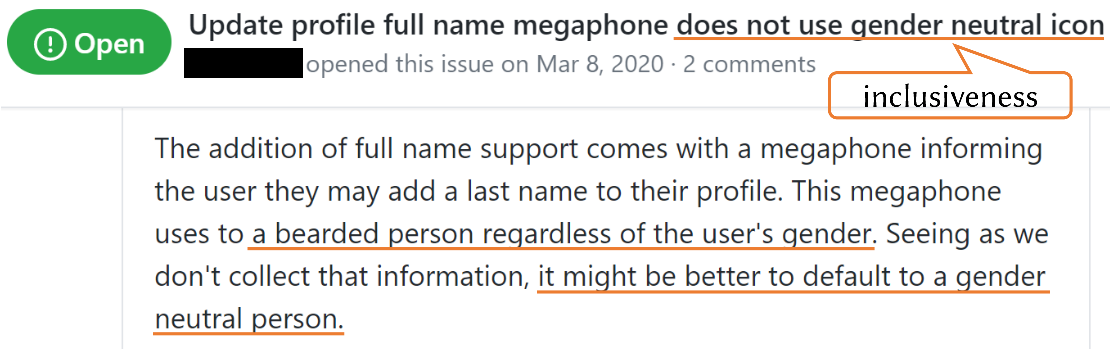
As described in Chapter 1, studies in marketing sciences have suggested that socio-technical factors, such as human values, in an application could influence users' decisions to install and use an application [5, 6]. This consideration of human values could offer software practitioners competitive advantages in the crowded application markets. However, considering human values in software development is not a trivial effort. The understanding of human values in software engineering contexts is still limited [7]. Furthermore, previous studies (see, e.g.,[13–15]) have mainly focused on the early stages of the development phases. Therefore, more efforts are needed to support human values in the later stages of software development, especially in the implementation or release phases.

The study described in this chapter addressed this gap by analysing human values in issue discussions. This approach was chosen over observational or ethnographic studies for several reasons. First, data on issue discussions are largely available and capture the software development process from the beginning to the latest. Meanwhile, observational or ethnographic studies need to be performed repeatedly to cover a large portion of the software development process. Second, observational or ethnographic studies have the potential to introduce some biases. These studies require voluntary participation, which does not necessarily represent all practitioners [90, 91]. In addition to that, these studies also have the potential for the Hawthorne effect, which occurs when participants behave differently because they are observed [91, 92]. Furthermore, these studies can involve direct questions to participants that can lead to response bias, a case in which participants may not respond accurately [90]. On the other hand, analysis of issue discussions involves participants in more realistic conditions (i.e. contributors who are involved in the discussions are those who are genuinely attracted to the issues). Additionally, the issues provide realistic discussions without the assumptions that the discussions will be analysed.

Issue discussions were chosen because they could be generated in any phase, especially during the implementation and release phases. Additionally, a discussion of an issue (i.e. *issue discussion*) captures concerns, problems, and feedback from a project's stakeholders during software development [30–32]. Figure 3.1 shows how a user of an open-source application expressed his or her opinion of inclusiveness being present in the application as an issue report. Contributors of the application could respond to the report indicating their consideration of the suggestion. This example supports the idea that it is possible to discover human values in issue discussions.

Other software development artefacts may also potential as a source for discovering human values. For example, codes of conduct in open source projects provide the standard behaviour (e.g. friendly or inclusiveness) in software development activities [93]. However, this study focused on issue discussions as it facilitates the dynamic contributions of project stakeholders throughout the software development process.

Moreover, codes of conduct is a guideline of intent and not necessarily followed by the development team.



Update profile full name megaphone does not use gender neutral icon

[redacted] opened this issue on Mar 8, 2020 · 2 comments

The addition of full name support comes with a megaphone informing the user they may add a last name to their profile. This megaphone uses to a bearded person regardless of the user's gender. Seeing as we don't collect that information, it might be better to default to a gender neutral person.

FIGURE 3.1: An example of human values identified in an issue discussion

Issue discussions can be found in modern software repository platforms. This study considered issue discussions from GitHub (i.e. GitHub Issues) among other platforms for several reasons. First, GitHub is one of the most popular repository platforms and is widely used in software development research [29]. Second, the issues functionalities in GitHub not only allow for discussion about application development but are also equipped with task management [94] and can be cross-referenced with other artefacts [95]. These features make GitHub Issues capture most software development activities.

This chapter describes the methods used to find human values in issue discussions (**RQ1**) and presents and discusses the results. The empirical study described in this chapter resulted in the following contributions:

1. This study identified 20 themes related to human values (i.e. *value themes*) in the issue discussions of three messaging applications and provided their mapping onto Schwartz's values.
2. This study provided contextualised software engineering descriptions for each of the 20 discovered value themes.
3. This study provided a dataset of 1,097 annotated issues that consisted of 5,615 posts. This dataset is publicly available [96] to be used by others to replicate, validate, or extend.

The remainder of this chapter is organised as follows: Section 3.2 discusses the research questions and the methodology of the study. Section 3.3 presents the results of the study. Section 3.4 discusses the results. The threats to the validity of this study are presented in Section 3.5. Finally, Section 3.6 concludes this chapter with a pointer to the next stage of this research.

3.2 Methodology

This empirical study aimed to address the first research question: *to what extent are human values present in issue discussions?* This thesis focused on the issue discussions artefact because it captures activities in the later stages of software development, such as the implementation phase [36, 97, 98] and the release phase [32, 70]. This artefact also provides users' perceptions on an application because it is generated by both the applications' users and contributors [99, 100]. Because human values are broad (see Figure 2.1), it is also necessary to understand which human values are present in issue discussions. Additionally, each application has different goals and functionalities. Thus, the presence of human values could be different from one application to another. Therefore, to address to what extent human values are present in issue discussions, this study defined the following sub-research questions:

RQ1.1 What values are discovered in issue discussions?

RQ1.2 Does the presence of values differ across projects? If so, how?

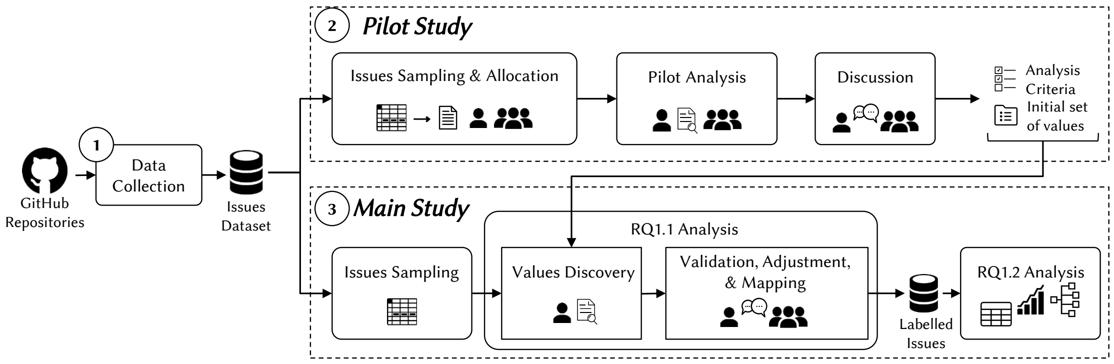


FIGURE 3.2: Methodology to answer RQ1

Figure 3.2 presents the methodology of this empirical study. The empirical study consisted of three phases: the data collection, pilot study, and main study. The data collection phase involved downloading issue discussions from each project's repository. In the pilot study, a smaller set of issue discussions was analysed to develop criteria for discovering the presence of values. The main study used these criteria to analyse the presence of human values. The following subsections describe each phase in more detail.

3.2.1 Data Collection

The case study investigated open-source Android projects for the following reasons: first, open-source projects' repositories are publicly available, allowing anyone interested (including users) to contribute to the projects [101]. The availability of the repositories also allowed for the collection of their artefacts for the case study. Second, the Android operating system is popular and has almost 73% of the market share of mobile operating systems worldwide (as of August 2021) [102].

This study used applications available in the Google Play Store, an official Android application store. The Google Play Store categorises its applications into several categories, such as Art and Design, Beauty, or Business. This study purposively considered applications under the Communication category. The Communication category includes applications that allow users to communicate or connect. This decision was made because applications in the Communication category tend to be used frequently and require more permissions than those in other categories [103, 104]. Consequently, a discrepancy in the behaviour of the applications (e.g. a breach of personal information) would encourage users to express their concerns. This increases the chance of finding human values being discussed.

To select the applications for the case study, the following criteria were developed:

- The application has a substantial number of downloads (e.g. at least five million downloads).
- The repository of the application has a substantial number of contributors (e.g. at least 100 contributors).
- The repository of the application has a substantial number of issue discussions (e.g. at least 2,000 issues).
- The application has a statement of supporting human values (e.g. *privacy* or *freedom*) on its homepage.

The first three criteria ensured the applications had adequate users, contributors, and issue discussions. The last criterion increased the chance of finding human values during analysis. Applying these criteria to review the applications under the Communication category resulted in the three applications listed in Table 3.1. These three applications, namely Signal Private Messenger, K-9 Mail, and Firefox Focus, serve different purposes, namely instant messenger, email client, and web browser, respectively. All three applications have a statement of supporting privacy, with additional statements that could be potentially considered as values (e.g. freedom, togetherness, ease of use, and efficiency). Analysing these three applications allowed for comparisons of the human values presence across the projects. The first two applications were used in the pilot study due to their similar functionalities. The main study included the third application to understand whether the different functionalities of an application influence the presence of values.

Table 3.1: Applications that met this study's criteria

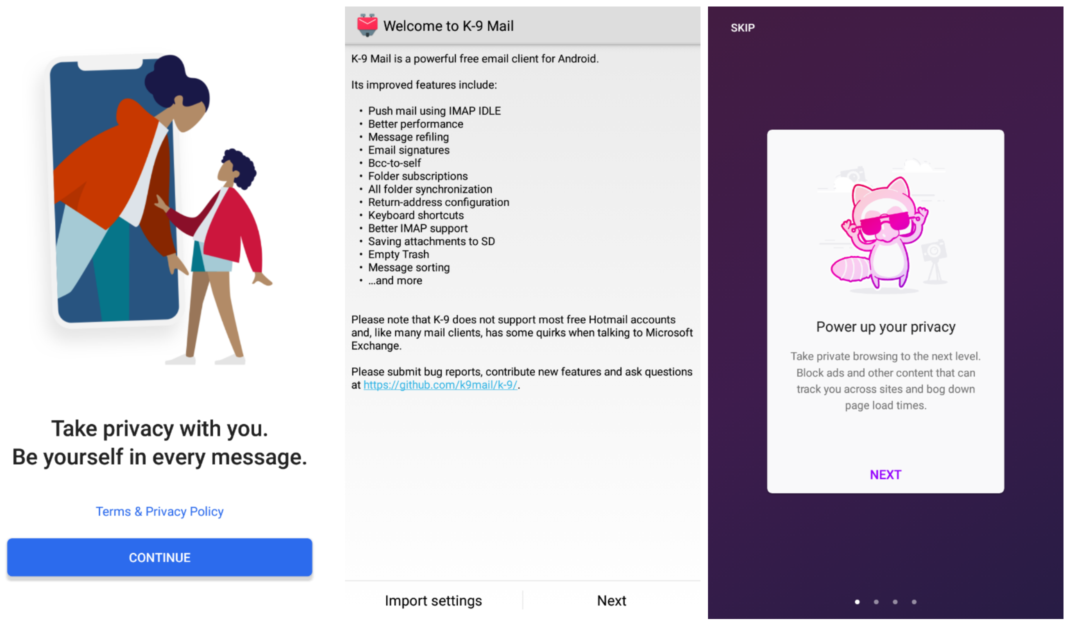
Application	**Type**	**# Downloads**	**# Contributors**	**# Issues**	**Values Statement**
Signal Private Messenger	Instant messaging	50 millions	248	6,899	privacy, freedom, togetherness
K-9 Mail	Email client	5 millions	232	2,499	privacy, ease of use
Firefox Focus	Web browser	5 millions	112	2,530	privacy, efficiency

These three projects are described as follows:

1. *Signal Private Messenger* (or ***Signal***) is a privacy-oriented messenger application supported by a non-profit organisation: the Signal Foundation. This application is determined to provide a private messaging to its users. Since its first public beta release in 2010 [105], this application has evolved from providing only encrypted SMS (this application is formerly known as TextSecure) to an integrated private communication application, which supports voice and video calls on top of the text messaging function. In 2015, Signal phased out support for encrypted SMS [106] in favour of data-based communication and support for PC and iOS. As of 1 October 2020, this application has been downloaded more than 10 million times in the Google Play Store. The website of Signal displayed a statement of privacy, freedom, and togetherness [107]. Figure 3.3a shows the welcome message of the application having a privacy statement.

2. *K-9 Mail* (or ***K-9***) is an email application supporting PGP/MIME for private communication [108]. K-9 intends to provide seamless interaction for users to send and receive encrypted emails [109]. As of 1 October 2020, this application has been downloaded more than five million times from the Google Play Store. The website of K-9 displayed a statement of privacy and ease of use [110]. Figure 3.3b shows the welcome message of the application.

3. Firefox Focus (or ***Focus***) is a dedicated privacy browser application provided by Mozilla. This browser has functionalities, such as blocking advertisements and online trackers, that result in better performance while maintaining the privacy and security of its users [111]. As of 1 October 2020, this application had been downloaded more than five million times from the Google Play Store. The website of Focus contains a statement of privacy and efficiency [112]. Figure 3.3c shows the onboarding page of the application showing a statement of privacy.



Welcome to K-9 Mail

K-9 Mail is a powerful free email client for Android.

Its improved features include:

- Push mail using IMAP IDLE
- Better performance
- Message refiling
- Email signatures
- Bcc-to-self
- Folder subscriptions
- All folder synchronization
- Return-address configuration
- Keyboard shortcuts
- Better IMAP support
- Saving attachments to SD
- Empty Trash
- Message sorting
- …and more

Please note that K-9 does not support most free Hotmail accounts and, like many mail clients, has some quirks when talking to Microsoft Exchange.

Please submit bug reports, contribute new features and ask questions at https://github.com/k9mail/k-9/.

Import settings
Next
SKIP
Power up your privacy
Take private browsing to the next level. Block ads and other content that can track you across sites and bog down page load times.
NEXT

(A) Signal (B) K-9 (C) Focus

FIGURE 3.3: Screenshots of the applications

Scripts were developed to collect issue discussions from each project's repository. These scripts utilised the github3.py library to access GitHub's public API (application programming interface). The collected dataset included 11,928 issues consisting of 62,526 posts from the Signal, K9, and Focus repositories. Table 3.2 shows the number of collected issues and their corresponding number of posts. These three projects were used in both the pilot study and the main study.

3.2.2 Pilot Study

The pilot study was carried out to inform two key study design decisions: (i) identifying the criteria to determine whether human values were present and (ii) exploring the possibilities to discover themes specific to software engineering that may not have

TABLE 3.2: The number of issues and posts collected

Project	Repository URL	# Issues	# Posts
Signal	https://github.com/signalapp/Signal-Android	6,899	39,580
K-9	https://github.com/k9mail/k-9	2,499	11,663
Focus	https://github.com/mozilla-mobile/focus-android	2,530	11,283
Total		**11,928**	**62,526**

been covered in the Schwartz model. An exploration of themes specific to software engineering was necessary because, to the best of our knowledge, there is no statement in prior works that Schwartz's model (or other values models) are exhaustive for software engineering fields. However, it is necessary to use the concepts and definitions of human values from Schwartz's theory to ensure a similar understanding between the analysts involved. Figure 3.4 shows how the analysis in the pilot study was related to Schwartz's theory. Initially, Schwartz's theory was used to understand human values. Then, the study reflected the definitions for the values analysis with Schwartz's values concepts and definitions.

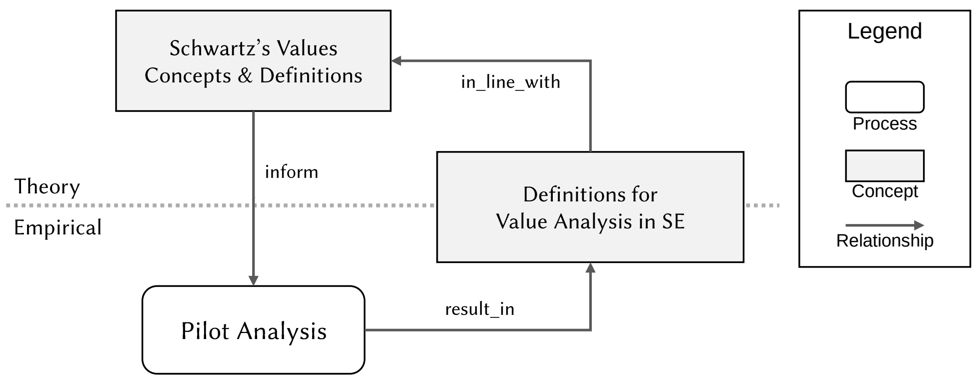


FIGURE 3.4: The relation between the pilot analysis and Schwartz's theory

3.2.2.1 Pilot Process

The pilot study consisted of three steps: issues sampling and allocation, pilot analysis, and discussions, described as follows:

Issues sampling and allocation: The pilot study focused on Signal and K-9, as they have similar functionalities. The pilot study investigated 30 randomly selected issues that contained 20 posts each from these two projects. In total, 600 posts were used for the pilot study. This number of issues and posts was chosen following the analysts' availability to perform the study. The pilot study involved six analysts (two females, four males) ranging from PhD students to an associate professor. All analysts had a reasonable understanding of human values before the study started. All except one had a software engineering background. Because values are subjective concepts [113], there was a risk that the analysts would interpret human values and their definitions differently. This risk was mitigated by asking the analysts to read seminal papers critically (e.g. [1, 28]) to ensure they had a similar understanding of the concept of human values. The analysts were then divided into five pairs, in which a *primary analyst* (i.e. the present author) was paired with the five other analysts (i.e. *secondary analysts*). The decision to have a primary analyst was made to overcome the limited availability of the other analysts.

Pilot analysis: Each secondary analyst was asked to analyse six issues independently. In parallel, the primary analyst analysed all 30 issues. The analysis was conducted independently by each analyst. The analysts were instructed to find whether a concept that could be related to human values was mentioned or expressed in issue discussions. The analysis followed the open coding approach to allow themes to emerge from the issue discussions (i.e. *emerging themes*). The analysts were allowed to propose any theme as a value without strictly following Schwartz's model. For such cases, the analysts were required to describe their emerging values based on the phrases they found in the issue discussions. There was no limitation on the number of values the analysts could find in a discussion. The analysts were given two weeks to finish their pilot analysis.

Discussions: Once the analysis process finished, the primary analyst held a meeting with each of the secondary analysts. These meetings mainly aimed to understand (i) the process and criteria employed by the analysts to discover values and (ii) the discovered themes and their descriptions to develop an initial set of emerging values. Afterwards, the primary analyst discussed the insights collected from these individual meetings to finalise the definitions for the values analysis in the main study (see Section 3.2.2.2).

3.2.2.2 Pilot Results

The pilot study uncovered two perspectives on human values analysis. The first perspective concerned the (social) interaction between contributors (i.e. *contributor-to-contributor*). For instance, a contributor was *grateful* to another contributor who guided them to solve an issue: '*Thanks for all the effort!*'. The second perspective concerned how the application supported or violated users' values (i.e. *application-to-user*). This analysis perspective could come from developers or end users of the application. For example, a user expressed their anger when a malfunction of the app cost him/her (i.e. *wealth* value): '*This app is heavy on my data-bill*'. In another example, developers expressed their concerns about a possible privacy breach in the application: '*It's bad enough that [the app] uses phone numbers plus SMS for signup as opposed to usernames plus proof-of-work*'. Although the contributor-to-contributor perspective was also interesting, the analysts agreed to focus on the application-to-user perspective for the main study.

The pilot study led to the following observations: first, all analysts were comfortable using terms from Schwartz's models for some themes related to Schwartz's values. Second, when all themes were compared, some proposed themes had similar descriptions but were labelled with different terms by the analysts. Those terms were calibrated using the descriptions and examples provided by the analysts. A consensus name was then reached for those similar terms. Table 3.3 shows the themes discovered by the analysts, the mapping between the potentially similar themes, and the proposed name for each value. For example, there was the term *'convenience'* from a secondary analyst (SA4), which was later refined into two different terms: *usability* and *efficiency*. Discussions revealed that for some cases, *'convenience'* referred to how easy the app was for users, whereas in another case, it referred to how quickly users could perform tasks in the app. This list of themes served as an initial set of value themes for the main study.

The third observation concerned how to determine whether a theme could be considered as human values. Some emerging themes, such as *efficiency* and *accessibility*, were relevant to software engineering concepts (e.g. NFRs) but could not be easily linked to Schwartz's values. To address this, definitions were developed during the discussion as the criteria for identifying human values. The discussion suggested referring a theme as a value theme by the following definitions:

Definition 3.1 (Value Theme)**.** A theme that emerges from issue discussions can be considered a value theme if it captures a response of dismay or appreciation by a contributor towards the app.

This definition comes from a characteristic of human values: *'when values are activated, they become infused with feeling such that people feel aroused when their values are*

TABLE 3.3: The mapping of values discovered in the pilot study
PA: Primary Analyst; SA#: Secondary Analyst

PA	SA1	SA2	SA3	SA4	SA5	Proposed
Accordance	-	-	Conformity-Rules	-	-	Conformity
Pleasure	-	-	Pleasure	-	Pleasure	Pleasure
-	-	-	Universalism-Tolerance	-	-	Inclusiveness
Freedom	Freedom	-	-	Freedom	-	Freedom
-	-	-	-	-	Independence	Independence
Financial	-	-	-	-	Wealth	Wealth
Privacy	Privacy	-	-	-	Privacy	Privacy
Security	Security	-	Security	Security	Security	Security
Secrecy	-	-	-	-	-	Secrecy
Trust	-	-	-	-	-	Trust
-	-	-	-	-	Accuracy	Correctness
Compatibility	-	-	-	-	-	Compatibility
Financial	-	Sustainability	-	-	-	Longevity
Efficiency	-	-	-	Convenience	Intelligent	Efficiency
Usability	Usability	-	Usability	Convenience	-	Usability
Accessibility	-	-	Universalism-Concern	-	-	Accessibility
Transparency	Transparency	Transparency	-	-	-	Transparency
-	-	-	Humility	-	-	Humility
-	-	-	-	-	Health	Health

threatened' [1]. This definition is also related to previous literature that proposed, '*emotion is a source for values importance*' [114]. As an example for Definition 3.1, a contributor reported that '*... a mobile search (without server) using a single word query on 200 email[s] takes **more than a minute** and k9 freezes and screen goes black, ... [This] is **extremely annoying***'. In this example, the contributor valued *efficiency* such that the slow search annoyed him/her. Although the concept of efficiency is related to NFRs (see Section 3.4.4), this study included it, as it satisfied the value theme definition. Subsequently, to determine whether a value theme was considered present in issue discussions, the following definition was developed:

Definition 3.2 (Value Theme Presence)**.** A value theme is present when a theme from Definition 3.1 is discovered in issue discussions.

For example, a contributor posted their opinion: '*I **dislike** this because it is **reliant** on additional apps and resources*'. Based on Definition 3.2, a value theme is present in this phrase because the value theme of *independence* can be found in the phrase. At this point, some of the value themes from the pilot study had a similar concept in Schwartz's model. Some others were closely related to software engineering. To illustrate the extent to which value themes could be related to human values, the main study proposed mapping these themes to Schwartz's values. The two definitions developed in this pilot were later used in the main study as criteria to identify the presence of human values in issue discussions.

3.2.3 Main Study

The main study analysed issue discussions for values using the definitions developed in the pilot study. It also used themes found during the pilot study as an initial set of value themes (see Table 3.3) but still allowed new value themes to emerge from the issues. The main study included the third application, Focus, which has different functionalities from Signal and K-9. This inclusion was designed to enable the understanding of whether the functionalities of an application influence the value themes discovered.

The main study consisted of three phases. The first phase sampled issues from the three applications to be analysed. The second phase involved an analysis of issue discussions to address what values were discovered (RQ1.1). The third phase investigated the presence of human values in different applications (RQ1.2). The following subsections explain the details of each phase.

3.2.3.1 Issues Sampling

This step involved randomly sampled Signal, K-9, and Focus issues that had been collected from each application's repository (Table 3.2). This step resulted in 1,097 issues consisting of 5,615 posts from the three projects to obtain a significant number of samples. This number of issues corresponded to a sample that gave a 95% confidence level with a 5% margin of error. Table 3.4 shows the number of issues and posts from each project for the main study.

3.2.3.2 Analysis for RQ1.1

To answer RQ1.1 (i.e. *what values are discovered in issue discussions*), a qualitative analysis was conducted using open coding with constant comparisons. Using the constant comparison method, newly found themes were constantly compared with previously found themes [115]. The technique was used to refine the descriptions and ensure

Project	# Sampled Issues	# Sampled Posts
Signal	364	2,233
K-9	333	1,541
Focus	400	1,841
Total	**1,097**	**5,615**

TABLE 3.4: The number of issues for the main study

the consistency of the value themes. The findings of the pilot study suggested that a value theme could be closely related to software engineering (see Section 3.2.2.2). To illustrate the extent to which these themes were related to human values, the themes in the empirical analysis were mapped to Schwartz's refined value model. Figure 3.5 shows the relation between Schwartz's values and the results of the main study.

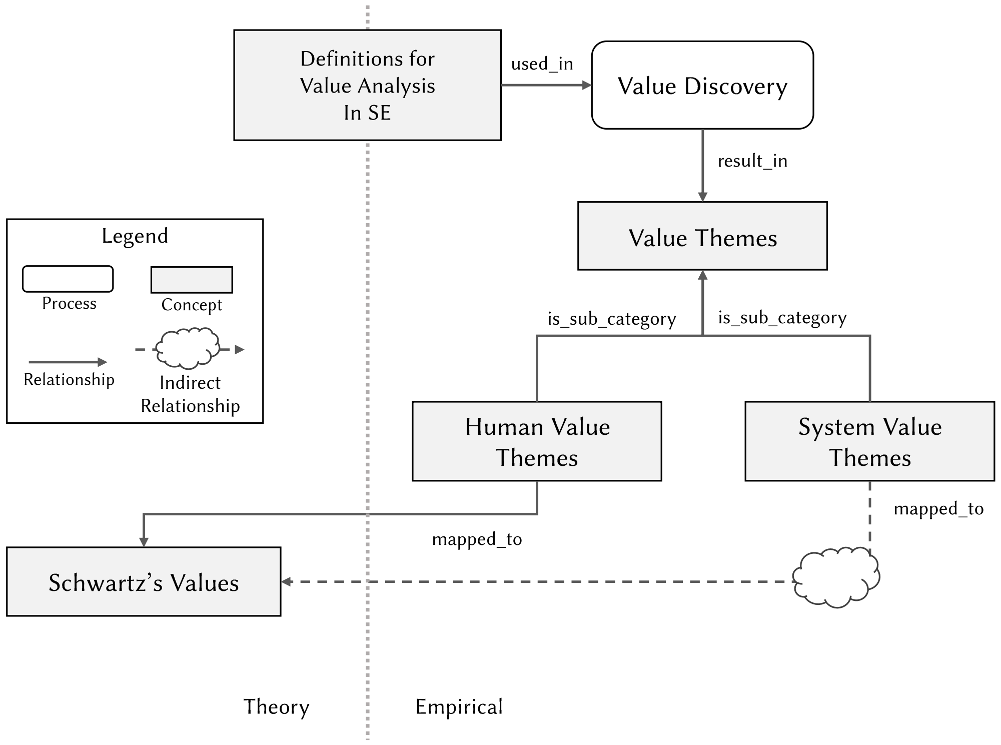


FIGURE 3.5: The relation between the main study and Schwartz's values

During the qualitative analysis, there was a limitation in terms of the availability of the analysts. To overcome this, the analysis process was divided into two phases: values discovery followed by validation and adjustment. The primary analyst conducted the values discovery phase in the pilot study. Then, three validators were employed to validate a subset of the primary analyst's results in the following phase. The details of each phase are as follows.

Discovery of values: The primary analyst analysed the sampled issue discussions for value themes. The primary analyst was allowed to propose any value themes without strictly referring to the initial set of themes from the pilot study. The analyst formulated a description from the findings in the issue discussions for each value theme found. The main study did not limit the number of value themes discovered in an issue discussion. Although the primary analyst needed to consider an issue as a whole unit, the analyst also indicated in which post within the issue human values were found. It took approximately two months for the primary analyst to complete this phase.

Validation, adjustment, and mapping of values: Once the values discovery phase was finished, a validation was conducted on the analysis results. The validation involved three validators. The validators ranged from research fellows to a professor. All of them had backgrounds in software engineering. Two of the validators were involved in the pilot study. A new validator was introduced to address possible bias in the validation due to previous involvement in the pilot study. Before the validation, the new validator was also asked to study literature on human values.

To obtain a significant sample for the validation, a random sample of the issues analysed in the previous step was selected using a confidence level of 95% with a 5% margin of error resulting in 349 issues consisting of 1,976 posts. The first, second and third validators were allocated 150, 100, and 99 issues, respectively. This allocation scheme was based on the availability of each validator. The validators were asked to review each assigned issue. They had to indicate whether they agreed or disagreed with the primary analyst's analysis and provide reasons. The validators required two weeks to complete the validation.

At the end of the validation process, the primary analyst met with each validator. Before the meeting, the agreement between the validators and the analyst was 72% with a Cohen's Kappa of 0.47 (i.e. moderate agreement). After the meeting, the agreement reached 89% with a Cohen's Kappa of 0.78 (i.e. substantial agreement). The primary sources of disagreement were as follows:

- *Refinements to the definitions of the emerging values.* The validators proposed several refinements to the description of emerging values based on their findings. For example, a validator found a conversation in issue discussions indicating that users still needed help from a contributor to configure the app if they changed their phone. A validator was then proposed to broaden the description of *independence* by adding *relying on someone else* to cover the issue. The analyst and the validators agreed on these refinements.

- *Different inference process in the analysis.* A validator proposed to include themes relevant to human values that did not directly fit the definition of value theme *presence* (Definition 3.2), i.e. that required some steps of inference. For example, a user reported that in Android Auto, Signal displayed only the latest message from a contact, not including all previous messages as in regular Android phones. The validator inferred that a value theme of *helpful* was present in this issue based on the validator's reasoning that displaying all messages from a contact is more helpful to a user. The analyst and the validators could not agree to resolve this difference. For this case, the analysis of the primary analyst was chosen.

Once the validation was completed, the primary analyst updated the final labelling for 349 issues. The primary analyst then reviewed the remaining 748 issues and made adjustments based on the insights from the validation phase. It took approximately three weeks for the primary analyst to adjust the dataset.

After the adjustment, the analysis continued with proposing potential mapping between value themes discovered and Schwartz's Values. For this purpose, the primary analyst reviewed each value theme, its description, and its findings on the issues and mapped it to the list of values of the Schwartz model [28]. The primary analyst then discussed the mapping results with two validators, who were also involved in the previous validation process. The discussions resulted in two categories of value themes based on whether the themes could be directly mapped to Schwartz values, namely human value themes and system value themes (see Figure 3.5).

Table 3.5 shows some examples of the analysis of values in issues discussions and their mapping process to human value themes and system value themes. As an example of the process for the human value themes, on the left side of Table 3.5, a user shows concern for *inclusiveness* to support users with different cultural backgrounds. This inclusiveness theme was mapped to ***universalism–concern*** because it is directly aligned with the idea of equality for all users regardless of their origin. On the other hand, system value themes required additional interpretation or assumption to be mapped to Schwartz values. For example, on the right side of Table 3.5, some contributors complained that the application could not work on their specific devices. These issues

Table 3.5: Examples of Value Analysis in Issue Discussions and their Mapping Process to Human and System Value Theme

	Human Value Theme	***System Value Theme***
Issue Excerpt	*'Please implement the rtl [Right to Left] layout for ... language like Persian, Arabic, etc. This is needed for the general UI of the app and also header and body sections of the emails.'*	*'Moreover, often the audio call quality degrades for some seconds. It always happens. Verified on 5 Android phones so far.'*
Code	Supporting diverse languages	Not working on some phones
Value Theme (Description)	Inclusiveness (the software should support foreign languages)	Compatibility (the software should supports user devices)
Mapping Process / **Mapping Process (interpretation)**	Supporting users ***equally***, regardless of ***their languages***	***Assuming devices as material resources***, supporting them ***enables users*** to use the software
Schwartz's Values (Description)	Universalism—concern (Commitment to ***equality***, justice and protection for ***all people***)	Power—resources (***Power*** through control of ***material*** and social ***resources***)

were coded as compatibility because, although they are more technical, users consider them important enough to raise issues. As it is more technical, it is hard to directly map this theme to the Schwartz values. However, because the theme of compatibility was mainly related to devices, the analysis assumed that these devices could be considered as materials or resources. Thus, the compatibility theme was assigned indirectly to ***power—resources***, because supporting users' devices allows users to use the software.

3.2.3.3 Analysis for RQ1.2

The analysis for RQ1.2 used the aggregated values annotation on the issue level from RQ1.1. To answer the first part of RQ1.2, that is, *does the presence of values differ across projects*, a quantitative analysis was carried out to obtain the number of issues where values were present in each project. Because each project had a different number of analysed issues (see Table 3.4), the analysis used the percentage of issues where values were present in each project instead of the number of issues.

To answer RQ1.2, the analysis first identified and compared the values that were more prevalent in each project. Afterwards, the analysis compared the presence of values across projects. To this end, the analysis identified which values had a substantial difference in presence between projects. This was followed by a qualitative analysis to comprehend the nature of their corresponding issue discussions and whether they were different between projects.

3.3 Results

This section describes the results of both the qualitative and quantitative analyses in the main study. The section first presents the value themes discovered and, then, describes the number of value themes found across projects. Finally, the presence of human values based on the proposed mapping is presented to illustrate the relation between value themes and human values.

3.3.1 Value Themes Discovered (RQ1.1)

The analysis of three applications (Signal, K-9, and Focus) discovered 20 value themes. Fifteen of the themes were the same as the value themes identified in the pilot study. The remaining five themes emerged in the main study. The mapping of these 20 themes to Schwartz's values resulted in 10 of them could be mapped directly to Schwartz's values (i.e. *human value themes*). Meanwhile, the other 10 themes were technical and required an extra step of inference to Schwartz's values or could not be easily mapped (see Section 3.2.2.2). These themes were referred to as *system value themes*, as they were important for users when using the application (or system-human interaction). These two themes are explained in the next subsections.

3.3.1.1 Human Value Themes

Ten discovered themes could be mapped to Schwartz's values. Table 3.6 presents these themes, their description, and their mapping to Schwartz's refined theory of basic values [28]. The description of each theme was conceptualised from the issue discussion and represented how its corresponding human value could be manifested in software development. *Conformity* was about how an application follows rules, regulations, guidelines, or standard practices. For example, a Signal user complained about the displayed text that did not align with the Android design guidelines. This theme could be mapped to ***conformity—rules*** in Schwartz's model. *Pleasure* promoted the enjoyment of using the application. For instance, a contributor believed that emojis in Signal were amusing: *'I do not see what you are complaining about, it [emojis] is a fun feature'*. *Pleasure* has a similar concept to **Hedonism** in Schwartz's values. *Dignity*, which could be mapped to **face**, was about maintaining respect and honour of the users while using the application. For example, a user reported *'I have been in a huge fight with the Vodafone customer service, called them at least ten times, have ruined their shop over here in Germany, write multiple heavy complaints to them and was really upset - until my girlfriend noticed that this Issue appeared since we have been using [the application]'*. *Inclusiveness* ensured people with diverse backgrounds could use the application. This theme could be mapped to ***universalism—concern***. Inclusiveness in an application could manifest itself as user interface translation or other region-specific settings such as a right-to-left layout to support the Arabic language. *Sense of belonging* was about how users could connect and interact with their peers through the application. Losing these abilities could make users anxious or uncertain about their social circles. In this way, this theme could be mapped to ***security—personal*** as this Schwartz value is related to users' social circles as their immediate environment. As an example of this theme, a Signal user complained *'I have been invited to a group of friends, and I have not received invite or messages'*.

The themes *freedom* and *independence* could both be mapped to ***self-direction—action***, but they were somewhat different. Freedom is about providing options for users. The following post, *'people are being forced to register with TextSecure [Signal] Message Server based in the USA'*, demonstrated that a lack of options becomes a threat to freedom. However, independence emphasises not being influenced by or dependent on any third parties. As an example, a contributor of K-9 disliked the idea to add third-party spam filtering because *'it is reliant on additional apps and resources'*. The value theme of *wealth* could be mapped to ***power—resources***. It related to the

TABLE 3.6: Human value themes discovered in issue discussions
(*): mapping result of the value theme to Schwartz's refined theory of basic values [28]

No.	Value Theme	Human Value*	Description
1	Conformity	Conformity—rules	Follow rules, regulations, guidelines, or standard practices during the application development
2	Pleasure	Hedonism	Promote enjoyment and satisfaction while using the application
3	Dignity	Face	Maintain the honour and respect for users
4	Inclusiveness	Universalism—concern	Facilitate different origins, languages, cultures, and level of knowledge in the application
5	Sense of belonging	Security—personal	Connection and interaction between users
6	Freedom	Self-Direction—action	Options for user preferences
7	Independence	Self-Direction—action	Little or no reliance on any third parties in the use of an application
8	Wealth	Power—resources	Monetary costs of using the application
9	Privacy	Self-Direction—action	Protection of personal information
10	Security	Security—personal	Protection against threats and attacks from foreign entities

monetary cost needed to use an application. For instance, a user was found to have complained about the way K-9 fetching emails could be *'heavy on [his/her] data bill'*. The last theme in this category, *privacy*, could be mapped to ***self-direction—action***. Privacy, which is about protecting personal information, has also become the centre of attention in software development lately. For example, K-9's contributors discussed the possibility of supporting network proxy to enhance users' privacy. *Security* was concerned with protecting users against threats or attacks from foreign entities. This security description is more technical and slightly more narrow than a similar concept in Schwartz's theory that defines security as *'avoiding danger and feeling cared about'* (i.e. personal security) and *'social order and government stability'* (i.e. societal security) [28]. In this case, the security theme found in the analysis was more suitable to be mapped to ***security—personal***.

Based on the mapping of these human value themes to Schwartz's refined model [28] in Table 3.6, 7 out of 19 values were discovered. Two of the seven values, namely

security—personal and ***self-direction—action***, corresponded to more than one discovered theme. ***Security—personal*** corresponded with security and sense of belonging. Meanwhile, ***self-direction—action*** was found to correspond with freedom, independence, and privacy. The remaining human values corresponded with one discovered theme each.

3.3.1.2 System Value Themes

Users valued 10 themes, that required an extra inference to be mapped to Schwartz's values. This study placed them into a different category and referred to them as system value themes. This name was chosen for the 10 themes because they were considered important by the contributors to be present in the system. Table 3.7 presents the system value themes discovered, along with their descriptions.

TABLE 3.7: System value themes discovered in issue discussions

No.	Value Theme	Description
1	Trust	Compelling users that an application is harmless
2	Correctness	The application provides expected information and behaviour
3	Compatibility	Supports a range of user devices and systems
4	Portability	Migratable to other devices
5	Reliability	Fewer occurrences of application failures
6	Efficiency	Less time and effort when conducting tasks in an application
7	Energy Preservation	Less use of power or energy
8	Usability	Ease of use of the application
9	Accessibility	Ease of access in regards to information or functionalities in the application, especially for users with special needs
10	Longevity	Prolonged availability of the application in the future

Trust indicated how an application convinces users that it is harmless. For instance, contributors of K-9 raised their concern in the following post: *'if one does not trust K-9 to behave, there is no hope'*. *Correctness*, *compatibility*, *portability*, *reliability*, and *efficiency* were related to the quality attributes of software [53]. The analysis found evidence that these themes could affect users. *Correctness* aligned information or behaviour of the app with users' expectations. For instance, Signal's users reported that pressing their own message showed the recipient's number instead of their number. *Compatibility* related to the support of an app in various devices and internal systems. For example, some users of K-9 reported application crashes after they updated their systems. *Portability* allowed users to migrate their data and configurations in the app from one device to another. For instance, the following comment in a Signal discussion reflected a case where users could not change their device to use the application: *'Currently there is no way to change the phone without losing existing data'*. *Reliability* concerned the stability of the app during use. For example, a lack of reliability encouraged a user to report, *'Unfortunately, that is all I have... :/ I tested a bit, and it seems to crash at random positions. Not the same post every time, not the same type of post'*. *Efficiency* could be associated with the time and effort needed to perform tasks in the app. For example, a user of Signal pointed out how the low performance of the app could discourage users: *'... group chat isn't even that fast, and this is discouraging my friends a lot'*. *Energy preservation* was similar to *efficiency* but related to energy usage. The analysis separated *energy preservation* because many discussions demonstrated users' concerns about how the app drained their devices' power. For instance, a K-9 user reported how the application was '*eating*' his battery when there was no network connection.

Accessibility and *usability* both related to how easily users could use the application. The difference was that *accessibility* was related to accessing information and features, especially for disabled users. The lack of accessibility support in the app made a K-9 user mention, *'It probably also makes the app much harder or even impossible to use with accessibility services'*. Meanwhile, *usability* was concerned with the ease of use of the app. For instance, a K-9 user complained that the *'mail view should remember the scrolled position ... This is a tiny usability thing, but I think I would make a big difference'*. Lastly, *longevity* ensured the future development and use of the app. For example, a contributor reminded the forum, *'**Be aware**, that 95% of other users will have lost their backups because you removed support for external SD cards.'*

As explained above, these system value themes could not be easily mapped to values in Schwartz's model. Nevertheless, based on the empirical findings, an attempt was conducted to map these themes to Schwartz's values. This mapping illustrated the possibilities of mapping technical themes to human values. Table 3.8 shows the proposed mapping between the system value themes and Schwartz's values. The theme of *trust* could be loosely mapped to **conformity—interpersonal** with the similar objective

Table 3.8: The mapping between system value themes and human values in the context of the three projects

No.	Value Theme	Human Values
1	Trust	Conformity—interpersonal
2	Correctness	–
3	Compatibility	Power—resources
4	Portability	Self-Direction—action, Power—resources
5	Reliability	–
6	Efficiency	Power—resources
7	Energy Preservation	Power—resources
8	Usability	Benevolence—caring
9	Accessibility	Benevolence—caring, Universalism—concern
10	Longevity	Security—personal

of avoiding harm to others. The *compatibility* theme could be mapped to ***power—resources*** in terms of how devices (i.e. resources) were supported by the application. The *portability* theme could be mapped to ***self-direction—action*** in terms of how the application should allow users to migrate to other devices. *Portability* could also be mapped to ***power—resources*** in terms of its relation with devices as the resource. The themes of *efficiency* and *energy preservation* could both be mapped to ***power—resources***. Both themes were related to resources management, namely time and energy. The themes of *usability* and *accessibility* could be mapped to ***benevolence—caring*** because they were about supporting users in using the application. Additionally, accessibility could also be mapped to ***universalism—concern*** because this theme also aimed to support users with special needs. *Longevity* could be mapped to ***security—personal*** because it was related to the assurance to users that the application would be available in the future. Any suitable mapping from Schwartz's values was not found for the *correctness* and *reliability* themes. This mapping between system value themes and human values was limited to the context of the three case study projects.

The analysis discovered 20 value themes in issue discussions and described each of them. The themes could be broadly classified into two groups:

(a) Ten value themes that have a corresponding concept in Schwartz's refined theory of basic values: *conformity*, *pleasure*, *dignity*, *inclusiveness*, *sense of belonging*, *freedom*, *independence*, *wealth*, *privacy*, and *security*

(b) Ten other value themes that could not be directly mapped to Schwartz's values: *trust*, *correctness*, *compatibility*, *portability*, *reliability*, *efficiency*, *energy preservation*, *usability*, *accessibility*, and *longevity*

The analysis also found 7 out of 19 values from Schwartz's refined model corresponded to the discovered human value themes, namely *self-direction—action*, *security—personal*, *hedonism*, *universalism—concern*, *power—resources*, *conformity—rules*, and *face*.

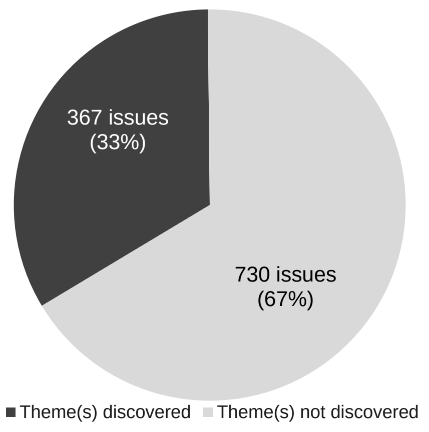
3.3.2 The Presence and Variance of Values Across Projects (RQ1.2)

This subsection describes the comparison of the presence of human values across projects. First, it describes a comparison of the presence of the value themes in all projects. Second, based on the mapping of the value themes to the human values (Table 3.6), a comparison of the presence of human values across these three projects is presented.

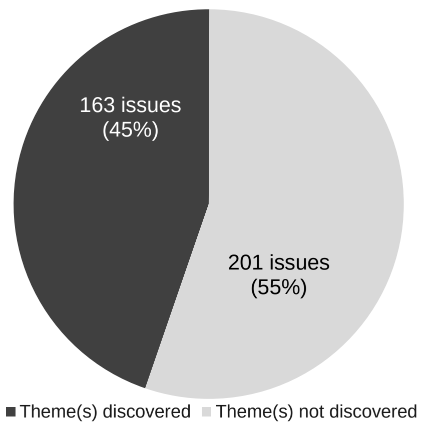
3.3.2.1 Value Themes Across Projects

Figure 3.6 shows the prevalence of value themes in the three projects. As shown in Figure 3.6a, value themes were present in 33% of the sampled issues constructed from all three projects. Breaking it down to each project, Signal had the most issues where value themes were present: 163 out of 364 issues, or 45% (Figure 3.6b). It was followed by K-9, where 31% of issues (102 out of 333 issues) contained value themes (Figure 3.6c). The project with the least presence of value themes was Focus with 25%, or 102 out of 400 issues (Figure 3.6d).

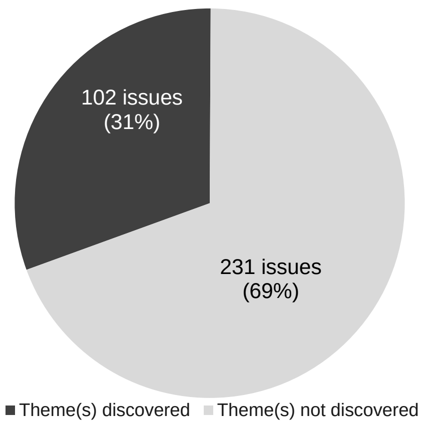
Table 3.9 shows the presence of value themes in each project. For human value themes, in Signal, the top three most prevalent themes were *privacy*, *security*, and *freedom*. Aside from these three themes, the remaining human value themes were relatively higher than they were in other projects. The top three most prevalent human value



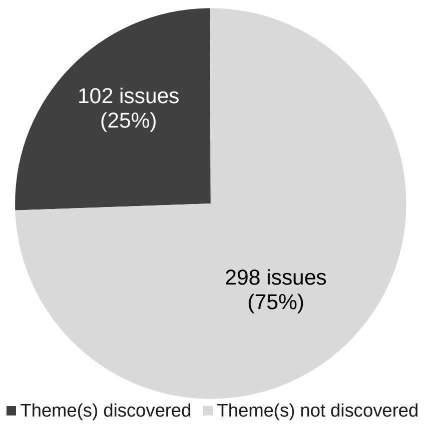
(A) All three projects



(B) Signal



(C) K-9



(D) Focus

FIGURE 3.6: The presence of values in issue discussions

themes in K-9 were similar to those in Signal. Meanwhile, Focus had similar top three most prevalent themes to Signal and K-9; however, *inclusiveness* was more prevalent than *security*. The theme of *Sense of belonging* was not present in K-9. This theme, along with *dignity*, were not present in Focus at all.

For system value themes, Table 3.9 shows that in Signal, the top three most prevalent themes were *usability*, *efficiency*, and *correctness*. The top three most prevalent system value themes in K-9 were similar to the themes in Signal, except for *energy preservation*, which was more prevalent than *correctness*. *Longevity* was not found in K-9. Focus also had similar prevalent system value themes to the other two projects, except for *accessibility*, which was more prevalent after *usability* and *efficiency*. Unlike the other two projects, the theme of *portability* was not found in Focus.

TABLE 3.9: Percentage of number of issues where value themes are present
Asterisk (*) denotes top three most prevalent themes of each category in each project

Category	Value Theme	Signal	K-9	Focus
Human Value Theme	Conformity	2.20	0.90	2.00
	Pleasure	3.57	0.30	0.25
	Dignity	1.37	0.90	0
	Inclusiveness	2.75	1.20	*3.00
	Sense of belonging	3.30	0	0
	Freedom	*6.32	*2.40	*4.00
	Independence	2.47	1.20	2.25
	Wealth	2.47	0.60	0.50
	Privacy	*9.07	*2.10	*8.75
	Security	*4.95	*6.61	2.25
System Value Theme	Trust	1.37	0.30	1.00
	Correctness	*2.20	0.90	0.50
	Compatibility	1.37	0.90	0.25
	Portability	1.10	0.30	0
	Reliability	0.27	0.30	0.50
	Efficiency	*7.42	*5.11	*3.75
	Energy Preservation	1.65	*2.10	0.75
	Usability	*13.19	*11.71	*8.75
	Accessibility	1.10	0.60	*1.75
	Longevity	1.10	0	0.25

Figure 3.7 makes it clear that the presence of *privacy* in Signal and Focus was substantially higher than it was in K-9. The findings showed that Signal and Focus contributors gave more attention to privacy-related functionalities provided by the applications, such as private messaging in Signal or private browsing in Focus. For example, a contributor of Focus remarked how an implemented functionality might not be appropriate: '*... something that feels less appropriate for* ***a privacy-centred app***'. This finding was aligned with the strong statements of privacy from both Signal and Focus.

In addition to *privacy*, Figure 3.7 shows that *pleasure* also had a substantially higher presence in Signal than in K-9 and Focus. The analysis found that users of Signal caused this, as they demanded the application be enjoyable to use. Regarding this, many requests were found that related to the application's look and feel (e.g. themes and colours) and support for individual expressions (e.g. multimedia and emojis). For instance, a Signal user suggested having the functionality to colour incoming messages based on their authors for the following reason: '*People love to customise their stuff, and many are [quite]* ***emotional*** *regarding colours*'.

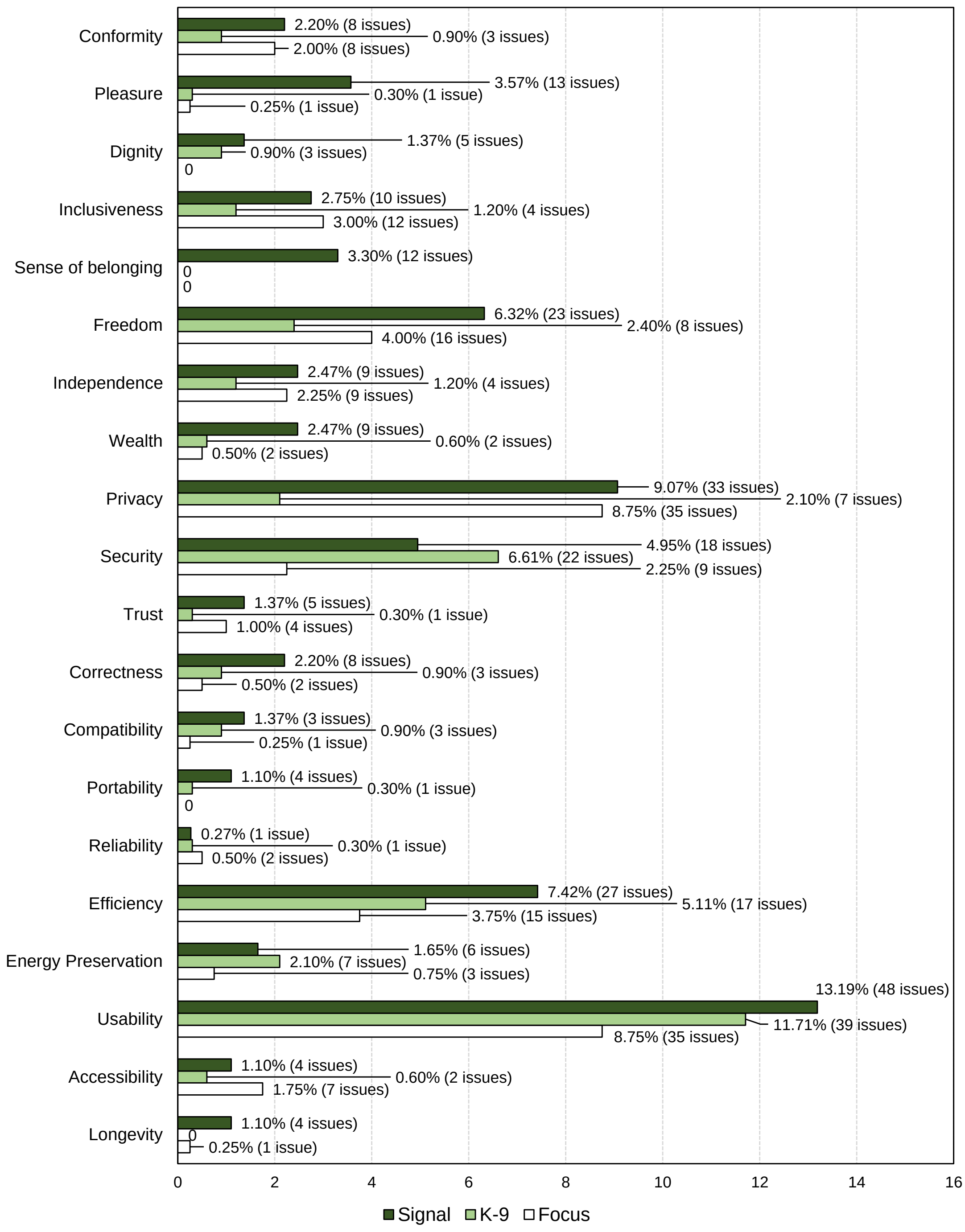


FIGURE 3.7: Percentage (%) of number of issues where values are present in the projects

Sense of belonging was only discovered in Signal. The analysis indicated that all discussions about this value came from the messaging functionality, especially group messaging, which became personal to users. For example, a user expressed his/her dissatisfaction: *'I am just **dismayed** at the effort I put into pushing my whole family & all my friends to use it [Signal] & now I cannot keep it working'*.

Similar to *sense of belonging*, some themes were also not discovered in K-9 (e.g. *longevity*) or Focus (e.g. *dignity*, *portability*, and *secrecy*). However, the presence of these themes was low (below 2%) in the other projects. The remaining themes' presence was not substantially different (less than 2% difference) across the three projects.

3.3.2.2 Human Values Across Projects

The presence of seven values from Schwartz's refined model based on the mapping of the human value themes is shown in Table 3.10. This table did not include the illustration of the mapping between the system value themes and human values in Table 3.8. ***self-direction—action*** was prevalent in both Signal and Focus. The analysis found that the discussions on the issues of Signal and Focus were mainly about privacy and freedom-related functionalities provided in both applications. Meanwhile, the most prevalent value present in K-9 was ***security—personal***. This finding was in line with the focus of K-9 as a secure mail client application.

Table 3.10: Percentage of human values occurrence

No.	Human Value	Signal	K-9	Focus
1	Self-Direction—action	14.56	4.20	11.00
2	Security—personal	7.97	6.61	2.25
3	Hedonism	3.57	0.30	0.25
4	Universalism—concern	2.75	1.20	3.00
5	Power—resources	2.47	0.60	0.50
6	Conformity—rules	2.20	0.90	2.00
7	Face	1.37	0.90	0.00

The presence of other values in Signal was relatively high (more than 1%). Meanwhile, the values that were present in more than 1% of K-9 issues were ***security—personal***, ***self-direction—action***, and ***universalism—concern***. In Focus, values that were present in more than 1% of the issues were *self-direction—action*, *universalism—concern*, *security—personal*, and *conformity—rules*. The value of ***face*** was not found in Focus because Focus is more general and less personal (e.g. not used to communicate with other people).

The analysis showed that almost one-third of the studied issues in the Signal, K-9, and Focus applications included discussions where value themes could be found. The common prevalent human value themes across projects were *privacy* and *freedom*, while the common prevalent system value themes were *usability* and *efficiency*. However, the value themes of *privacy*, *pleasure*, and *sense of belonging* were not evenly present across these apps, which could likely be attributed to the apps' values statements and functionalities. Based on the mapping of human value themes to a human value model, *self-direction—action* and *security—personal* were found to be the most prevalent values in the three applications.

3.4 Discussion

This section highlights and discusses the findings from the issue discussions. This section also discusses the possibility of developing human values definitions for software engineering and highlights the need to develop an automated detection tool.

3.4.1 Human Values are Present in Issue Discussions

The presence of values in design artefacts has been theorised in general. The results of this study contribute empirical evidence that human values are indeed found in software development repositories and demonstrate which values can be found in issue discussions. The pilot study identified two distinctive perspectives of value themes from issue discussions: *contributor-to-contributor* and *app-to user* (see Section 3.2.2.2). This study focused on the *application-to-user* perspective by analysing project contributors' concerns and opinions as found in issue discussions. This perspective allowed for understanding of how and to what extent project contributors are concerned about human values while developing software applications. Using this approach, the study discovered 20 value themes and formulated their descriptions based on a qualitative analysis of 5,615 posts from three projects. The examples found in the issue discussions for the themes were similar to the concept of *value instantiation* [43, 114, 116]. Value instantiation is defined as *'the representation of values in specific situations, issues, or behaviours'* [43, 114]. In this case, the discussion of issues provided a representation of human values in software engineering contexts.

Among the 20 value themes, 10 of them could be mapped to 7 out of 19 values from Schwartz's refined model. The other 10 themes were technical, such that mapping these themes to human values required an extra step of inference. These findings confirmed the much earlier opinion that technological artefacts could contain human values [17]. Specifically, these findings were in line with previous works [18–26] that discovered values in software development artefacts.

3.4.2 The Prevalence of Human Values in Software Engineering

The refined Schwartz's model [28], the most widely used model in social sciences [1], includes 19 values. In this study, only 7 of those 19 values were found in the issue discussions (Table 3.10). This result was in line with previous work [10], which discovered that 60% of Schwartz's value items (35 out of 58 value items) were not found in software engineering publications. This could mean some values are still under-represented in both research and practice, and thus, further research is needed. For example, further research could investigate the reasons behind this under-representation (e.g. software practitioners are unfamiliar with these values). Another line of research could investigate how to support these values in software engineering, e.g. provide a contextualised software engineering definition for the under-represented values.

3.4.3 Definitions of Human Values in Software Engineering

To the best of our knowledge, there are no human value models developed exclusively for software engineering. Notable work in the field of HCI called value-sensitive design defines a subset of values that have ethical imports [13] that could be somehow transferred into software engineering. However, the social sciences' definition of human values is much broader, including *'what an individual, group or wider society believes to be important'* [50], such as *public image* or *achievement*. The lack of practical definitions for the broader human values in the SE context was argued by Mougouei et al. [7] to be one reason why software engineering practices have not been entirely successful in integrating human values into the software. Previous works have attempted to address this issue using existing models from the social sciences as a starting point. For example, some previous works chose to use Schwartz's model [7–10, 12] because of its extensiveness in comparison with other models. However, these models still need to be adapted for the SE context. This adaptation is necessary because of the possible *'differences in the meaning and interpretation of values in different contexts'* [10]. For instance, the findings of *security* in the software engineering context have a limited scope within the app, unlike the definition of security in Schwartz's model, which encompasses the whole society [1]. This study addressed to the lack of practical human values definitions for software engineering by proposing descriptions of human values from issue discussions. Although the descriptions can be developed further, these initial descriptions could empower software practitioners to consider human values in their software engineering practices. For instance, if some users have concerns regarding the independence of an application, the developers may respond by considering to what extent they want to use third-party services. Some of the themes, especially those that are well known in SE, such as *efficiency* or *usability*, may already have been defined in SE. In this case, future work could compare the descriptions from the empirical study with their definitions in the SE literature. Moreover, future research is still necessary to investigate how these definitions can help incorporate human values into software engineering.

3.4.4 Human Values and Non-Functional Requirements

It could be argued that the system value themes (Section 3.3.1.2), such as *efficiency* and *usability*, are similar to NFRs. The NFRs are usually related to the quality properties, characteristics, or attributes of software [53–55]. On the relation between NFRs and human values, Barn [55] mentioned that human values are not referred to in NFRs studies. Barn concluded that *'either values are systematically ignored in the practice of NFRs elicitation or values may not be NFR'* [55]. The definition of the presence of human values (Definition 3.2) used in this study together with study's findings supported the argument that human values are similar to NFRs in terms of the concerns a user or a contributor may have towards an application. However, human values have a much broader sense, which includes non-technical requirements. In other words, NFRs can be seen as a subset of human values. In this sense, this thesis agrees with Barn's proposal of utilising NFRs frameworks to help integrate values into software [55].

3.4.5 Factors that Influence the Presence of Values

This empirical study found that 33% of the discussions investigated in the three projects include at least one value theme. Value themes appeared the most in Signal, followed by K-9, and then Focus. The findings showed that at least two factors could explain these differences. First, the functionality provided by an application could inspire discussions about a specific value theme. For instance, functionalities such as voice calls and group chats in Signal could trigger the discussions of *pleasure* and *sense of belonging*. This finding was in line with a previous study [117] that argued that different categories of applications have different types of NFRs. However, this also means that the list of discovered value themes may not be exhaustive. There might be other value themes discussed during software development that this study did not discover but that are present in other projects that have different functionalities. For instance, this study did not find any themes that corresponded to some values in the Schwartz models (Figure 2.2), such as *stimulation* or *achievement*, which may appear in different software categories, such as computer games or educational applications [118–120].

Second, the difference in the presence of values is the emphasis on specific values (i.e. *values statement*) supported by an application. The findings in this case study showed that this emphasis on values instigates a higher expectation of the values from the application's users and contributors. For example, it was discovered that Signal and Focus contributors frequently attempted to raise privacy concerns in the discussions (see Section 3.3.2.2). These findings indicated that an application's functionalities and values statements can influence the values present in the discussions. This thesis argues the need for further research to investigate the dedication to values and functionalities of an application as factors influencing the presence of human values.

3.4.6 Towards the Development of an Automated Detection Tool

The abundance of software development data on GitHub makes complete manual analysis of human values infeasible. Previous works [37, 121] have proposed automated techniques to analyse various aspects of software development from a large amount of GitHub data. To the best of our knowledge, there is no automated tool to analyse the presence of value themes, except for themes well known in software engineering, such as *privacy* [68], *security* [20, 21], or *energy preservation* [25]. This empirical study was used as the foundation to develop automated human values detection techniques (Chapter 4). This thesis argues that AI-based approaches (e.g. machine learning approaches) can be leveraged to detect issue discussions that include human values. Furthermore, such AI-based approaches can identify which human values are present in issue discussions. For this purpose, the manually analysed discussions from this study

[96] could be used as a dataset to develop such automated approaches. The criteria to discover value themes seem to correspond with the relation between human values and sentiment or emotion. For example, the literature on social sciences believes that values are infused with feelings [1] and that *'emotion is a source for values importance'* [114]. For this reason, it is appropriate to suggest using sentiment analysis or the presence of emotions to identify the presence of a particular value. Future work could consider utilising sentiment analysis, as demonstrated in previous work [56, 122–124], to better identify values in discussions.

Automating the discovery of values in issue discussions can help stakeholders on the development side, such as product owners or developers, gain insight into values that are important to users. For software practitioners, an automated tool can inform the value implications that an issue can have. Thus, software practitioners can prioritise issues based on the values discovered. Applying the tool in the repositories will allow product owners to track human values in issues across development phases. This application enables the monitoring of whether values are discussed sufficiently early in development. From a software testing perspective, testers can pay more attention to those issues where values are discussed. These examples show the potential of analysing issue discussions to support human values in software development.

3.5 Threats to Validity

A threat to the **construct validity** came from a definition of human values that has not been tested in software engineering. At the same time, there are also no theoretical considerations to suggest that this definition does not transfer to software engineering. This was mitigated by not limiting the analysts to only considering Schwartz's basic values but also to providing descriptions of how the values look in SE. All persons involved in the analysis had a basic understanding of values theory. Moreover, the values discovered in both the pilot and main study were validated and discussed to ensure there was no divergence of definitions across the analysts.

A threat to the **internal validity** came from the analysis process to discover values and the involvement of a single analyst in the main study. Although this could help increase the consistency of the findings [125], the understanding of values and the subjectivity of the primary analyst may have affected the discovery of values. This issue was mitigated by performing a pilot study using an open coding approach used in other similar work [37, 126] involving the primary analyst and five other analysts. In the main study, the primary analyst used the criteria and considered the initial values from the pilot study. Furthermore, three validators assessed and validated the primary analyst's results. Another possible threat to the **internal validity** may have come from the use of issue discussions as a single data source , which may have made the intentions of the issue reporters unclear in the analysis. This threat was mitigated by analysing the whole thread of the issue as the analysis unit. We argue that the thread of an issue can provide a sufficient amount of information about contributors' concerns. The rationale behind this comes from the nature of issue discussions, where contributors will ask for clarification from the reporter. The analysis used the definitions developed in the pilot study to find the values. This process may present another threat to internal validity because some issues with values might be excluded. This threat was mitigated by using the bottom-up approach when performing the thematic analysis during both the pilot and the main study.

A threat to the **external validity** arose from the limited number of projects we selected. All projects involved had a substantial number of issues, but we accept that the results may not be generalisable to other projects. Due to our limitation, we only analysed a random portion of all issues. We accept the possibility that the results may have differed if all issues were considered. Still, we mitigated this using a statistical technique used in similar work [127–129] to obtain a significant sample size.

3.6 Concluding Remarks

This chapter describes our investigation to address the first research question of this research study: *to what extent are human values present in issue discussions*. A case study of 1,097 issue discussions collected from three Android projects was conducted to understand human values in software development artefacts. The findings showed that value themes could be found in issue discussions (33% of the inspected issues). 20 value themes were discovered that included themes directly corresponding to Schwartz's values (*human value themes*) and themes that could not easily be mapped to Schwartz's values (*system value themes*). The human value themes that we found could be mapped to 7 out of 19 of Schwartz's values. Additionally, we illustrated a mapping of system value themes to Schwartz's refined values in the contexts of three applications. Finally, descriptions for each value theme were proposed to inform and empower software practitioners to integrate human values in software development.

The results of this study showed that human values are present in issue discussions. However, to make the discovery of human values in a large scale feasible, an automated tool is required. Manually labelled human values issue discussions from this study could be used as a dataset for the development of such a tool. The next chapter of this thesis describes the attempt to develop an automated detection tool for human values in issue discussions (Chapter 4).

Chapter 4

Automating the Detection of Human Values in Issue Discussions

Investigating human values in issue discussions provide the perspective of values in software development. However, manual labelling of issues requires considerable effort. For instance, the analysis of the 1,000 issue discussions presented in Chapter 3 took two months to complete. Recent studies have suggested the use of automated labelling approaches to alleviate this problem. However, current studies do not focus on the labelling of human values, nor do they consider their labels to be similar to human values. To this end, this study experimented with four machine learning approaches, namely, support vector machine (SVM), random forest (RF), multi-layer perceptron (MLP), and logistic regression (LR). This study involved several evaluation parameters, such as units of classification, imbalanced dataset handling (or resampling techniques), and feature set. The results showed that implementing MLP using an issue-level classification unit with term frequency-inverse document frequency (TF-IDF) as the feature and applying the undersampling technique offered the best F1 score (0.650) and Matthew's correlation coefficient (MCC) (0.445) performance. The results also revealed that using the issue-level dataset provided better performance than using the post-level dataset. Meanwhile, the effects of the use of resampling techniques and the sentiment scores of the issues as a feature varied depending on each classification method. These results indicated that human values can be automatically detected in issue discussions.

4.1 Introduction

As discussed in Chapter 3, detecting human values in issue discussions has the potential to support development teams in integrating human values in an application. However, detecting human values in issues in practice has its own challenges. The first challenge concerns the unrestricted way of writing an issue. The functionalities of GitHub Issues allow project contributors to report issues related to a software project [94]. Within this functionality, there is virtually no limitation in terms of what contributors can submit. A post in an issue could be a question, a feature proposal, or a bug report [130]. This first post is then followed up with other comments by contributors to form a discussion (i.e. an *issue discussion*). In addition, these discussions used informal language, thereby raising another challenge for analysis.

To address the problems of managing various types of issues, GitHub allows specifying labels on each issue report [131]. These labels are helpful in providing instant ideas about the issues to project maintainers, allowing categorisation of issues, and filtering for important issues [132]. Regarding the labels, GitHub provides nine default labels, namely: *bug*, *documentation*, *duplicate*, *enhancement*, *good first issue*, *help wanted*, *invalid*, *question*, and *wontfix*, for all project repositories [133]. Additionally, GitHub allows project contributors to define their own labels. Recent studies have found that the most commonly used labels in open-source projects are *enhancement*, *bug*, *question*, *feature*, *documentation*, *wontfix*, and *task* [132]. Each label is used to indicate how the issues should be dealt with. For example, the *duplicate* label denotes that similar issues have been reported [133] so that an issue with this label will be referred to the previous similar issue before it was closed. Additionally, some labels, such as *bug*, could be considered more critical to be addressed than others (e.g. *question* or *wontfix*). This approach has the potential to be used to indicate the presence of human values in an issue.

The second challenge of human values analysis comes from the tremendous efforts required to analyse and label the issue discussions. For example, analysing ~1,000 issues took two months (see Section 3.2.3.2). The current labelling mechanism [133] requires project maintainers to manually read through and determine the appropriate labels for each issue. Thus, this process is considered labour-intensive, time-consuming, and error-prone for issue management [130, 131], especially for large projects with numerous issues.

Previous work proposed some automated approaches to alleviate this manual issue labelling problem. For example, machine learning techniques were utilised to distinguish bugs among other issues with an average F1 score of 0.8 in [131]. Another study used an efficient linear model called *fastText* to determine whether an issue was a bug report, an enhancement, or a question with an average F1 score of 0.75 [130]. The use of automated approaches potentially reduces the efforts needed by the development team to label the issues.

To address the first challenge, the study described in this chapter proposed representing human values as labels in issue discussions. The human values label can be given to issues that have human values detected. To address the second challenge, i.e. reducing manual labelling efforts, a few well-known automated approaches were used to identify the presence of human values in issues. These approaches could label the issues with human values without manual labelling. Consequently, a development team would have the information support needed to consider human values during software development, similar to what they have with other labels [132].

Recent studies have investigated the presence of human values in text documents. For instance, the k-nearest neighbours approach was used to label sentences for human values in net neutrality debate documents [60]. Other approaches, such as support vector machines (SVM) [64], latent value model (LVM) [61], and simulated annealing [66, 67], were also used to identify values in the same documents. The performance of human values detection was improved from an F1 score of 0.48 [60] to an F1 score of 0.74 [67]. Although these works did not apply the study in software engineering fields, they demonstrated the possibility to use automated approaches to detect the presence of human values in text documents.

Regarding the detection of human values in software repository artefacts, recent studies have investigated concepts that are similar to human values that are well known in software engineering, such as security [20, 22, 23, 56], privacy [18, 19, 24, 57–59, 68], and energy efficiency [25, 26]. However, these studies focused on this limited number of concepts and did not specifically address them as human values.

This chapter describes the attempt to investigate the extent to which the detection of human values in issue discussions could be automated (**RQ2**). This study used the dataset and definitions from Chapter 3 to evaluate four classification methods to automate the detection of human values in issue discussions. To this end, the evaluation considered four experimental parameters, namely, unit of classification, dataset resampling technique, feature set, and classification method. The main contributions of the study described in this chapter are as follows:

1. This study evaluated automated classification methods to detect the presence of human values in issue discussions.
2. This study identified the effects of using other experimental parameters (i.e. unit of classification, dataset resampling technique, and feature set) on the performance of the detection of human values in issue discussions.

The remainder of this chapter is organised as follows: Section 4.2 presents the methodology used in this chapter. Section 4.3 describes the results of this study. Section 4.4 discusses the results. Threats to validity are discussed in Section 4.5. Finally, Section 4.6 concludes the work presented in this chapter.

4.2 Methodology

To answer the second research question of this research, *to what extent can the detection of human values be automated*, this study formulated the detection of human values as a classification problem. In other words, the detection of human values was performed by classifying whether an issue discussion had human values present. To this end, this study evaluated four well-known classification methods with three additional experimental parameters, namely unit of classification, resampling technique, and feature set.

Because an issue consists of several posts, the classification could be performed on either the whole issue as a unit or just a single post. Therefore, this study evaluated both units of classification (i.e. the whole post and a single post) to determine which one performs better. Furthermore, the dataset from Chapter 3 is imbalanced: human values are present on 33% of the issues (see Figure 3.6). To overcome this, this study evaluated two resampling techniques to handle the imbalanced dataset. These techniques work by resampling the minority cases (e.g. issues where human values are not present) to balance the dataset. The third parameter is related to the features used for the classification. This study evaluated two well-known features in text classification, namely, bag of words (BoW) and TF-IDF. In addition to these features, this study also evaluated whether including sentiment features offers better performance. The reason for using sentiment features came from the previous empirical study that found the presence of human values corresponds to the presence of appreciation or dismay towards an application that could be related to sentiment (see Section 3.4.6). Finally, this study evaluated the performance of four well-known classification methods, namely, support vector machine (SVM), random forest (RF), multi-layer perceptron (MLP), and logistic regression (LR). Section 4.2.4 describes each experimental parameter in more detail. To guide the experiments performed in this study, the following sub-research questions were posed:

RQ2.1 How does a different unit of classification affect the performance of the detection of human values?

RQ2.2 How does the use of resampling techniques affect the performance of the detection of human values?

RQ2.3 How does the use of different feature sets affect the performance of the detection of human values?

RQ2.4 To what extent is the performance of detecting human values influenced by the chosen classification methods?

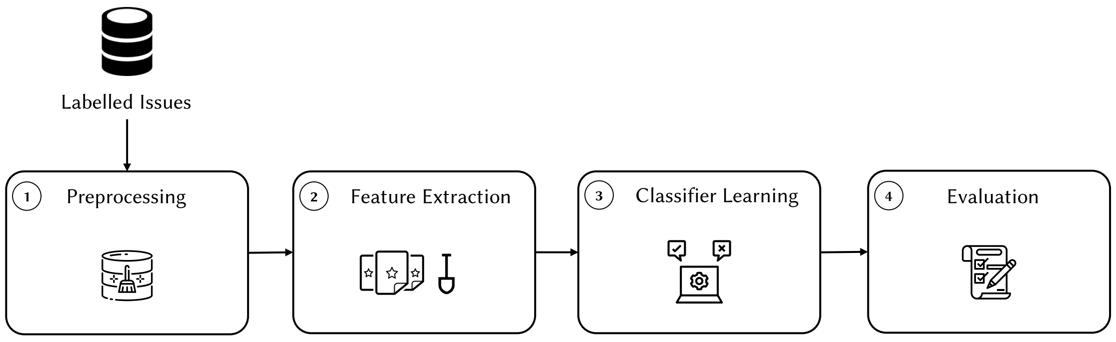


FIGURE 4.1: Methodology to answer RQ2

Figure 4.1 presents the methodology for addressing the sub-research questions. This study used the labelled issues from Chapter 3. In the first step, preprocessing was performed on the labelled issues. Then, the classification features were extracted from the issues' contents. Afterwards, classifiers for each classification method were developed to determine whether human values are present. Finally, this study evaluated the performance of the classifiers. The following subsections describe each step of this study in more detail.

4.2.1 Preprocessing

Before extracting the features, two preprocessing activities were performed, namely content abstraction and data cleansing. This sub-step performed content abstraction to *'abstract content to their types'* [126]. Table 4.1 lists the abstracted content types and their string abstractions found in the issues. For example, the content abstraction replaced a mention in issue discussions with a **^mention^** string. This abstraction is a common step in text classification in a case where the type of the content *'is more important than the actual content'* [126]. In this case, the information on whether an issue has any mention is more important than who is mentioned in that issue.

GitHub API provides the content of an issue in both plain text and HTML formats. The preprocessing step used the HTML format for practical purposes. The data cleansing step removed punctuations, numbers, source codes, stop words, and HTML tags from the dataset. The stop words removal utilised the Natural Language Toolkit library (NLTK) [134].

TABLE 4.1: Content types found in issues and their abstractions

Content Type	**Description**	**Abstr. String**
Mention	A reference to another contributor's username [135] (e.g. @username)	^mention^
Issue number	A reference to a relevant issue number [95] (e.g. #123)	^issue^
Commit	A reference to a relevant commit [95] (e.g. a2c1423)	^commit^
Image	An image posted in the issue	^img^
URL	A link posted in the issue	^url^
Email	An email address posted in the issue	^email^

4.2.2 Feature Extraction

This study used two types of features, namely, statistical features and sentiment features. These features were extracted from the title and all posts of an issue (including the first post by the reporter). The issues were tokenised by words before the feature extraction. Those features are described in the following subsections.

4.2.2.1 Statistical Features

The statistical features used in this study were based on the occurrence of words in each classification unit (e.g. an issue or post). This study used two statistical features, namely, bag of words and TF-IDF. These two features have been used in previous studies for human values classification and their related concepts in software engineering (e.g. [60, 72, 117, 122]). These features are explained as follows:

- **Bag of Words (BoW)** is a feature where the text in a classification unit (e.g. documents or issues) is represented as terms and their number of occurrences in that classification unit [136]. Here, the BoW feature serves as the *'quantitative digest'* of that classification unit [136]. Each classification unit is represented by a vector of its terms' occurrences in this feature.

- **Term Frequency Inverse Document Frequency (TF-IDF)** is a feature that considers the importance of each term based on the frequency of the classification unit (e.g. documents or issues) [136]. In this feature, rare terms that are present in fewer classification units are considered more important. For each term t in a unit of classification d, the TF-IDF is calculated as follows [136]:

  $$\text{TF-IDF}_{t,d} = tf_{t,d} \times \log \frac{N}{df_t},$$

  where:

  - $tf_{t,d}$ is the term frequency of a term t in a classification unit d,
  - N is the number of classification units in a dataset, and
  - df_t is the number of classification units in the dataset that contains the term t.

  This feature represents each classification unit as a vector of the TF-IDF values of the terms found in the whole dataset.

In this study, the BoW feature collects the occurrences of each word in the dataset. Meanwhile, to obtain the TF-IDF feature, the number of occurrences of each word in the dataset (i.e. the *term frequency* – TF) is multiplied by the inverse of the number of issues or posts where each word is present (i.e. the *inverse document frequency* – IDF).

4.2.2.2 Sentiment Features

As discussed in Section 3.4.6, the criteria to determine the presence of values (Definition 3.1) corresponds to the presence of sentiment. For example, the privacy value is considered present in an issue if in that issue, users express their appreciation towards the privacy features provided by an application. This study investigated whether the use of sentiment features influences the performance of the classification.

A sentiment feature is a feature that captures the results of the sentiment analysis of each classification unit. Sentiment analysis is commonly used to *'analyse people's opinions, sentiments, and emotions towards entities (e.g. products)'* [137]. Sentiment analysis produces a level of positive, neutral, or neutral sentiment from a text [137]. This feature represents each classification unit with the sentiment score of its text contents.

To determine the sentiment score of each classification unit, this study utilised SentiStrength [138] for the following reasons: first, SentiStrength supports sentiment analysis in informal text communication [138]. Second, the SentiStrength tool[2] provides two sentiment strengths, namely, positive and negative sentiment, in accordance with the appreciation and dismay criteria from the empirical study (see Definition 3.1). Recent studies have used SentiStrength to analyse sentiments in the software engineering domain with reasonable results (e.g. [139]). Furthermore, using software engineering-specific sentiment analysis tools, such as SentiStrengh-SE [140], *'may lead to contradictory results if different levels of unit of analysis are considered'* if they are used *'off-the-shelf'* [141]. To achieve better results, Novielli et al. recommended retraining the sentiment analysis tool for each case [141]. Therefore, for practical reasons, this study decided to use the regular SentiStrength as a preliminary. In SentiStrength, negative sentiment is scaled from -1 (not negative) to -5 (extremely negative). Meanwhile, positive sentiment is scaled from 1 (not positive) to 5 (extremely positive). In this study, the sentiment scores calculated from the dataset were normalised to their absolute values (e.g. 1 to 5).

4.2.3 Classifier Learning

This study formulated the detection of human values as a binary classification problem to identify whether there are any human values present in the issues. The study started

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

with binary classification for the presence of any values rather than for specific values because the dataset contains a small number of cases for each value (see Figure 3.7). Furthermore, the dataset is quite imbalanced, with the number of issues wherein values were identified being only one-third of the total issues in the dataset (see Figure 3.6). This study considered these dataset characteristics in the evaluation experiments.

The experiments evaluated four well-known supervised learning methods, namely, support vector machines, random forest, multi-layer perceptron, and logistic regression. This study used these methods because of an earlier study of identifying human values in text documents that reported that a deep learning approach has lower performance in smaller datasets and *'achieve[s] good results in data-rich settings'* [63]. All of these methods have been used in prior studies to classify the content of GitHub repositories [131, 142–148]. The experiments used the implementation of those methods in the *scikit-learn* library [149]. These classification methods are described as follows:

- **Support vector machines (SVM)** is a supervised learning technique applicable to classification and regression problems [150]. The SVM method aims to separate the training set using hyperplanes that maximise the distance to the nearest cleanly split examples [150]. For classification, the SVM maps the input vectors of the new data in the N-dimensional space and decides on which side of the trained hyperplanes the new data should be located [117]. This method allows for specifying a kernel function as the decision function that calculates the similarity between the clusters of the training set and the new data [150]. This study experimented with several kernel functions (see Section 4.2.4.4) for detecting human values in issues.
- **Random forest (RF)** is an ensemble classifier that uses multiple decision trees to vote for the most popular class for the new data [151]. This method randomly generates vectors from the training set and builds decision trees from those vectors. New data are classified by running the data through the decision trees. The final class for the new data is the class that receives the majority of votes by the trees. This study experimented with different numbers of decision trees and a few measurement functions that determined the tree split (see Section 4.2.4.4).
- **Multi-layer perceptron (MLP)** is a neural network-based classifier [152] that *'made up of layers of nodes with connections between layers'* [144]. A node or perceptron is a linear classifier that uses an activation function that determines the class of an input if the output of the function satisfies certain thresholds [144].

In the training process, a loss score is calculated by comparing the predicted and the actual output. The next iteration of the training attempts to reduce this loss by updating the weight throughout the networks [144]. This study experimented with various activation and solver functions for weight optimisation (see Section 4.2.4.4).

- **Logistic regression (LR)** is a binary classification method that uses the logistic function [153] to estimate the probability of a class based on *'a linear combination of the input features'* [143]. First, the input features are formulated into a logistic model. Then, this classification method estimates the weight parameters for the features using optimisation algorithms. This study experimented with several optimisation algorithms to determine the one with the best performance (see Section 4.2.4.4).

4.2.4 Evaluation

To evaluate the performance of the classifier, this study utilised 10-fold cross-validation. This technique is commonly used to evaluate classification methods, including those in software engineering research (e.g. [124, 127]). In the 10-fold cross-validation, the dataset was split into 10 equal-sized parts. Then, a classifier is trained using nine parts and evaluated using the remaining one. The training and evaluation process is repeated 10 times such that each part was evaluated once. For the final score, the average and standard deviations of the performance of each fold were calculated.

This study used precision, recall, F1 score, and Matthew's correlation coefficient (MCC) for the performance measures. These performance measures are described as follows:

- **Precision** is defined as the fraction of correctly classified instances for a specific label (t_p) to the total of classified instances for the same label ($t_p + f_p$) [117, 136]:

$$P = \frac{t_p}{t_p + f_p}$$

  A higher level of precision means the classifier produces a higher number of correctly classified instances (true positives) and fewer incorrectly classified instances (false positives) for a particular label.

- **Recall** is defined as the fraction of correctly classified instances for a specific label (t_p) to the total number of classified instances for that label (t_p+f_n) [117, 136]:

$$R = \frac{t_p}{t_p + f_n}$$

A higher level of recall means the classifier includes a higher number of correctly classified instances from the total number of classified instances for a particular label.

- **F1 score** or *balanced F measure* is a measure that represents a harmonic mean of precision and recall [117, 136]. This measure provides a single measure that balances the weight of the precision and recall:

$$F_1 = \frac{2PR}{P + R}$$

A higher F1 score means better balanced performance between precision and recall.

- **Matthew's Correlation Coefficient (MCC)** is a classification measure that is similar to Pearson correlation [154] in statistics. This measure uses true positives (the number of correctly classified instances for a label), true negatives (the number of correctly classified instances for the opposite label), false positives (number of incorrectly classified instances for a label), and false negatives (the number of incorrectly classified instances for the opposite label). This measure is formulated as follows [155]:

$$MCC = \frac{t_p \times t_n - f_p \times f_n}{\sqrt{(t_p + f_p)(t_p + f_n)(t_n + f_p)(t_n + f_n)}}$$

Unlike the other measures mentioned above that have a range between 0 and 1, an MCC score can range from -1 to +1: A +1 score represents perfect prediction, a -1 score represents perverse prediction, and a 0 MCC score represents random prediction [155]. Other scores can be interpreted as follows: [127, 156] an MCC score below 0.2 is considered as low. An MCC score of 0.2 and less than 0.4 is interpreted as fair. An MCC score of 0.4 and less than 0.6 is interpreted as moderate. An MCC score of 0.6 and less than 0.8 is considered strong. Finally, an MCC score of 0.8 and above is interpreted as very strong.

The precision, recall, and F1 score are three standard performance measures for classification problems (e.g. [117, 124, 127]). The MCC score was included because recent studies argue that it provides an unbiased measure of performance [154, 155].

The experiments considered four parameters, namely, unit of classification, resampling technique, feature set, and classification method, to address the sub-research questions. The following subsections explain those parameters in detail.

4.2.4.1 Unit of Classification

The unit of classification represents the scope of the classification for a particular problem [157, 158]. An issue discussion starts with a post by a reporter, followed by subsequent posts from either project contributors or the issue reporter. Thus, there are two possible classification units with a reasonably sufficient amount of information to identify human values in an issue, namely, issue level and post level. The experiment considered two values for the unit of classification which are described as follows:

1. **Issue level**. This parameter value defines the use of the entire content of an issue, from the first report to the last post, as the unit of classification. The reason for considering this classification unit is because the whole issue contains all information of that particular issue. However, it is possible that including the whole issue might provide irrelevant information that makes it harder to identify values. On the other side, new issues usually still have a small number of posts. For this classification unit, the experiments used the dataset analysed in Chapter 3.
2. **Post level**. This parameter value defines the use of a post in an issue as the unit of classification. For this classification unit, the experiments used the annotation in the post level from the previous empirical study (see Section 3.2.3.2).

4.2.4.2 Resampling Technique

The dataset of human values in issue discussions is imbalanced (see Figure 3.6). An imbalanced dataset can affect the performance of a classifier towards the majority class [159]. The experiments investigated the influence of an imbalanced dataset using the following values for the dataset balance parameter:

1. **Without resampling (no sampling)**. This parameter value defines the use of the whole dataset in the experiments. The distribution of human values present in this dataset is shown in the Issue Level column of Table 4.2. This column shows that human values were discovered in one-third (367 out of 1,097 issues) of the dataset. The 1,097 issues in the dataset comprised 5,615 posts. The post level unit of classification included the issue title because human values could also be identified in the title. Thus, the post level classification unit's total data comprised 6,712 records, including posts and issue titles. The Post Level column in Table 4.2 shows the distribution of human values present in the post level. This column shows that the distribution is substantially imbalanced (628 records where human values were present compared to 6,084 records where values were not present).

2. **Oversampling**. Oversampling is a technique used to balance a dataset by generating new samples for under-represented classes [160]. In this case, an oversampling technique generated new samples for the records that had human values present. For oversampling, this experiment uses SMOTE (synthetic minority oversampling technique) [161], which works by generating samples based on *'the features of the k-nearest neighbours of instances of the minority cases'* [162]. It has been used in many classification experiments in software engineering research (e.g. [143, 162, 163]). The implementation of SMOTE in the *imbalanced-learn* library [164] was used for this purpose.

3. **Undersampling**. Undersampling is a technique used to balance the dataset by selecting a subset of a class with a majority number of samples [160]. In this case, the undersampling technique was applied to samples that had no human values present. Undersampling has also been used in recent studies in software engineering as an alternative way to handle imbalanced datasets (e.g. [165, 166]). To this end, here, this technique selected a random subset of samples using the RandomUnderSampler implemented in the *imbalanced-learn* library [164].

TABLE 4.2: The distribution of values present in the dataset

Values Presence	**Issue Level**	**Post Level**
Yes	367	6,084
None	730	628
Total	1,097	6,712

4.2.4.3 Feature Set

As discussed in Section 4.2.2, the study used two types of features from the dataset. The experiments specified four parameter values for the feature sets. The first two feature sets used only the statistical features, namely BoW and TF-IDF. The other two feature sets involved both statistical and sentiment features. This combination resulted in **BoW+Sentiment** and **TF-IDF+Sentiment** feature sets for the experiments.

4.2.4.4 Classification Method

To detect the presence of values in issues, the study experimented with SVM, RF, MLP, and LR. To obtain the best parameter for each classifier method (*hyper-parameter tuning*), the grid search process was used on a limited set of values for the methods' parameters. This approach has been used in previous work (e.g. [142, 143, 148]) for classification experiments. The arguments and their values used in the experiments are shown in Table 4.3. The experiments then compared the performance of each classifier with its best hyper-parameter results.

TABLE 4.3: Arguments for the classification methods

Method	**Arguments**	**Description**	**Values**
SVM	kernel	Kernel function for the SVM algorithm	(polynomial, rbf, sigmoid)
RF	max_depth	The maximum depth of the tree	(4, 5, 6, 8, 100)
	criterion	The function to measure the quality of the decision split	(gini, entropy)
	n_estimators	The number of trees in the random forest	(10, 100, 1000)
MLP	activation	Activation function for the hidden layer	(identity, logistic, tanh, relu)
	solver	The solver function for the weight optimisation	(lbfgs, sgd, adam)
LR	solver	The algorithm to use in the optimisation problem	(newton-cg, lbfgs, liblinear, sag, saga)

4.3 Results

This section describes the results of the classification experiments using the parameters described in Section 4.2.4. These experiments involved four parameters, namely, unit of classification, resampling technique, feature set, and classification method. Table 4.4 shows the list of experiments that included all evaluation parameters. The inclusion of two parameters, feature set and imbalanced handling, on the left side of the table resulted in 12 classification experiments. Adding two other parameters, unit of classification (i.e. issue level and post level) and classification method (i.e. SVM, RF, MLP, and LR), resulted in a total of 96 experiments. The experiments used the F1 score, which provides *'the balance between precision and recall'* [143] as the primary metrics to determine the performance of the classifier. The remaining metrics were used to provide different perspectives on the classification results. This approach is considered common in classification studies, including those in software engineering (e.g. [126, 131, 143]). The complete results for the experiments are provided in Appendix A. This section presents the results in charts to assist in comparing the results. This section discusses the experiment results from each parameter's perspective to answer the sub-research questions.

TABLE 4.4: Experiments using all the evaluation parameters

No.	Feature Set		Resampling	...	Class. Unit	Methods
1	BoW		No Sampling	...	Issue level	SVM
2	TF-IDF		No Sampling	...	Issue level	RF
3	BoW		Oversampling	...	Issue level	MLP
4	TF-IDF		Oversampling	...	Issue level	LR
5	BoW		Undersampling	...	Post level	SVM
6	TF-IDF		Undersampling	...	Post level	RF
7	BoW	+ Sentiment	No Sampling	...	Post level	MLP
8	TF-IDF	+ Sentiment	No Sampling	...	Post level	LR
9	BoW	+ Sentiment	Oversampling	...	...	...
10	TF-IDF	+ Sentiment	Oversampling	...	...	...
11	BoW	+ Sentiment	Undersampling	...	...	...
12	TF-IDF	+ Sentiment	Undersampling	...	...	...

4.3.1 Unit of Classification (RQ2.1)

To determine how the unit of classification affects the performance of the detection of human values, the best F1 performers from each classification method on the issue level dataset and the post level dataset were compared. Figure 4.2 shows the top F1 score results from each classification method applied in the whole dataset without any resampling techniques. The figure shows that the classification on the issue level dataset performed better than that on the post level dataset. The much smaller number of cases where human values are present in the post level dataset (see Table 4.2) was suspected to contribute to the lower F1 scores. An alternative explanation is that the larger number of text contents in the issue level dataset provided more apparent evidence to detect the presence of values.

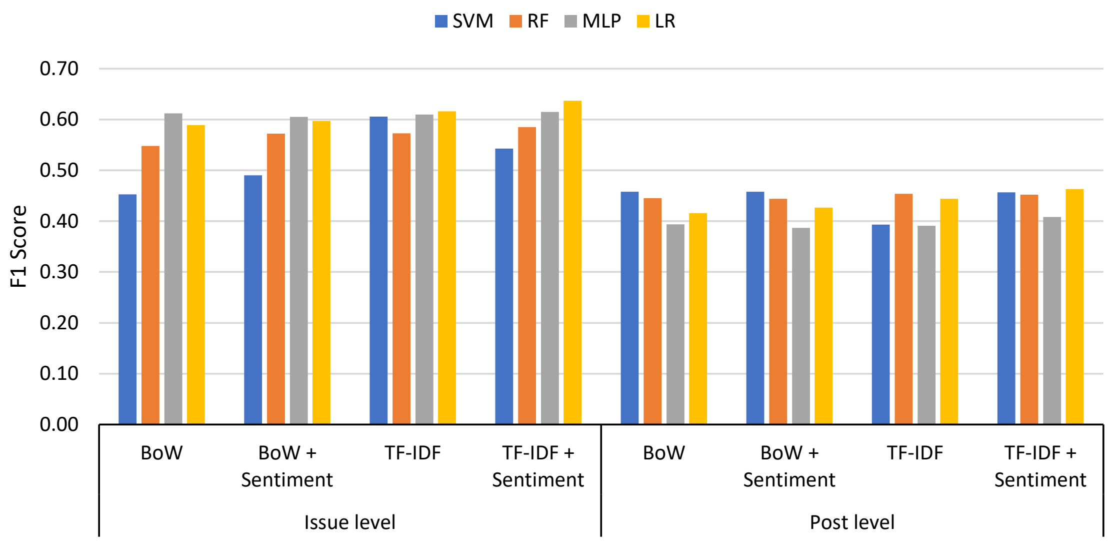


FIGURE 4.2: The top F1 score of each method for the imbalanced dataset comparing issue level with post level

The results of the oversampled and undersampled approaches in Figure 4.3 and Figure 4.4 also demonstrate that the issue level dataset outperformed the post level dataset. These approaches had substantially lower performance than using the original dataset. Section 4.3.2 describes these results from the use of resampling techniques perspective.

The results showed that the classification using the issue level dataset performed better than that using the post level dataset among all classification methods and feature sets.

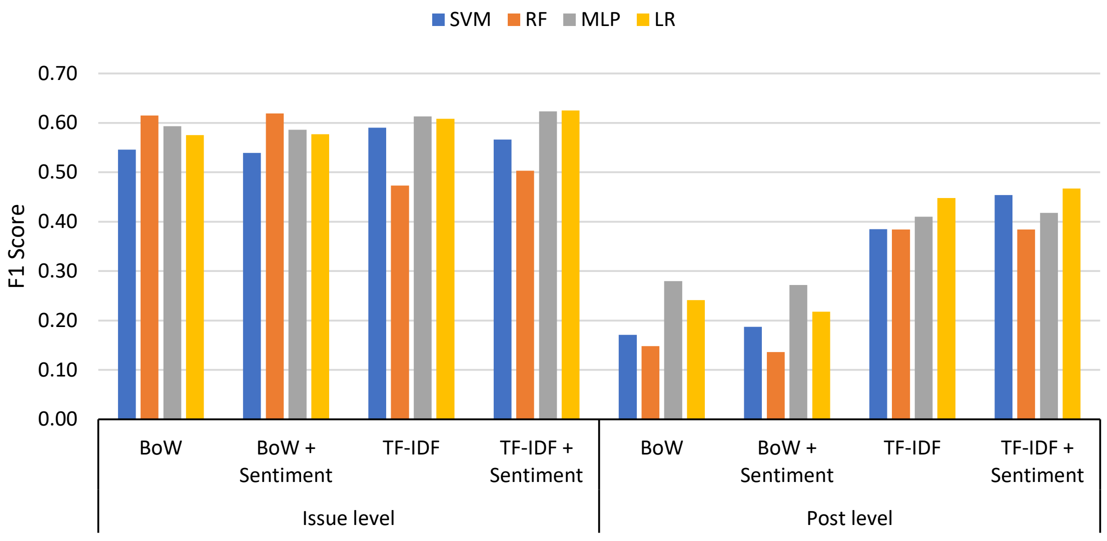


FIGURE 4.3: The top F1 score of each method for the oversampled dataset comparing issue level with post level

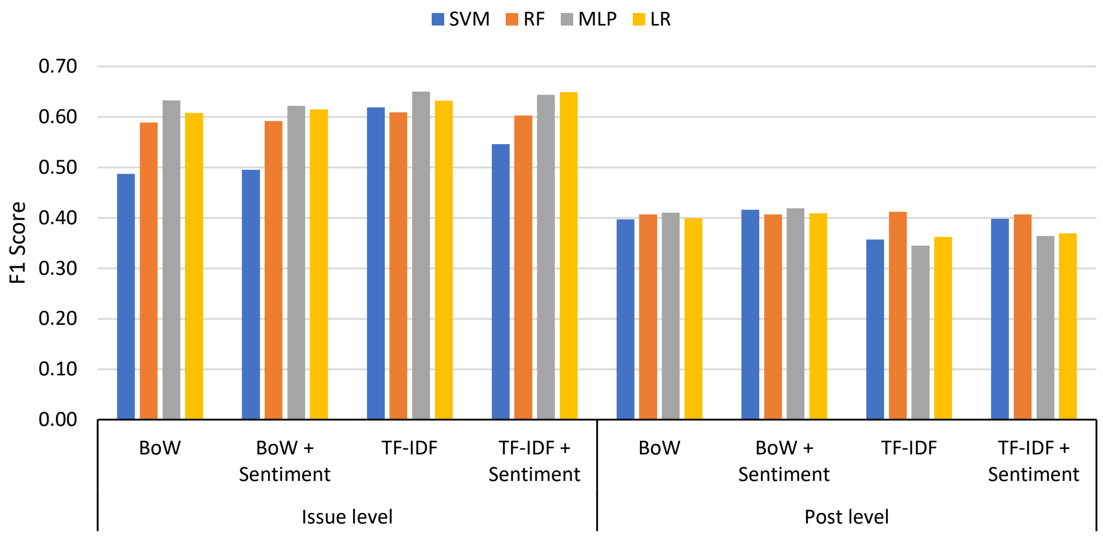


FIGURE 4.4: The top F1 score of each method for the undersampled dataset comparing issue level with post level

4.3.2 Resampling Technique (RQ2.2)

To address the effects of resampling techniques on the human values detection performance, the best classifier performances with and without resampling techniques were compared. For this parameter, the evaluation focused on comparing the top F1 scores for the issue level dataset because this classification unit performed better than the post level dataset (see Section 4.3.1). Figure 4.5 shows these scores for the SVM and RF classification methods. The oversampling and undersampling approaches generally resulted in higher top F1 performances except for the TF-IDF (with and without sentiment) feature sets for the SVM method. The oversampling approach mostly performed better among the feature sets for the SVM method. The difference in performance could reach 0.09 points (using the BoW feature).

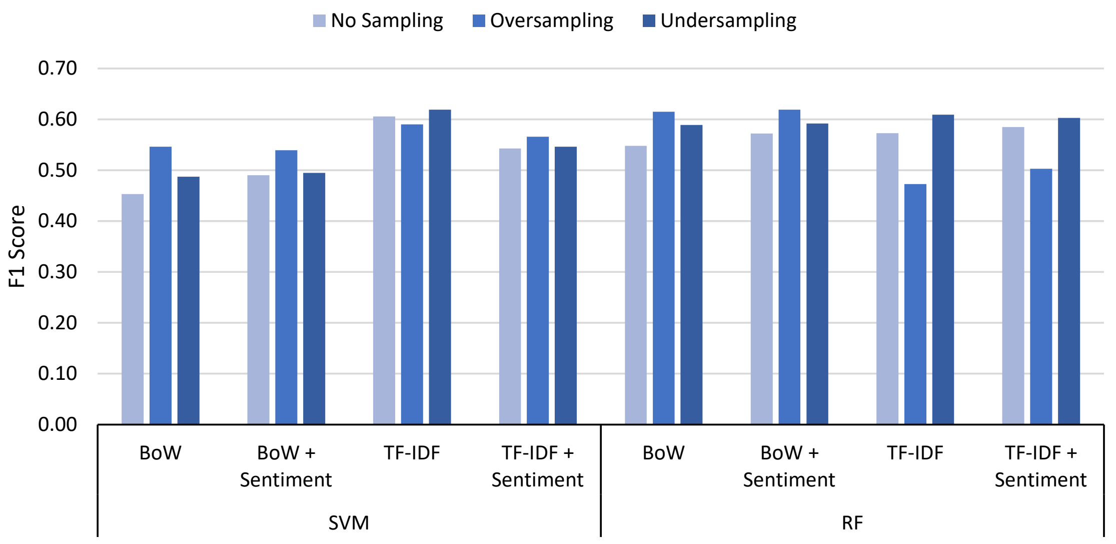


FIGURE 4.5: The top F1 score of the SVM and RF methods for the issue level dataset comparing the use of oversampling and undersampling techniques

For the RF method, Figure 4.5 shows that using oversampling and undersampling methods showed an increase of the top F1 scores for the BoW (with and without sentiment) feature set. In this feature set, classification using oversampling methods provided the best performance for the RF method. However, this is not the case with the RF method using TF-IDF feature sets. In TF-IDF (with and without sentiment), the oversampling method delivered worse performance than that without any sampling approach. The use of the undersampling approach for the RF method in the TF-IDF feature set offered the best performers of the F1 score. Similar to the SVM method case, the RF method's difference in performance could achieve 0.14 points (using the TF-IDF feature set).

The best F1 scores for the MLP and LR are presented in Figure 4.6. In the MLP method, the best F1 performances were slightly worse for the BoW features but better for the TF-IDF features than the original dataset. However, applying an undersampling technique in the MLP method provided the best performances of the F1 score. Nevertheless, the performance differences among those best F1 performances were less than 0.05 points.

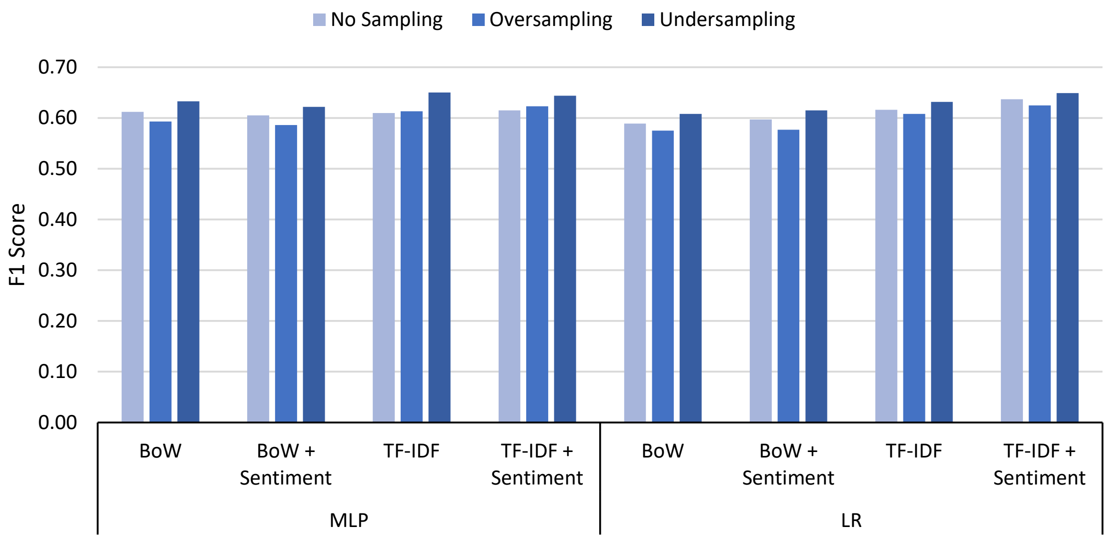


FIGURE 4.6: The top F1 score of the MLP and LR methods for the issue level dataset comparing the use of oversampling and undersampling techniques

For the LR method, the right side of Figure 4.6 shows that the best performances that used oversampling techniques were worse than the performances using the original dataset. However, the undersampling technique offered the best performances for the LR method for all feature sets. Similar to the results of the MLP method, the differences in the performances were also less than 0.05 points.

The effects of using oversampling and undersampling techniques were varied for each classification method and feature set. For the SVM and RF, the differences in performance were less substantial than those for the MLP and LR methods.

4.3.3 Sentiment Features in the Feature Set (RQ2.3)

To investigate whether the sentiment features influenced the classifiers, the evaluation compared the performance of the classifiers using the regular BoW and TF-IDF features against the same features with the addition of sentiment features. Figure 4.7 shows the perspective of feature sets for the SVM and RF methods. The figure shows that the feature set that offered the best performance for the SVM method was TF-IDF without using the sentiment feature. The addition of the sentiment feature provided better performances for the BoW feature but not for the TF-IDF feature. For the RF method, adding sentiment features generally led to better performance on both the BoW and

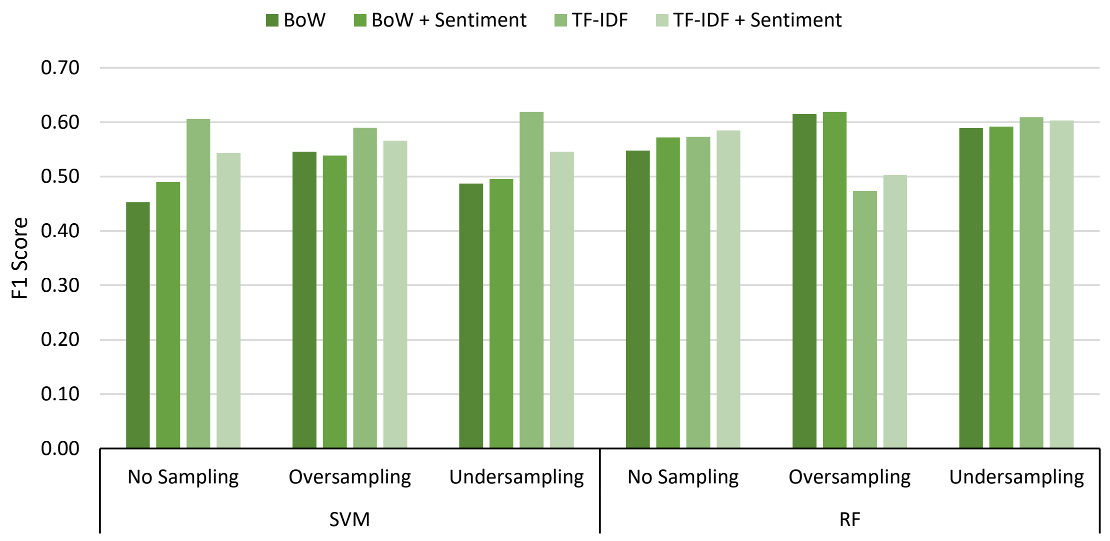


FIGURE 4.7: The top F1 score of the SVM and RF methods for the issue level dataset comparing different feature sets

the TF-IDF feature sets. The results also showed that the TF-IDF feature performed better than the BoW except when using oversampling techniques in the RF method. Different feature sets in these two methods could result in 0.15-points performance differences (found in the SVM without any resampling techniques and the RF with oversampling technique).

Figure 4.8 shows the results of the MLP and LR methods from the feature set perspective. For the MLP method, the TF-IDF feature performed better than the BoW feature. The addition of the sentiment feature led to worse performance in the BoW feature

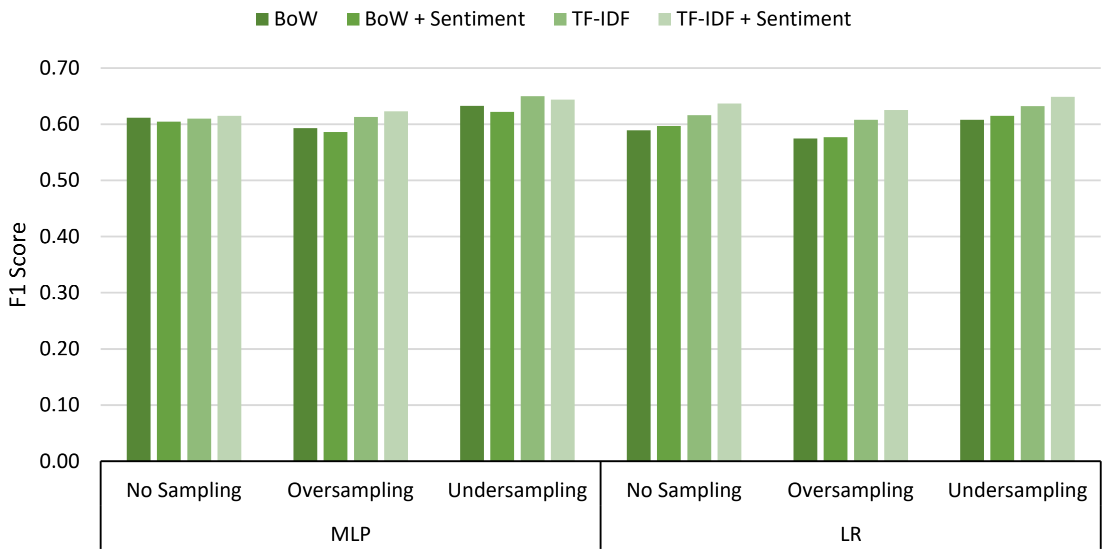


FIGURE 4.8: The top F1 score of the MLP and LR methods for the issue level dataset comparing different feature sets

but better performance in the TF-IDF feature. However, the performance differences were less than 0.05 points for this method.

For the LR method, Figure 4.6 shows that the TF-IDF feature provided better performance than the BoW feature. The sentiment feature also performed better than the feature set without sentiment in all cases of the resampling technique. Similar to the MLP method, the performance differences were less than 0.6 points.

To understand these results, the strength of the sentiment for each case of values presence in the issue dataset was investigated. Figure 4.9 shows this distribution of sentiment strength. The vertical axis denotes the negative sentiment strength (-1 to -5), while the horizontal axis represents the positive sentiment strength (1 to 5). The figure shows the number of issues wherein *'no value'* cases (cases where values were not present) were more prevalent on the upper left area of the chart. In the lower right of the figure, where the sentiments were stronger, the number of cases wherein values were present were much higher.

Figure 4.9 could mean two things. First, the cases where values were present might have had stronger sentiment than the cases where values were not present. However, because there were also many cases where values were present when the sentiment strength were low (upper left side of the figure), the classifiers could not utilise the sentiment features to detect the human values presence. Second, the figure demonstrates that human values are different from sentiments. Human values can be present anywhere, even if the sentiment strength is low.

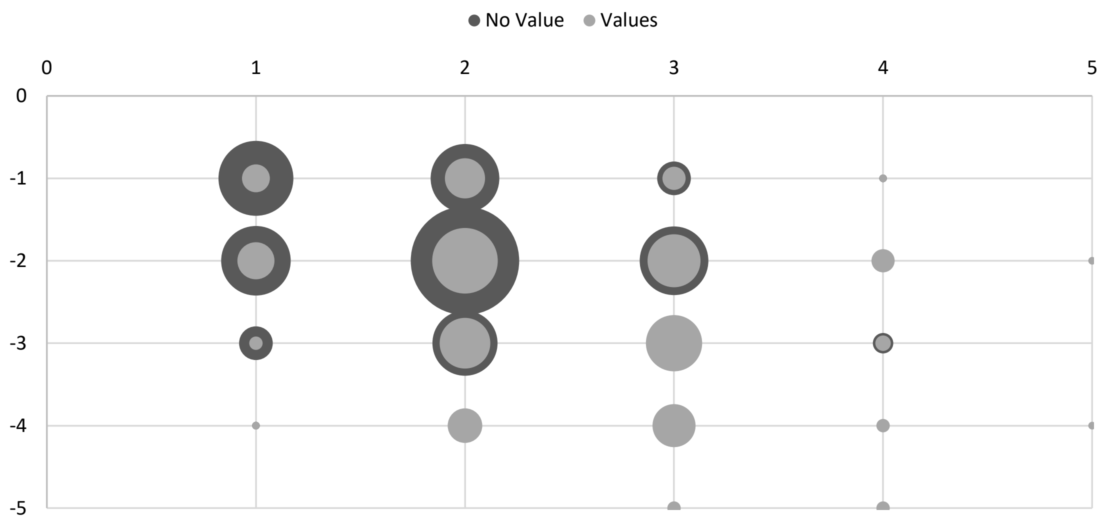


FIGURE 4.9: The distribution of sentiment strength in issues

In the majority of the cases, the TF-IDF feature performed better than the BoW feature. There are also varying effects from the use of sentiment features on the performance of the classifier methods. Similar to the resampling technique perspective, the performance differences in the SVM and RF methods are more substantial than those in the MLP and LR methods.

4.3.4 Classification Methods (RQ2.4)

To determine which classification method performed the best, the best performers in each classifier method were selected and compared. Here, the evaluation focused on the issue level dataset because of its better performance than the post level dataset (see Section 4.3.1). Table 4.5 shows the best F1 performers of each classification method on the issue level dataset. The best performance of the SVM method was demonstrated using the undersampling technique and the TF-IDF feature with kernel parameter radial basis function. The BoW with sentiment features and the oversampling technique performed best for the RF method. The hyper-parameter setup for this performer used the entropy information gain and 1,000 decision trees for the RF method. The best F1 performers for these two methods had the same F1 score (0.619). However, the RF method offered better precision but slightly lower recall than the SVM method.

TABLE 4.5: The best F1 performance for each classification method

Method	**Imb. Handling**	**Feature**	**Precision**	**Recall**	**F1**	**MCC**
SVM	Undersampling	TF-IDF	0.575±0.051	0.676±0.059	0.619±0.038	0.407±0.064
RF	Oversampling	BoW+Sentiment	0.637±0.063	0.610±0.085	0.619±0.058	0.438±0.078
MLP	Undersampling	TF-IDF	0.582±0.043	0.741±0.062	0.650±0.032	0.451±0.053
LR	Undersampling	TF-IDF+Sentiment	0.570±0.045	0.757±0.047	0.649±0.035	0.445±0.059

For the MLP method, Table 4.5 shows that the best performance used the undersampling technique and TF-IDF feature. The best hyper-parameter setup for this method was using the hyperbolic tan function (*tanh*) for the activation function and the L-BFGS (Limited-memory Broyden–Fletcher–Goldfarb-Shanno) solver. Similarly, for the LR method, the best performer was using the undersampling method and the TF-IDF but with sentiment feature with the stochastic average gradient (SAG) solver. The F1 scores for the MLP and LR were very close (0.001 difference). However, the precision of the MLP was slightly better than that of the LR method. Conversely, the recall of the MLP was slightly lower than that of the LR method.

Comparing the precision for all methods, Table 4.5 shows that the RF method was the best. As for the recall, the LR method had the best score. The MCC score was aligned with the performance rate of the F1 scores. The MCC score for the RF method was higher than that of the SVM method, although the F1 scores were the same. This condition means the RF method offered a better overall prediction than the SVM method. Nevertheless, the MLP method was still the best performer among all these methods, with the highest F1 and MCC scores.

TABLE 4.6: The average values of confusion matrix of all folds in the MLP (Precision=0.58, Recall=0.74, F1=0.65, MCC=0.53)

		Predicted		**Total Actual**
		Values found	**Values not found**	
Actual	**Values found**	**27**	**10**	37
	Values not found	**20**	**53**	73
Total Predicted		47	63	110

Table 4.6 shows the average values for confusion matrix of all 10 folds in the top performer of the MLP classifier mentioned in Table 4.5. The Total Actual column of this table shows the imbalanced nature of the testing set (i.e. 36 values issues : 73 no values issues). This means the undersampling was only applied in the training set. The confusion matrix shows that the MLP classifier correctly identified the majority of the issues where values were found (i.e. 27 out of 37 issues). However, this classifier had lower performance in detecting issues where values were not found (i.e. 53 out of 73 issues). The classifier incorrectly identified 20 issues to have values (i.e. false positives). Meanwhile, only 10 issues were incorrectly identified to have no values found (i.e. false negatives). This results in a higher recall (0.74) and a lower precision (0.58).

The multi-layer perceptron method performs the best compared to the random forest, support vector machine, and logistic regression methods in this study. However, the difference observed was not large across the classification methods.

4.4 Discussion

This section discusses the results of the evaluation experiments of human values detection in issue discussions. This section also discusses the challenges in automating human values detection for issue discussions.

4.4.1 Factors that Influence Human Values Detection

The results demonstrated that the issue level dataset as the classification unit generally performed better than the post level dataset. These results might have been influenced by the substantially imbalanced number of cases in the dataset of the post-level classification unit (see Table 4.2). The use of resampling techniques influenced the performance of the classifiers on the imbalanced dataset. Although the outcomes varied, the best performers in each classification method used one of the resampling techniques (see Table 4.5). The performance of the classifiers while using sentiment as an additional feature varied. However, this feature offered the best performance in the RF and LR methods. The use of SentiStrength may (partially) have contributed to these variations. The sentiment strength distribution shows that the cases wherein human values were present in the dataset had stronger sentiments but did not influence the classification much. Further studies could provide a higher weighting scheme to the stronger sentiment. Although SentiStrength can provide positive and negative sentiment, it is not specifically designed for software engineering discussions. Future studies could experiment with sentiment analysis tools developed for software engineering or consider other additional features.

The imbalanced nature of datasets was also found in earlier studies that investigated human values in text documents. These studies (e.g. [65, 66, 167]) also reported low performance for specific values with a low number of occurrences in the datasets. Nevertheless, these findings showed that the detection of human values in issue discussions could be automated to some extent. To improve detection performance, some approaches could be used, such as experimenting with various classification techniques or increasing the size of the dataset using cost-effective annotation schemes [63, 167]. Using emerging approaches, such as word embedding or deep learning, could be a potential direction for future research. However, the datasets may need to be expanded to cater to such approaches. Alternatively, other methods that do not require large datasets, such as keyword-based or rule-based approaches could also be utilised. For instance, keywords could be collected from literatures [25, 56], both from social sciences and software engineering, related to each specific value.

4.4.2 Limitations of Automated Human Values Detection

The experiments indicated that the best F1 performance was 0.65. The best performance of each classification method, in general, provided higher recall scores than precision scores (0.61 to 0.75). This means that the automated approaches managed to find 60 – 75% of the issues where values are found. The results also show a higher number of false positives than false negatives. The MCC scores ranged from 0.407 to 0.451 for the best performance of each classification method. These MCC scores could be interpreted as moderate scores [127, 156]. Recent studies on the detection of human values in text documents initially reported low performance (F1 score of 0.45 [60]), but this was improved by a series of studies in the following years [61, 64, 66] resulting in better performance (F1 score of 0.74 [67]). These studies demonstrated that developing automated detection of human values is not a trivial task. Abstract concepts of human values may contribute to this challenge. Future studies are needed to improve the performance of the classifiers (see Section 4.4.1). Meanwhile, the understanding of how usable these performance results are in practise is still limited. Software practitioners may also have preferences for some metrics in their real-world settings. To obtain this understanding, another study is essential that involves software practitioners.

This study is also limited in determining whether human values are present in issue discussions. Current classifiers are not designed to detect specific human values (e.g. conformity or face). This limitation came from the limited number of cases wherein specific values were discovered (see Figure 3.7). The imbalanced nature of datasets was also found in previous studies of the analysis of human values in text documents (e.g. [60, 61, 64]). Future work could expand the datasets for specific values by targeting specific types of applications. For example, the hedonism value could potentially be discovered in issue discussions of computer games. Nevertheless, at this stage, there is still limited knowledge on how human values detection results could be used to support software development. Therefore, a study with software practitioners is required to investigate how the detection of human values could be helpful in practice.

4.4.3 Potential Uses of the Automated Human Values Detection

Automated human values detection indicates the presence of human values in issue discussions. Displaying the detection results with the corresponding artefacts has been one common use of automated classification in previous software repository studies (e.g. [126, 168]). In this way, the detection results help software practitioners decide what they should do with these values-related issues. However, software practitioners may not be familiar with using an automated detection tool. To this end, to be practical and helpful, the automated detection tool needs to be integrated into the software development tool chain such that software practitioners can use it in their development activities. To this end, further research involving practitioners is needed to understand their perceptions of human values and how a values detection tool could help them during software development.

The experiments showed that the detection of values works best at the issue level. This result provides a higher level of indicator (i.e. issue level instead of post level) where values are detected in a values detection tool for software development (e.g. a dashboard). This means that software practitioners are required to read through the whole issue. Alternatively, a classifier trained using the issue level dataset could also be used to detect the presence of values in the post level. However, more studies are required to investigate the impacts of this finding in the industry settings.

4.5 Threats to Validity

This section discusses possible threats to the validity of this study in automating the detection of human values in issue discussions. A possible threat to the **construct validity** came from the use of performance measures in the evaluation experiments. This study used several evaluation metrics to investigate the performance of the parameters in the experiment from several different angles. These metrics have been used in many text classification studies in software engineering. Another potential threat may come from the use of non-software engineering-specific sentiment analysis tool in the study. The use of such a specific tool may result in better classifier performance.

A threat to the **internal validity** came from the analysis process when developing the datasets. This threat was mitigated by conducting a pilot study and involving three validators in the main study, as described in Section 3.5.

Regarding threats to the **external validity**, the results might have been different if this study had included all issues when developing the dataset. However, 1,097 issues represents a statistically significant sample size (see Section 3.5). These experiments also cannot be generalised to other software artefacts in the GitHub repository. Each artefact may have different characteristics in issue discussions. Moreover, there is no guarantee that the results of the experiments can be transferred to other projects. Using a different set of projects may result in different results.

4.6 Concluding Remarks

This chapter describes the investigation to address the second research question of this research: *to what extent can the detection of human values be automated?* This study consists of a series of experiments involving several evaluation parameters, such as classification unit, resampling technique, feature set, and classification method, to automate the detection of the human values. The best classifier for the detection of the human values was the multi-layer perceptron method with random undersampling and TF-IDF feature in the issue level dataset with a precision of 0.582, a recall of 0.741, an F1 of 0.650, and an MCC of 0.451. This study also revealed that the issue level dataset performs better than the post level. Furthermore, this study found that the resampling techniques and sentiment features had varying effects on the classification results. Further studies could improve the performance of the detection using sentiment analysis tools designed for software engineering and incorporating other feature sets.

To understand how automated values detection could be useful in software development practice, a study involving software practitioners is needed. Moreover, further study is required regarding the perceptions of software practitioners towards the performance of the automated detection approaches. To this end, a study involving software practitioners was conducted and is described in Chapter 5.

Chapter 5

A Human Values Dashboard for Software Development

There is a growing awareness of the importance of human values in software systems. However, there are limited tools available to support the integration of human values during software development. Most of these tools are focused on concepts related to specific human values that are well known in software engineering (e.g. privacy and security). This chapter describes a study on developing a human values dashboard to (partially) address the gap. As described in Chapter 1, the human values dashboard proposed in this research uses issue discussions as its source. The case study described in Chapter 3 showed that human values are present in issue discussions. Additionally, Chapter 4 demonstated that the detection of human values can be automated. The human values dashboard proposed in this research uses the automated detection approach as one of its components. However, to understand how such a tool could support the consideration of human values in software development, this chapter describes a multistage study conducted to develop the dashboard. First, an exploratory study was performed by interviewing 15 practitioners to investigate the possibility of using the dashboard to help address human values in software development and its potential benefits. Second, the dashboard was developed using the automated approach discussed in Chapter 4 as one of its components. Finally, a study involving 10 other practitioners was performed to investigate the usefulness of the dashboard and their perceptions of the automated approach utilised in the dashboard. The findings of this study highlight the potential benefits and challenges of and suggestions for improving the dashboard.

5.1 Introduction

As described in Chapter 1, several approaches have been proposed to support software practitioners in addressing human values in software. These approaches are commonly proposed in the form of frameworks, techniques, practices, and guidelines, such as value-based requirements engineering [15], value-sensitive design [13], and continual values assessment [11]. However, these approaches aim to consider human values at a specific phase of software development, such as the requirements or design phase, and satisfy a specific type of practitioner (e.g. system analyst). Furthermore, these approaches are not necessarily equipped with a tool to understand and consider human values during software development. Some tools are available for concepts similar to specific human values that are well known in software engineering, such as privacy or security. Nevertheless, general tools that cover the full range of human values are quite limited (e.g. [113]). Bridging this gap, this research proposes a human values dashboard as a tool to help software practitioners understand and address human values during software development (see Chapter 1). This human values dashboard uses issue discussions as its source for human values (see Chapter 3) and utilises the automated approach described in Chapter 4 to detect human values.

In software development, dashboards are commonly used to support decision-making [77, 78], promote awareness within a project [79, 80], and monitor development activities [83]. It is common for a dashboard in software development to use development artefacts as its source. For instance, Leite et al. developed a dashboard that used commit history to detect unusual events [83]. Several other dashboards have also been developed using artefacts from software repositories [85–88]. In terms of values in software development artefacts, previous studies have investigated a few values, such as security [20, 22, 23, 56], privacy [18, 19, 24, 57–59, 68], and energy efficiency [25, 26]. Although these works did not specifically address security, privacy, or energy efficiency as values, the works showed the possibility of discovering values in the artefacts. A dashboard is suitable for supporting the consideration of human values in software development because it allows information to be displayed visually to *'facilitate understanding'* [75]. A dashboard can help clarify the lesser-known and abstract concept of values [7, 10] to software practitioners.

This chapter describes an attempt to support human values considerations by developing a human values dashboard and investigating how it could be helpful for software development (**RQ3**). The contributions of the study described in this chapter are as follows:

1. This study captured software practitioners' perceptions on human values in software development.
2. This study identified six high-level requirements for the development of a human values dashboard.
3. This study developed a human values dashboard based on issue discussions using the automated approach described in Chapter 4.
4. This study compiled suggestions from software practitioners on the dashboard's benefits in various software development roles and phases.
5. This study gathered feedback for the improvement of the human values dashboard.

The remainder of this chapter is organised as follows: Section 5.2 discusses the sub-research questions and methodology of the study. Section 5.3 presents an exploratory study to understand how a human values dashboard could be useful for practitioners. Section 5.4 explains the development of the human values dashboard. Section 5.5 presents a study conducted to investigate participants' perceptions of the dashboard. Section 5.6 discusses the findings. Section 5.7 discusses the possible threats to the validity of the study. Finally, Section 5.8 concludes this chapter with potential future directions for the research.

5.2 Methodology

To address the third research question of this research, *how can a human values dashboard be useful in software development?*, this study was conducted in three stages, namely the exploration, dashboard development, and feedback stages. Figure 5.1 presents the methodology for this study. In the first stage, an exploratory study that included developing a dashboard prototype and interviews was performed. The results of this stage comprised a set of practitioners' perspectives on the envisioned human values dashboard. In the second stage, the results of the first stage were incorporated to develop a human values dashboard. Finally, an interview study was performed to obtain feedback from software practitioners to improve the human values dashboard. Sections 5.3 to 5.5 describe each stage in more detail.

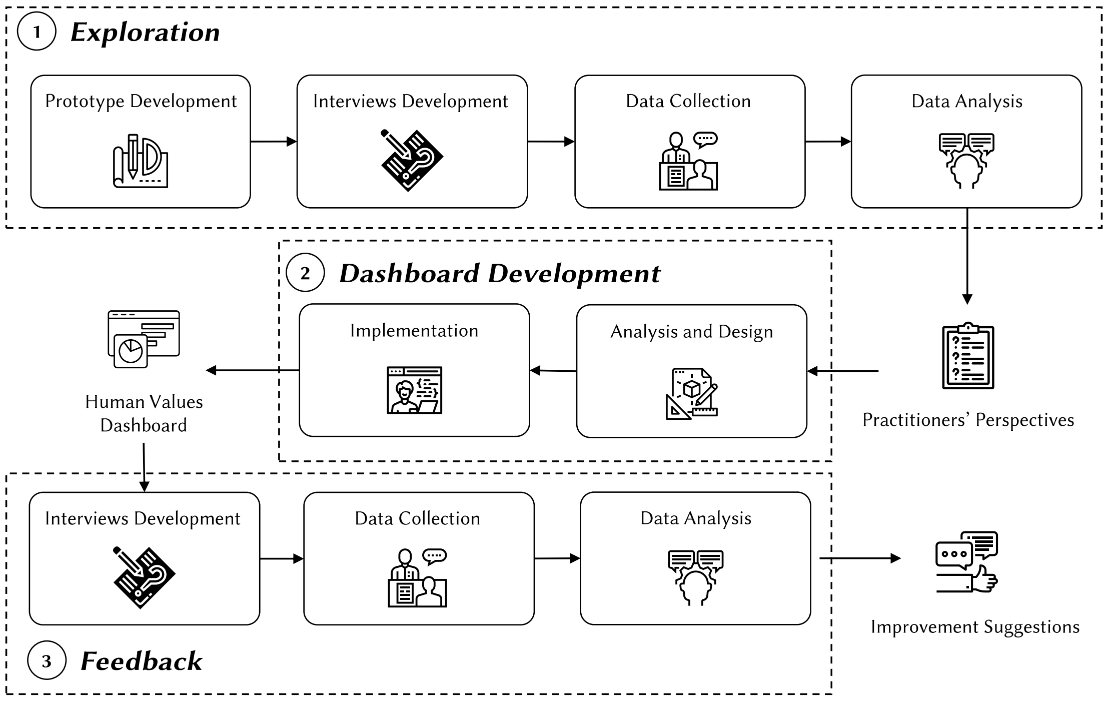


FIGURE 5.1: Methodology to answer RQ3

5.3 Exploration Stage

To make the detection of human values useful for software practitioners, this study argues that the detection results should be presented and incorporated in a tool used in software development activities. This approach has been used in previous studies to make the study results useful for practitioners [69, 126]. A human values dashboard is envisioned to be such a tool. Figure 5.2 presents a large vision of a that uses software development artefacts as its data source and displays identified human values in the artefacts. This research proposes that such a dashboard consists of a back end and a front end. The back end of the dashboard provides functionality to identify values from software development artefacts. Although the identification of values could be done manually (e.g. by the development team), it would require considerable effort by the development team (see Section 4.1). Thus, this research proposes the use of an automated approach. The back end is necessary because these artefacts naturally do not have values identified in them yet. The dashboard's front end displays values identified from various artefacts in different views (for different roles). This thesis uses the terms *human values dashboard* and *values dashboard* interchangeably.

To understand how a human values dashboard could support the consideration of human values during software development, it is first necessary to understand whether

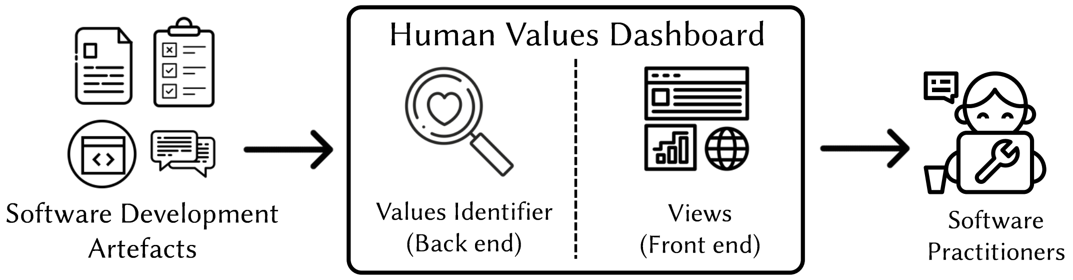


FIGURE 5.2: Human values dashboard

software practitioners consider human values important such that a tool is necessary. Then, it is necessary to explore the possible benefits of that tool for different roles in software development. As the dashboard uses software development artefacts as its source, it is also important to understand which artefacts are considered by practitioners as the most suitable. Finally, because software practitioners are the end-users of the dashboard, it is also necessary to know what they need to be implemented in the dashboard. Based on these, the following sub-research questions were developed:

RQ3.1 What are the perceptions of practitioners towards human values in software development?

RQ3.2 Who will benefit from and what are the potential benefits of a human values dashboard?

RQ3.3 Are software development artefacts suitable for identifying values for the dashboard? If so, which artefacts?

RQ3.4 What is needed for a human values dashboard to be helpful in software development?

This exploration stage used semi-structured interviews supported with prototyping to answer those sub-research questions. The interview method was chosen as the data collection method because it could provide insights and opinions about the object of interest from the participants [169]. Additionally, prototyping is commonly used for requirements elicitation to *'provide users with an idea of how a system will behave'* [170]. Developing an artefact as a prototype to accompany an interview is also commonly used in the field of information systems [171] and considered as the first iteration in design science research methodology [172–174]. Prototyping helped communicate the idea and obtain feedback from the interview participants to develop a human values dashboard here. This stage involved developing a prototype of the dashboard and interview questions. The following subsections describe these two processes in more detail.

5.3.1 Prototype Development

The prototype of the human values dashboard was developed to communicate the idea of such a dashboard. The development of the dashboard prototype started with examining the literature, especially the studies related to dashboard development (e.g. [76, 77, 79, 84, 175]). Furthermore, existing dashboards for software development [85–88] were examined to discover how they display information to support software development. These examinations revealed the following points:

(E1) Existing dashboards for software development have a common main objective to monitor the software development process, such as by determining how many activities are happening or how many remaining issues to be addressed. This functionality allows the development team to be aware of the status of the project. This finding resulted in the hypothesis that displaying values in a dashboard could be beneficial to promote awareness of values to the development team.

(E2) Software practitioners prefer to use a dashboard to support their operational tasks [77], and these tasks can be different for each role. Thus, this finding suggested that it is necessary to develop several dashboard views to cater to various roles in software development.

(E3) Artefacts from software repositories can be used as the data source for a dashboard [85, 86]. This finding suggested that the vast amount of data in repositories is suitable for a dashboard. Furthermore, practitioners are accustomed to repositories in their daily activities.

These findings were used to guide the decisions in developing the prototype of the human values dashboard. As a starting point, the issues artefact (i.e. GitHub Issues) was chosen as the source for the values identification to be displayed in the dashboard prototype (**E3**). This decision was made because issues represent tasks that need to be addressed by the development team [30, 176]. Furthermore, this is a place where discussion about values happens (Figure 3.7) as demonstrated in the previous case study (see Chapter 3). **E1** and **E2** suggested that the prototype could display issues and their identified values to support practitioners in addressing those values during development activities. Moreover, the dashboard identified and displayed which values were found and in which issues, and the number of issues wherein values were discovered in the software projects. These measures were regarded as sufficient to cater to various roles in software development (**E2**). For instance, information on which values where identified and in which issues they were identified could help developers and testers. Furthermore, the number of issues wherein values were discovered could be useful information for a project manager to ensure those issues are appropriately addressed. Therefore, it was argued that these measures could help practitioners address these values during development activities (**E1** and **E2**).

Following this, a number of issues from a random open-source project repository were sampled and labelled with values. The prototype then included three static web pages to display those issues and their values labels (Figure 5.4):

1. **Summarised values overview (OV)**. This view displays the number of issues containing each specific value (e.g. accessibility, pleasure) identified in a project. This view is commonly provided to facilitate a quick understanding of the metrics of interest [85, 86, 175]. A radar chart was used to display the number of issues where each value was identified to enable practitioners to compare values that need to be addressed in a project. Figure 5.3 shows this dashboard view. The upper part of this prototype view simulates the presence of six values (e.g. privacy, security, wealth, universalism, pleasure, and freedom) in two different projects. The values for this view were chosen randomly to illustrate the capabilities of comparison between values. This chart shows the number of issues where a specific value was present. For example, privacy was detected in 27 issues of Project A as shown in the upper left of Figure 5.3. The lower part of this prototype view illustrates a comparison of the presence of values in two different projects.

2. **Values-labelled list (LI)**. This view (Figure 5.4a) displays a list of issues with their corresponding value labels. This view is an adaptation of how GitHub displays issues but with labels or tags specific to human values. For this prototype, the values are labelled manually. The colour of each values label is used to distinguish one value from the other. Software practitioners are already familiar with this view (**F3**) because it can be found in repository platforms. Similar work has used this approach [126] to present the result of automated classification on an artefact from repositories.

3. **Values-labelled timeline (TM)**. This view (Figure 5.4b) presents the identified values chronologically based on when the issues are posted in the repository. The colours in this view represent whether an issue has been closed (green) or not (red). Timeline had been used in previous works [84, 177] to visualise software evolution.

This view was used to explore whether the emergence order of values during development could benefit software practitioners.

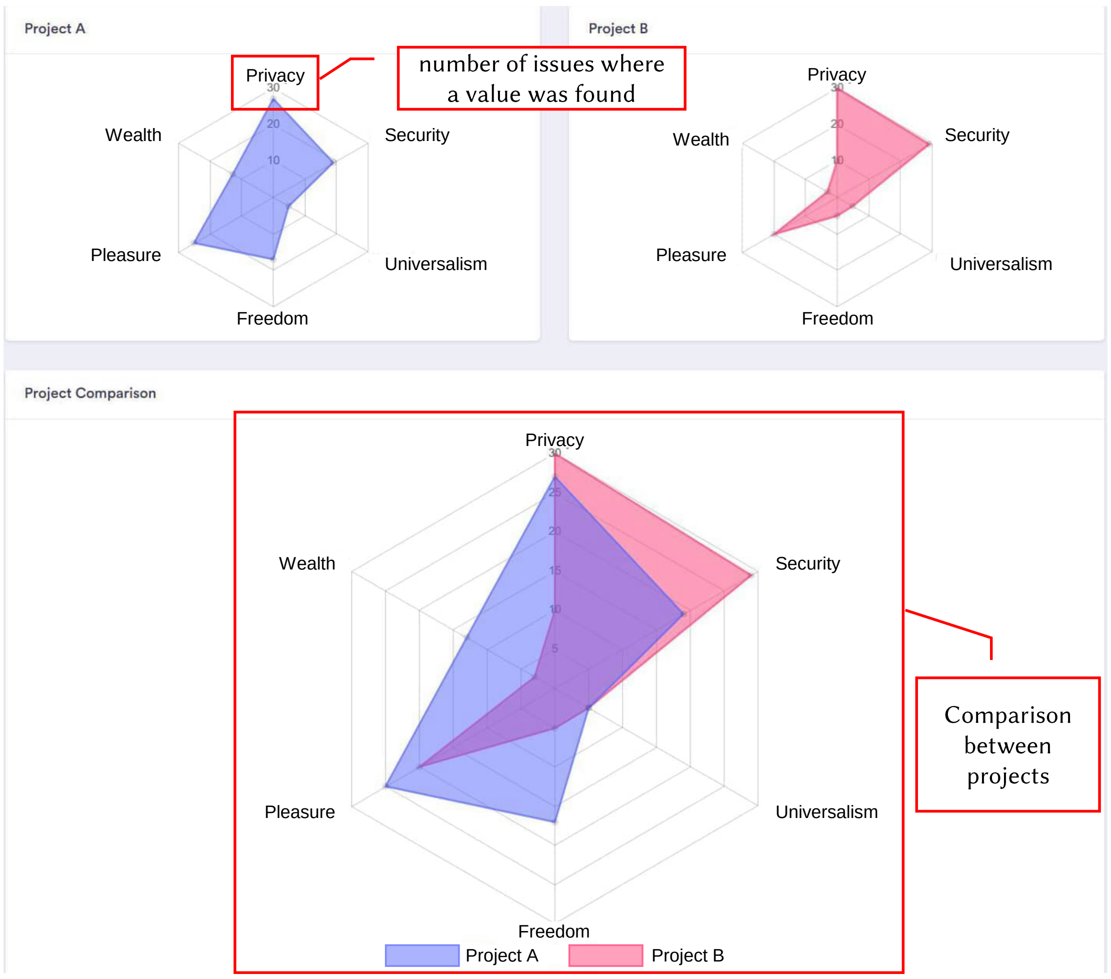
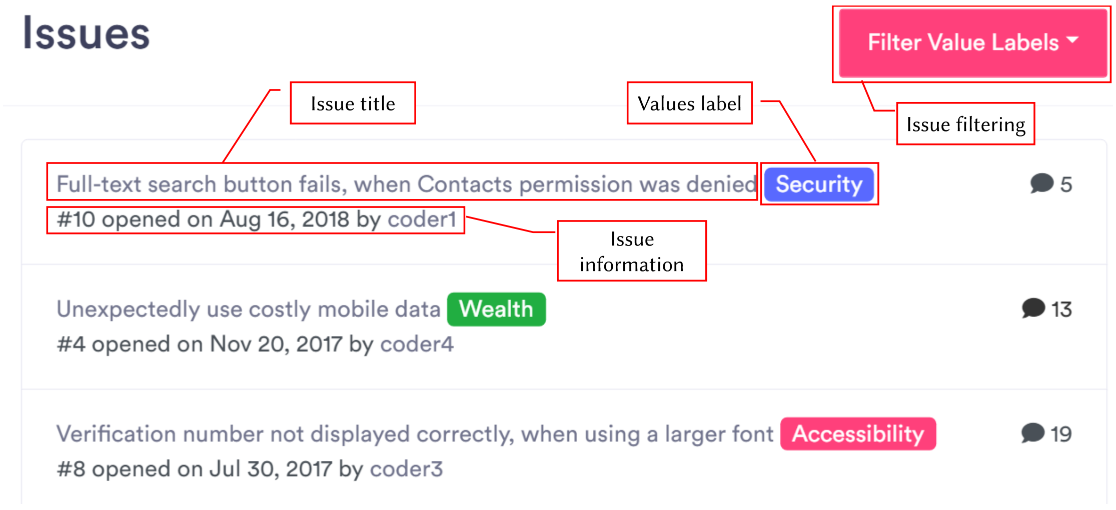


Figure 5.3: Summarised values overview (OV)
(Boxes in the red outline are not part of the dashboard)

5.3.2 Interview Guide Development

An interview guide was developed in line with the sub-research questions in Section 5.3. The semi-structured interview consisted of three parts. The first part of the interview asked for the demographic information of the participants, such as their roles and experiences. In the second part of the interview, the concepts of human values and Schwartz's model [1, 28] were explained. Then, practitioners were asked to share their opinions regarding the consideration of human values in software development. The last part of the interview presented the prototype to the participants. The


Issues

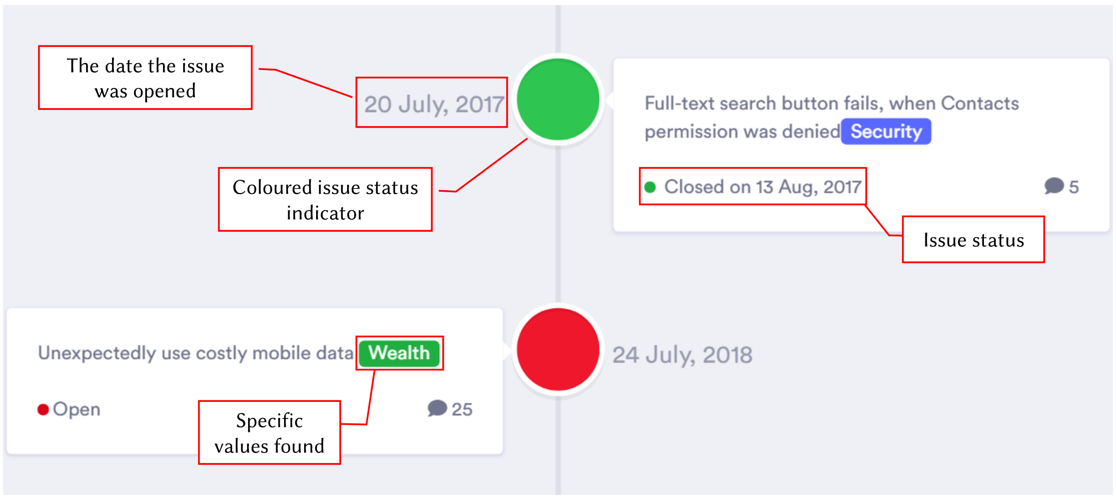
Filter Value Labels

Full-text search button fails, when Contacts permission was denied Security 5
#10 opened on Aug 16, 2018 by coder1

Unexpectedly use costly mobile data Wealth 13
#4 opened on Nov 20, 2017 by coder4

Verification number not displayed correctly, when using a larger font Accessibility 19
#8 opened on Jul 30, 2017 by coder3

(A) Values-labelled list (LI)


20 July, 2017

Full-text search button fails, when Contacts permission was denied Security
● Closed on 13 Aug, 2017 5

Unexpectedly use costly mobile data Wealth
● Open 25

24 July, 2018

The date the issue was opened
Coloured issue status indicator
Issue status
Specific values found

(B) Values-labelled timeline (TM)

FIGURE 5.4: Dashboard view prototype
(Boxes in the red outline are not part of the dashboard)

participants were asked about its usefulness and what they could suggest for the dashboard to be helpful for them. This interview guide was adjusted based on the feedback received from the supervisory team and other group members.

Before the actual interviews, a trial interview was performed with two software practitioners to simulate and refine the interview guide. Note that the trial interviews were not included in the data analysis. A note was prepared to capture ideas from the participants to adapt the questions. Some adjustments from the trial interviews were incorporated into the final interview questions. The interview questions are available in Appendix B.

5.3.3 Data Collection

Participants were recruited based on the two criteria described below. Then, an interview session was arranged for each participant. The remainder of this section describes the process in more detail.

Participant selection criteria. The criteria for recruiting the interview participants were software practitioners who (1) had been involved in a software development project and (2) were familiar with software development artefacts, including artefacts from software repositories. The rationale for the criteria came from the objective of helping practitioners integrate human values during software development with a dashboard.

Participant recruitment. The recruitment of participants involved emailing an invitation to open-source project contributors on GitHub. These email addresses were made available by the contributors themselves on their GitHub pages. Interested participants were asked to reply to the invitation email. The participation invitation was also published on our group web page and social media sites, such as LinkedIn and Twitter. Additionally, some of our colleagues were asked to spread the invitation to their networks. Interested practitioners were asked to inform us of their emails through our colleagues or an online form. The participant candidates were then contacted through email to request their consent and arrange an interview session.

Profile of the participants. Table 5.1 shows the profiles of the participants involved in this exploration stage. The participants had various roles, such as project manager, product owner, system analyst, developer, and user interface (UI) designer. Most of them had less than 10 years of experience in software development. Four had more than 15 years experience. The participants were spread across four continents, but most of them were located in Asia.

Interview protocol: Before the interview session started, the participants were requested to read the explanatory statement and fill out the interview's consent form. The interviews consisted of three parts. In the first part, the participants were asked about their professional backgrounds. The second part started with a short explanation of human values in software development. Examples of the consideration of human values in software development were also given. Then, the interview sought the participants' perceptions of human values and whether they considered any human values when developing software (e.g. *'Based on our examples, do you have any*

Table 5.1: Profile of the participants

ID	Role	Experience (years)	Location
P01	Developer	18	Europe
P02	System Analyst	21	Europe
P03	Project Manager	6	Asia
P04	System Analyst	5	Asia
P05	Developer	12	Australia
P06	Developer	3	Europe
P07	Developer	6	Asia
P08	UI Designer	21	Asia
P09	Developer	2	Asia
P10	Project Manager	14	Asia
P11	Developer	3	Asia
P12	Developer	8	Asia
P13	Developer	4	Australia
P14	Product Owner	30	North America
P15	Developer	7	Asia

similar experience when developing software?'). In the final part of the interview, the proposal of developing a human values dashboard using software development artefacts was presented. More specifically, the presentation first showed some examples of values discussions in GitHub Issues. Then, the prototype (Section 5.3.1) was presented and described to the participants. Finally, the interview probed for requirements and insights from the participants on how a values-driven dashboard should look to bring about benefits for software development (e.g. *'Is there anything you want to have on the dashboard to make it more useful for you?')*

This exploration stage recorded the interviews using a video conference system with the participants' permission. The number of interviews was not specified in advance. Instead, the recruitment and interviews continued in parallel with the analysis of the data until data saturation was reached [178, 179]. The convergence of answers and ideas became more apparent in the data analysis after 15 interviews. The mean duration time of the interviews was 47 minutes and 7 seconds. Professional transcription services transcribed all audio recordings of the interviews.

5.3.4 Data Analysis

The thematic analysis approach [180] was used to analyse the interview data. The present author conducted a large portion of the analysis. Analyses that primarily involve one analyst have been used in previous research [125, 181] by including reviews and discussions with other people. In the analysis of this stage, the supervisory team was consulted in the event of any doubts or difficulties and involved in reviewing the analysis results. Following the thematic analysis approach steps, the present author familiarised himself with the interview data by reading the transcriptions and listening to the audio recordings. Then, the present author analysed the transcriptions, generated codes, and organised them into themes. Following this, several meetings were organised with the supervisory team to review the identified codes/themes and determine their relations. Finally, the present author assigned a name and definition for each theme based on the discussions. The resulting themes were presented to the supervisory team and the other group members to obtain feedback. This feedback was then incorporated to create the final themes.

5.3.5 Results

This section presents the results of the exploration stage. The first subsection describes the practitioners' perceptions of human values. The second subsection presents the benefits of a human values dashboard, and the following subsection lists software development artefacts suitable for the dashboard. Finally, the practitioners' needs in a human values dashboard are presented.

5.3.5.1 Practitioners' Perceptions of Human Values (RQ3.1)

To understand the practitioners' perceptions of human values (**RQ3.1**), the interviews involved presenting 58 human values from Schwartz's model. It was observed that the first three participants were overwhelmed by the number of values being presented. Therefore, it was decided for the remaining interviews to only present a set of six values chosen randomly that included both well-known software engineering (e.g. privacy, security, and efficiency) and lesser-known ones (e.g. independence, wealth, and sense of belonging). Afterwards, the interviews presented some examples of the consideration of human values during software development. Then, the interviews probed whether the participants had similar experiences in their software development activities.

The analysis of the interviews indicated that the participants did not understand some values well. The participants also argued that they had already considered some of the values during software development. It was also found that some values were considered more important than others. The remainder of this section describes these findings in more detail.

Some values were not well understood by the practitioners. Many participants asked for an explanation for some values. The values that they questioned were the ones that are less known in software engineering, such as achievements, capable, or pleasure. A developer mentioned:

💬 ‘*I am a bit unsure about this area of the achievements and capable means ...*’. (P01 – Developer)

To explain the values that the participants asked about, the interview used definitions from Schwartz’s model [1]. After the explanation, some participants tried to interpret those definitions in software engineering contexts. For example, a developer attempted to relate wealth with development activities:

💬 ‘*I would say wealthy it is a bit confusing ... for example, in terms of development activities wealthy can be treated as [paused] so you are writing something, you programmed something, and you were unnecessarily adding something which is going to cost something from the user.*’. (P05 – Developer)

Some human values are already considered during software development. Some of the practitioners argued that some human values were already considered during the development of their applications. These values were not only the ones that are well known in software engineering, such as privacy or accessibility, but also the lesser-known ones, such as inclusiveness, independence, and sense of belonging. For example, a system analyst mentioned that they developed their project by taking into account the regions, languages, cultures, and education of their users:

💬 ‘*We may develop software for a particular region, a particular crop, maybe a particular season. We integrate all these things during design. ... We consider their language,* *their education, their culture, their financial capacity, how they can afford.*'. (P04 – System Analyst)

Some values are more important than others. The analysis of the interviews showed that many participants agreed that human values need to be considered in software development. A UI designer suggested that a software developer should focus not only on the functionality but also on the user of the application:

💬 '*It should not be only functional, it should be human also, because, at the end of the day, humans will use this software, right?*'. (P08 – UI Designer)

Some practitioners also believed that some human values are more important than others depending on the nature of the application being developed. For instance, a UI designer mentioned that if an application is made for the elderly or people with disabilities, then accessibility is more important than the other values:

💬 '*... Some project, accessibility will be the number one priority; like if you say projects, with disabled people or aged people, their accessibility should be the number one.*'. (P08 – UI Designer)

However, in general, many practitioners highlighted that some values are always more important regardless of the application functionality. These values are those that are well known in software engineering, such as privacy and security.

> Many participants deemed human values to be important in software development. However, their understanding of values was limited to those well known in software engineering.

5.3.5.2 Practitioners' Perspectives on Benefits of a Human Values Dashboard (RQ3.2)

To understand who will receive the benefits of the dashboard and what the benefits are (**RQ3.2**), the dashboard prototype was demonstrated to the participants. Then, the interviews asked for the participants' opinions on the prototype. The analysis of the interviews suggested that the identification of human values can generally benefit all

Table 5.2: Roles and benefits of the dashboard for those roles

Role	**Benefit**	**Soft. Dev. Phase**
All Roles	Align which values are necessary in the project	Inception
	As a knowledge base for the next stage or future software development	End of development
Project Manager	Support value-based task prioritisation and allocation in project management	Sprint planning
	Monitor the progress of each values-related issue/task	End of Iteration
	Communicate values with stakeholders	Release
Requirements Engineer	Identify necessary values before starting the implementation	Requirement
	Ensure identified values are administered correctly	Testing
System Analyst	Inform which values are necessary to be taken into consideration	Analysis and design
Developer	Support decisions in selecting values-specific tasks	Sprint planning
	Ensure which values are addressed during development	Implementation
Tester	Ensure all values-related issues/tasks are addressed before release	Testing
Product Owner	Support decisions in which values are the priority in software	Inception and release
General Audience (e.g. in open-source projects)	Provide information about the focus of values of a software project	Any phases

roles in software development. Table 5.2 presents the roles and the possible benefits of human values identification for these roles. These roles came from the participants themselves. The benefits spanned from the initial stage to the later stages of software development.

In the initial stages of a project's life cycle, being able to detect values potentially helps the whole development team (i.e. **all roles**) become aware of which values need more attention in the project:

💬 '*So from the beginning, everybody will be in line that on this project, security is our most important concern.*'. (P08 – UI Designer)

During development ('*mid-flight*'), the identified values in the dashboard can help a **project manager** to allocate and prioritise tasks based on values:

💬 '*Well, management would use this to see what needs to be done. ... Decision on what to do is done by the management. So, they would use this to prioritise what needs to be done first* '. (P06 – Developer)

Furthermore, they can potentially help **requirements engineers**, **system analysts**, **developers**, and **testers** become aware of which values need to be addressed in their respective tasks. Information about values also helps a **project manager** to communicate with stakeholders (e.g. the product owner). The **product owner** can then make an informed decision about the values' priority and the focus of the project:

💬 '*That is basically what it is up to the program owner to decide the direction the project will take. If [they] get all the fire because the application is not stable, probably we want to focus on stability*'. (P02 – System Analyst)

Even after the project development ends, the values information in the software development artefacts is still beneficial as a knowledge base for future software development. For instance, a value identified in an issue could inform **the development team** on how to address similar cases in the future:

💬 '*And as I mentioned, these kinds of things also useful for learning for future projects.*'. (P09 – Developer)

The interviews also revealed the dashboard's benefits for open-source projects. Because the development artefacts are publicly available, **general audiences** who are interested in an open-source project could use the dashboard to determine which values have become the focus of a project. For example, a **user** who wants to be a **contributor** could figure out the extent to which a project cares about a particular value:

💬 '*... And then when it comes to a person who is outside of a repo, ... they will also get an insight on this. You [are] like, this is very informative and from a perspective of a person who wants a clear summarisation on what is happening inside a repository, and what are the areas that it is being the main focus*'. (P07 – Developer)

A values dashboard is considered useful for various roles in various phases of software development. It can be used to determine values-driven priorities in a project and raise the development team's awareness of values.

5.3.5.3 Practitioners' Suggestions on Suitable Artefacts (RQ3.3)

To answer the question of which artefacts are suitable for the dashboard (**RQ3.3**), the interviews presented some examples of how values are discussed in a GitHub Issue. The interviews then probed the participants with other artefacts that are generated during software development, and then the interview asked which artefacts they thought were suitable for the identification of values.

Table 5.3: Potential artefacts for human values identification
* denotes more suitable

Artefact	**Soft.Dev. Phase**
Market research documents	Beginning of the project
Requirements documents*	Requirements
Design documents	Analysis and Design
Features specification documents	Analysis and Design
Issue discussions*	Implementation
Pull request discussions	Implementation

Table 5.3 shows the software development artefacts that the participants thought were suitable as a source for the identification of values. These artefacts are generated in different phases of software development. Some of them are not necessarily created in every software project. It depends on how the development team manages the project.

For instance, an open-source project may not consider market research documents as necessary. On the other hand, some artefacts, such as issue discussions, are generated when the development team contributes to the software repository.

Reasons for selection. The analysis revealed that many participants chose the artefacts in Table 5.3 for two main reasons. ***The first reason was that values can be identified within the artefacts***. For example, a UI designer believed that **market research documents** that are written before application development should capture the values of the users:

> 💬 '*So, it will be in the research report, and that will honour the values of the user.*'. (P08 – UI Designer)

Other artefacts, such as **pull requests**, have a discussion functionality that captures the conversations between stakeholders. A developer mentioned that it is possible to identify values in these conversations as well:

> 💬 '*So, pull requests are the areas that we are having most of the conversations as well. ... Like if we are having pull requests with security issues all the time.*'. (P07 – Developer)

The ***second reason*** many participants chose those artefacts was because ***the artefacts are used and referred to during software development***. Many participants agreed that identifying values in these artefacts would support them in addressing values in software development. For example, a developer mentioned that the identification of values in **issue discussions** would benefit them:

> 💬 '*[The repository] is the place where most developers, especially us, [are] busy with almost the day, every day. So, if we see this kind of issue tracking, that kind of thing happening in the [repository] site, I think it would be, especially for developers, it would be good.*'. (P05 – Developer)

Another practitioner also mentioned that the identification of values in **design documents** could assist in planning the software development:

> 💬 '*So, at the design level, we can target those areas where the accessibility issues can be pointed out so that the things are more planned accordingly in terms of accessibility.*'. (P10 – Project Manager)

Suitability for the identification of values. The interviews also asked the practitioners if some artefacts were ***more suitable*** as the targets of values identification. Some practitioners argued that **requirements documents** are more suitable. The reason for choosing this artefact was that these documents are created at the initial stage of development such that the identification of values can inform the next development stages as well:

💬 '*I think it will be better to run this on requirements because then we can identify everything at the initial stage.*'. (P13 – Developer)

However, a developer remarked that a bias might exist in the requirements documents because they are usually developed by only one person or a few people:

💬 '*... Like for the requirements and stuff, it will be most probably one person who is getting involved. ... That will be pretty biased.*'. (P07 – Developer)

Another more suitable artefact was **issue discussions**. A reason for this opinion was that such discussions are the places where stakeholders and contributors discuss different aspects of an application:

💬 '*Actually, I would say it would be issues and the public's comments. ... those are the basis that people engage from different aspects.*'. (P07 – Developer)

Furthermore, the identification of values in this artefact would help developers identify the impacts of an issue that needs to be addressed:

💬 '*... Until that point, whatever we did, is there any sort of violation, is there any sort of lacking that we may be unconsciously generated or produced? So, we can take that from there. ... We can concentrate, we can fix that before the release, for example*'. (P05 – Developer)

A wide range of software development artefacts can be used as a data source for a values dashboard. However, many participants mentioned that requirements documents and issue discussions are the most suitable sources.

5.3.5.4 Practitioners' Suggestions on Requirements for a Values Dashboard (RQ3.4)

To answer the question of what is needed for a human values dashboard to be helpful in software development (**RQ3.4**), the prototype was used as a trigger for a discussion. The discussion concerned what is required for a human values dashboard to support various roles in conducting their software development activities.

TABLE 5.4: Proposed requirements for the dashboard

Requirements	
R1	The identification of values in the dashboard shall be conducted automatically.
R2	It should maintain the traceability between the identified values and their artefact source.
R3	It shall allow the development team to determine the values priority of a project.
R4	It shall display the artefacts based on the values priority determined in a project.
R5	It shall inform the latest update on the artefacts where values are identified in a project.
R6	It shall provide different views for various roles to support addressing values in software development.

Table 5.4 shows what the practitioners believe is required for the dashboard. Their suggestions were formulated during the analysis as requirements using imperative words [182]. For the first requirement (**R1**), some participants believed that it would be helpful if the identification of values were conducted automatically. This functionality would reduce their effort in manual identification and provide third-party opinions on the work:

💬 '*I know in [repository], they also have worked with, try to tag issues and categorise them manually, but that often either is forgotten or it somehow fails.* '. (P01 – Developer)

Surprisingly, although some participants were aware of the limitations of automated approaches, they were still interested in using it:

💬 '*Also, even though machine learning is not always correct, you might also find that it sees things that you do not.*'. (P01 – Developer)

For the second requirement (**R2**), some participants asked for navigation to the artefacts where values are identified:

💬 ‘*If we have a report mentioning all the details, at least the file where it is happening right now would be good for us.*’. (P05 – Developer)

This functionality would particularly helpful if the dashboard and the artefacts were in different systems.

Many participants believed that some values are more important than others. Therefore, they requested a functionality to specify which values they consider important for their project (**R3**). This functionality would help them to support value-driven decision-making, such as by prioritising which values that they want to address:

💬 ‘*As I said, I want to prioritise security number one, visibility number two, pleasure number three, wealth number four. This is very important.*’. (P08 – UI Designer)

In **R4**, many participants wanted the dashboard to display the artefacts based on the priority of the values determined for a project. This functionality would help them to focus on specific values and reduce information overload:

💬 ‘*But as I said before, I like the idea of adding the feature that you could pick, which tag [of values] you are interested in and then only see them.*’. (P06 – Developer)

A values dashboard must also inform the development team if there is a change to an artefact where values are identified (**R5**). This information will keep the team up to date with the current status of values identified throughout the application development. This functionality could be implemented by displaying the last activity or sending a notification when there is an update on an artefact:

💬 ‘*I mean, you could also add a notification. It is just my guess, I mean, if it is a security thing that has been notified somewhere, or somebody wrote something that flags a security issue, you would want to know, put that right away*’. (P06 – Developer)

Many participants agreed that a values dashboard should cater to various roles in software development (**R6**). The multiple views that were provided in the prototype (Figure 5.4) could support these roles in addressing values during software development.

For example, the summarised values overview (**OV**) provided in the prototype (Figure 5.3) informs a project manager about the state of values identified in a project before going into the details:

💬 ‘*As a team leader or a project manager, I would prefer the second one [**OV**] because I will be going through the tickets all the time. ... So, I will just get an overview of what I am working on.*’. (P07 – Developer)

The list view or **LI** (Figure 5.4a) would help system analysts, developers, and testers to ensure values are incorporated in the project:

💬 ‘*And after that, they will also use this information [**LI**] for their QA [quality assurance]. And they said, okay, we have to test this thing. Okay. Is this a security issue? Is accessibility issue or anything else? They can use all this information.*’. (P11 – Developer)

Testers can use the timeline view or **TM** (Figure 5.4b) to monitor the artefacts where values are identified before conducting their tasks:

💬 ‘*For a tester, I would say a timeline is best. ... They can use the timeline to see what issues are currently in progress and what are closed and ready to test.*’. (P13 – Developer)

The first two requirements (R1 and R2) help the development team to identify values and their corresponding artefacts efficiently. The following two requirements (R3 and R4) utilise values as prioritisation criteria in the development activities. Meanwhile, the remaining requirements, R5 and R6, are in line with the benefit of the dashboard in promoting the awareness of values among software development teams.

Six high-level requirements were suggested for a human values dashboard. These requirements were considered to help various roles to address values during software development.

5.4 Dashboard Development Stage

This stage involves developing a human values dashboard as a proof of concept. The results of the exploration stage were considered during the development of the dashboard. The development of the dashboard used the views from the prototype (Figure 5.4) as a base. This stage started with designing the components and functionality of the dashboard. Then, the dashboard was implemented and populated with an artefact from open-source projects hosted in GitHub.

5.4.1 Analysis and Design

The analysis considered the practitioners' perspectives on what is required of a human values dashboard from the exploration stage (Table 5.4). The analysis of those high-level requirements is described as follows:

R1 **The identification of values in the dashboard shall be conducted automatically**. To address this requirement, a human values detector is used as a component in the back end of the dashboard. To support automatic detection, the human values detector utilised the machine learning model that was developed in Chapter 4.

R2 **The dashboard should maintain the traceability between the identified values and their artefact source**. This requirement was addressed by storing the web page URLs of the artefacts in a database. These URLs would be displayed in the front end along with the artefacts. Using this approach, practitioners could use the URLs to refer to the actual location of the artefacts in the repository.

R3 **The dashboard shall allow the development team to determine the values priority of a project**. The machine learning model from Chapter 4 is used to detect the presence of human values in artefacts. At the time, this model could only detect the presence of any human values without specifying the specific values (e.g. privacy or inclusiveness). Because of this limitation, it was assumed that the presence of any human values is the priority in the dashboard.

R4 **The dashboard shall display the artefacts based on the values priority determined in a project**. As explained in the previous requirement, the presence of any human values was assumed to be the priority. To address this requirement, the dashboard provided a filtering mechanism in the front end. This filtering allows the dashboard to display only the issues that had been identified to have values present. To inform the latest update on the artefacts, the dashboard displayed the date and time the artefacts were reported and closed. The dashboard also has notifications to inform users when the human values detector found human values in an artefact.

R5 **The dashboard shall provide different views for various roles to support addressing values in software development**. In the dashboard prototype (Figure 5.4), the dashboard provided three views for various roles in software development. The development of the dashboard included these views with some adjustments based on the availability of the artefacts (i.e. issue discussions) and the capability of the human values detector (i.e. in detecting whether any human values were present).

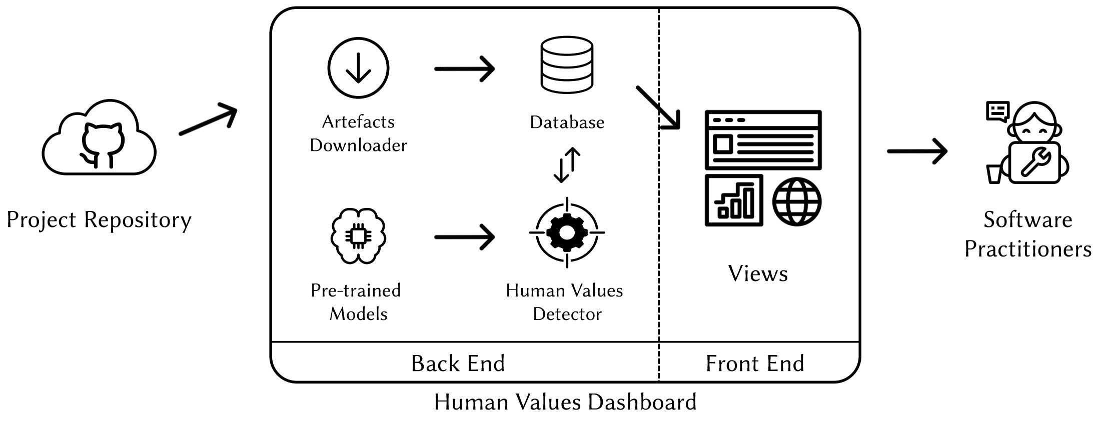
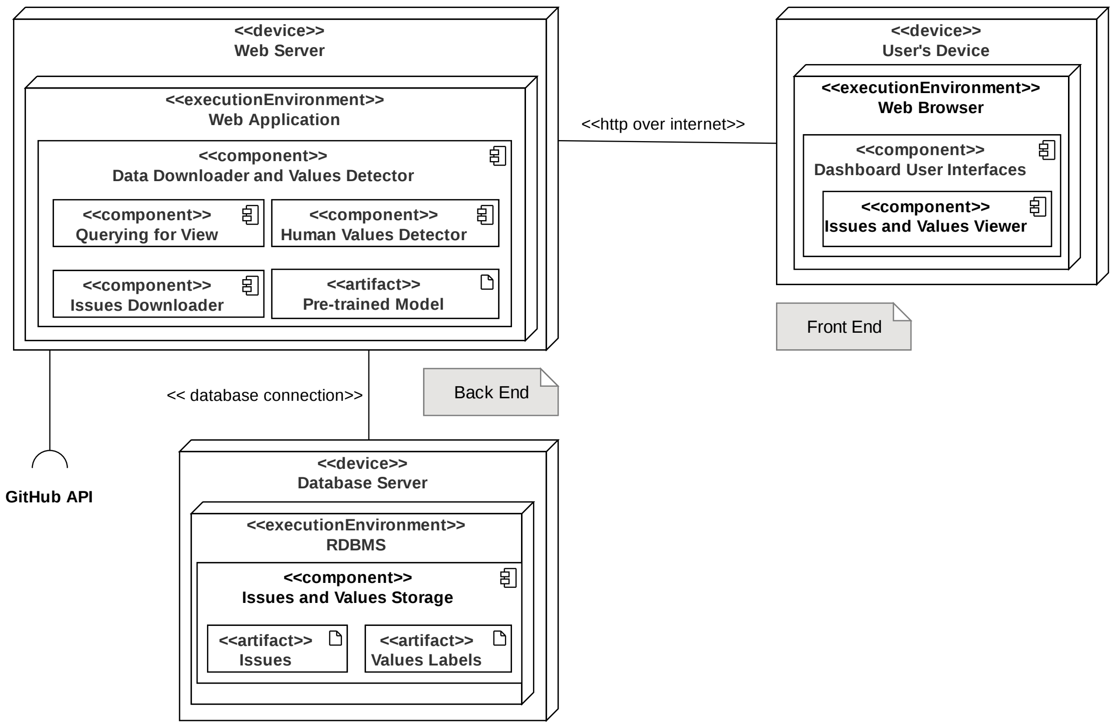


FIGURE 5.5: The human values dashboard and its components

After considering the high-level requirements from the practitioners in the exploration stage, this stage continued with designing the components of the human values dashboard. Similar to the prototype, the human values dashboard was designed to have a back end and a front end. The back end provides an automated downloading of artefacts from project repositories and automated labelling of human values in the artefacts. The dashboard's front end provides three views similar to those on the previous prototype with some adjustments based on the practitioners' suggestions in the exploration stage. Figure 5.5 shows the components of the human values dashboard. The first component in the back end, the **artefacts downloader**, allows development teams to specify repositories' URLs and download the corresponding artefacts. The downloaded artefacts are stored in a **database**. The **human values detector** could then be used to automatically detect the presence of human values in the downloaded



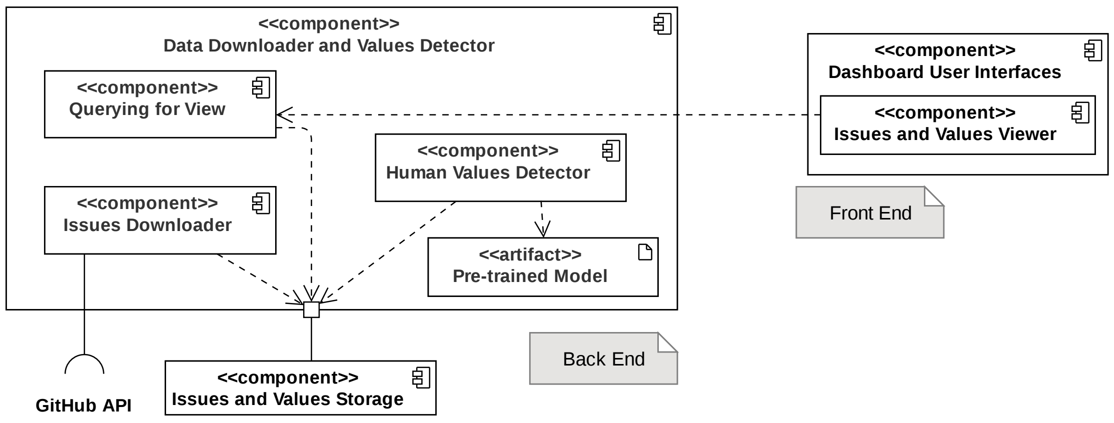
<<device>> Web Server
<<executionEnvironment>> Web Application
<<component>> Data Downloader and Values Detector
<<component>> Querying for View
<<component>> Human Values Detector
<<component>> Issues Downloader
<<artifact>> Pre-trained Model
<<http over internet>>
<<device>> User's Device
<<executionEnvironment>> Web Browser
<<component>> Dashboard User Interfaces
<<component>> Issues and Values Viewer
Front End
<< database connection>>
Back End
GitHub API
<<device>> Database Server
<<executionEnvironment>> RDBMS
<<component>> Issues and Values Storage
<<artifact>> Issues
<<artifact>> Values Labels

FIGURE 5.6: Deployment diagram of the human values dashboard

artefacts. It uses **pre-trained models** from the human value detection experiments (Chapter 4) and stores the results in the database mentioned above. The **views** in the front end provide visualisations of the detection results and their corresponding artefacts.

After conceptualising the components of the human values dashboard, the analysis continued by developing the deployment, component, and sequence diagrams for the dashboard. The deployment diagram shows *'the communication links between the physical machines and process'* [183]. Meanwhile, the component diagram shows the dependency and interaction between components of a system [183]. These two diagrams were used to model the physical hardware and components of the dashboard and their interactions. Additionally, the sequence diagram was used to model the issues download and human values detection.

Figure 5.6 shows the deployment diagram of the human values dashboard. The back end of the dashboard, that consists of the querying for view component, the issue downloader and the human values detector, is deployed in a web server. This web server is connected to a database server that stores the issues and their human values labels. The web server is connected to the GitHub API for issues download. The front end of the dashboard provides a viewer to to display the issues and their values detection results. The front end of the dashboard could be accessed using a web browser on the user's device.



<<component>> Data Downloader and Values Detector
<<component>> Querying for View
<<component>> Human Values Detector
<<component>> Issues Downloader
<<artifact>> Pre-trained Model
<<component>> Dashboard User Interfaces
<<component>> Issues and Values Viewer
Front End
Back End
GitHub API
<<component>> Issues and Values Storage

FIGURE 5.7: Component diagram of the human values dashboard

Figure 5.7 shows the component diagram of the human values dashboard. The issues downloader component in the back end connects to GitHub API for downloading issues and stores them in the issues and values storage. The human values detector component uses a pre-trained model to detect the presence of human values in issues and stores the results in the issues and values storage. The dashboard's user interfaces provide a viewer to display the issues and values using the querying for view component in the back end.

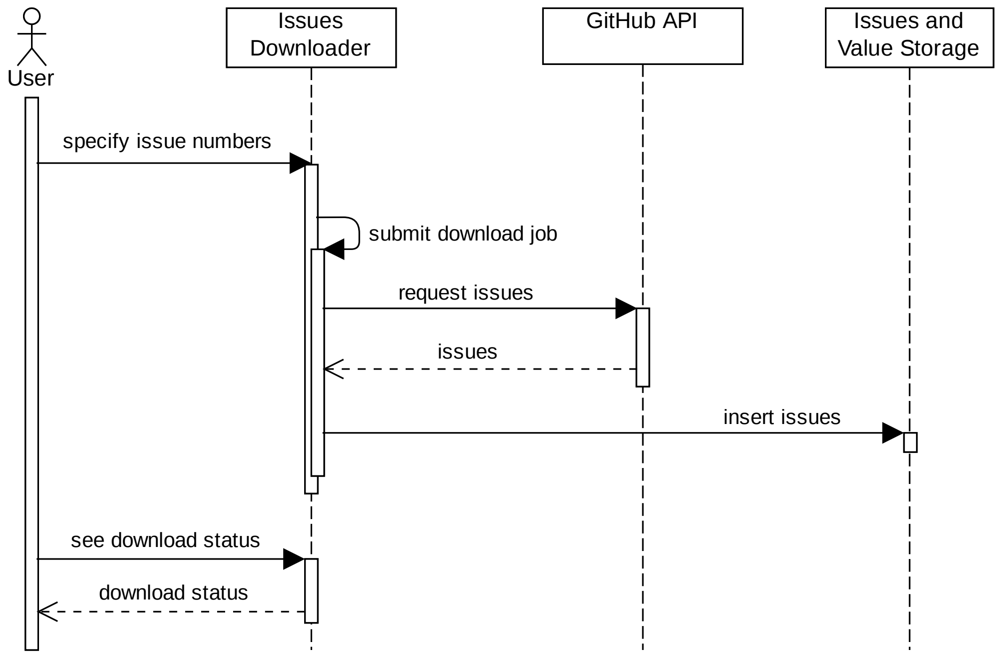


FIGURE 5.8: Sequence diagram of downloading issues from GitHub

Figure 5.8 shows the sequence diagram of downloading issues from GitHub. The process starts when users specify the range of issue numbers that they want to download. Issue numbers are the numbers of the first to the latest issues incrementally assigned by GitHub for each issue submitted to the project. Because the downloading process may take a while to complete, the issues downloader then submits a download job to be executed as a background process in the web server. This download job requests issues from the GitHub API and inserts the downloaded issues to the database. The users then can see the status of the download job that they submitted.

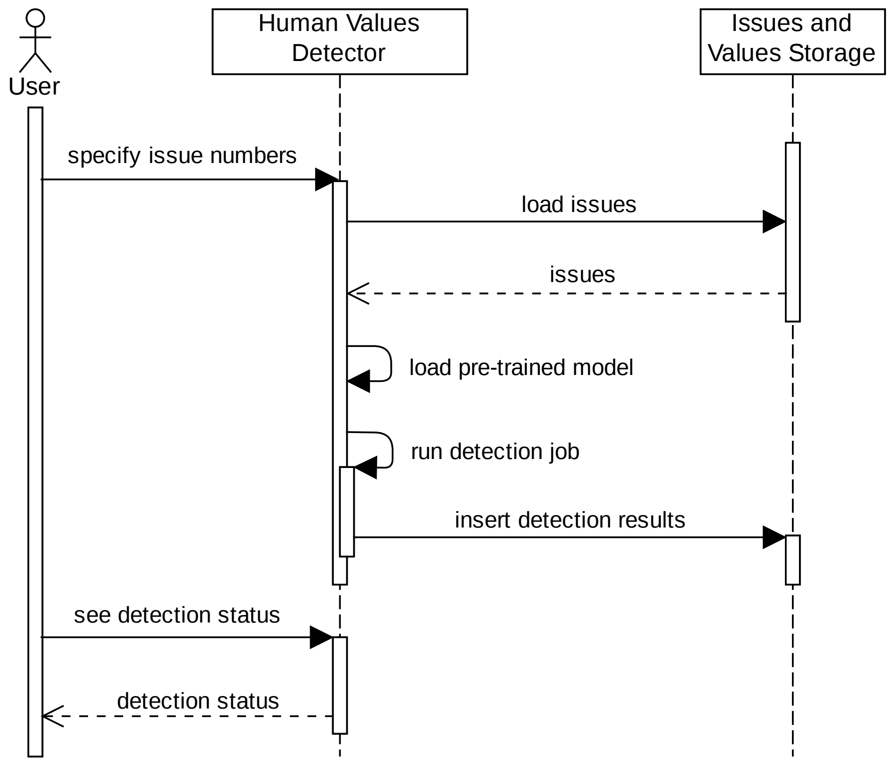


FIGURE 5.9: Sequence diagram of detecting human values in issues

Figure 5.9 shows the sequence diagram of detecting human values in issues. The process starts with users specifying the range of issue numbers that they want to be identified by the human values detector. Then, the human values detector loads the requested issues from the database. Afterwards, the human values detector loads the pre-trained model and submits the detection job as a background process in the server. This background process runs the detection and inserts the detection results to the database. Later on, the user can see the status of the detection job.

The conceptual model design for the database component of the dashboard is presented in Figure 5.10. The database consists of several entities, described as follows:

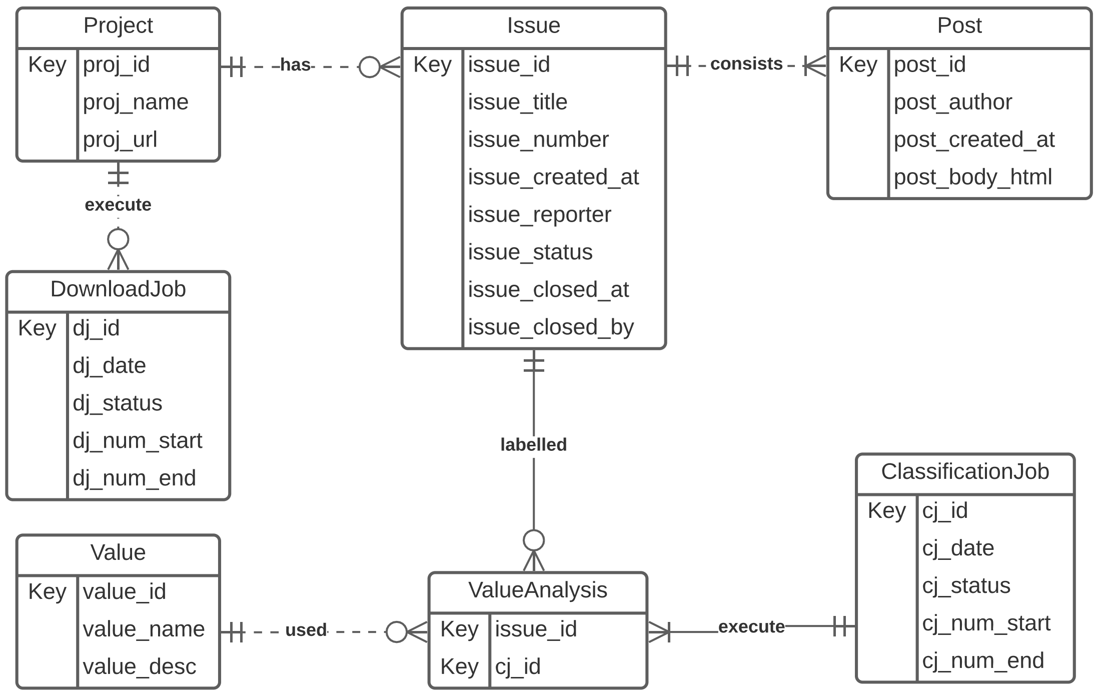


FIGURE 5.10: Database Conceptual Model for the Dashboard

- The **Project** entity stores the software project information, including the name and the location of the project in the repositories. This entity was necessary because it is common for a company or an organisation to have more than one software projects. This would also allow for comparisons between projects.
- The **Issue** entity stores the issue information that belongs to a project, including the title, reporter, and status of the issue.
- The **Post** entity stores information about posts in an issue. This information includes the content of the post that would be examined for human values.
- The **Value** entity stores human values information, such as the name and description of those values. However, this version of the dashboard only used this entity to indicate whether human values were present.
- The **DownloadJob** entity stores the jobs that were submitted to the server to download the issues from project repositories.
- The **ClassificationJob** entity stores the jobs that were submitted to the server to detect the values in the issues.
- The **ValueAnalysis** entity stores the results of the values detection in each issue.

5.4.2 Implementation

The dashboard was implemented using Flask[3], a web framework written in Python. A Python-based framework was chosen to facilitate the integration of the human values detector into the back end. The implementation of the dashboard focused on the use of issue discussions, as suggested by the empirical findings of Chapter 3 and the practitioners in the exploration stage (Section 5.3.5.3).

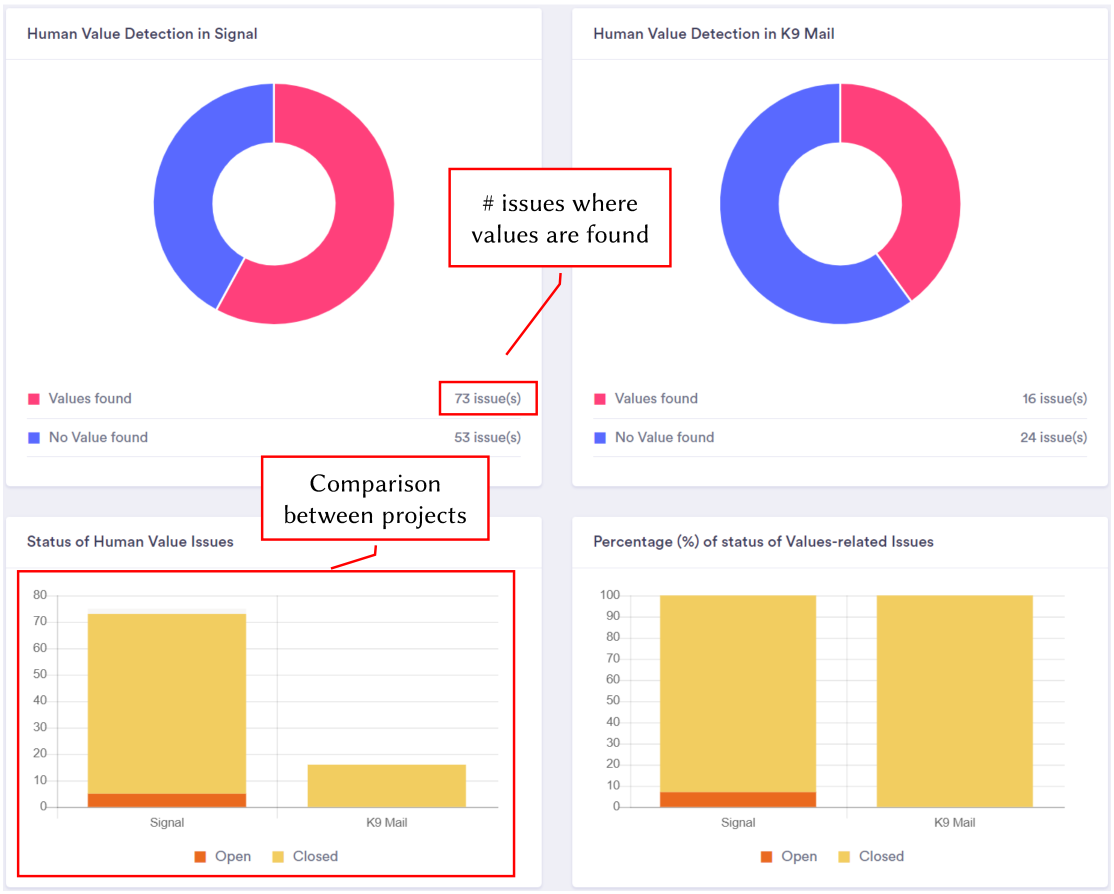
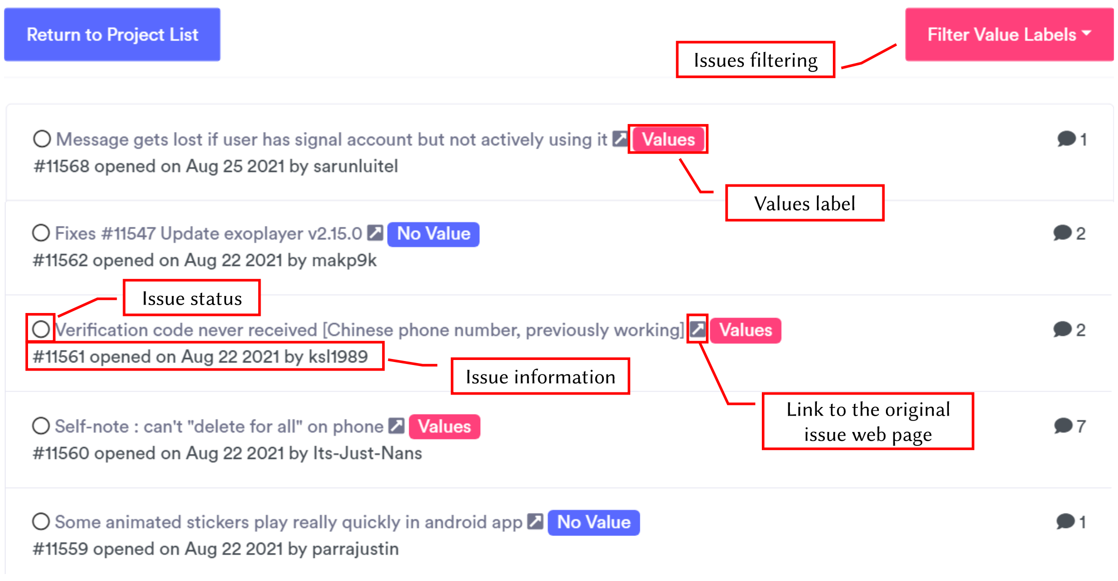

Human Value Detection in Signal
Human Value Detection in K9 Mail
# issues where values are found
Values found
73 issue(s)
No Value found
53 issue(s)
Values found
16 issue(s)
No Value found
24 issue(s)
Comparison between projects
Status of Human Value Issues
Percentage (%) of status of Values-related Issues
Signal
K9 Mail
Open
Closed

FIGURE 5.11: Dashboard summarised overview (OV)
(Boxes in the red outline are not part of the dashboard)

The front end's views were developed using the Chart.js library[4]. The three views proposed in the exploration stage were retained because the software practitioners considered those views useful for various roles in software development. However, due to the limitation of the machine learning approaches used (Chapter 4), some adjustments were made to the three views, which are explained below:

[3]https://flask.palletsprojects.com/en/2.0.x/
[4]https://www.chartjs.org/



Message gets lost if user has signal account but not actively using it Values 1
#11568 opened on Aug 25 2021 by sarunluitel

Fixes #11547 Update exoplayer v2.15.0 No Value 2
#11562 opened on Aug 22 2021 by makp9k

Verification code never received [Chinese phone number, previously working] Values 2
#11561 opened on Aug 22 2021 by ksl1989

Self-note : can't "delete for all" on phone Values 7
#11560 opened on Aug 22 2021 by Its-Just-Nans

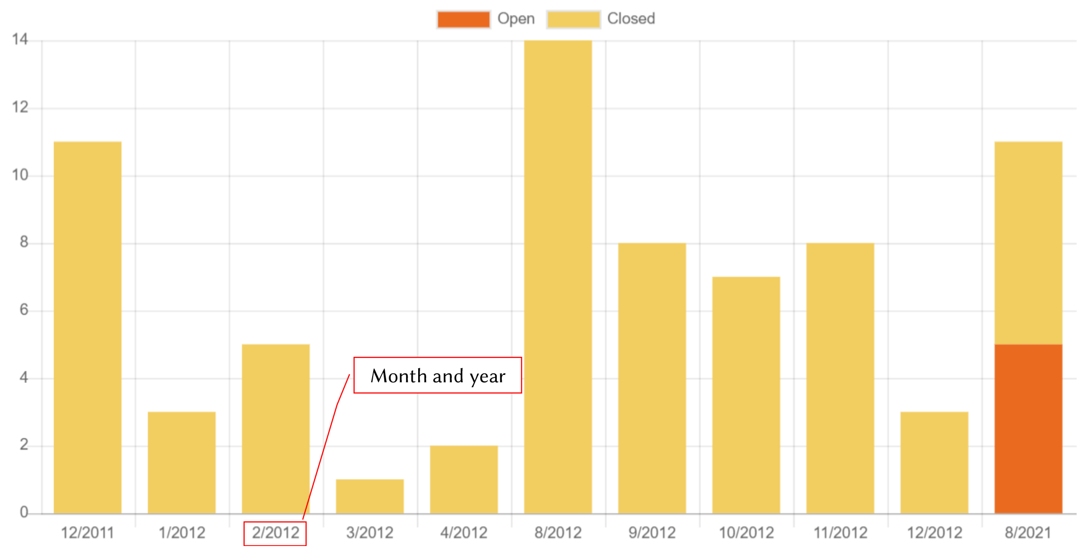
Some animated stickers play really quickly in android app No Value 1
#11559 opened on Aug 22 2021 by parrajustin

FIGURE 5.12: Dashboard values-labelled list (LI)
(Boxes in the red outline are not part of the dashboard)

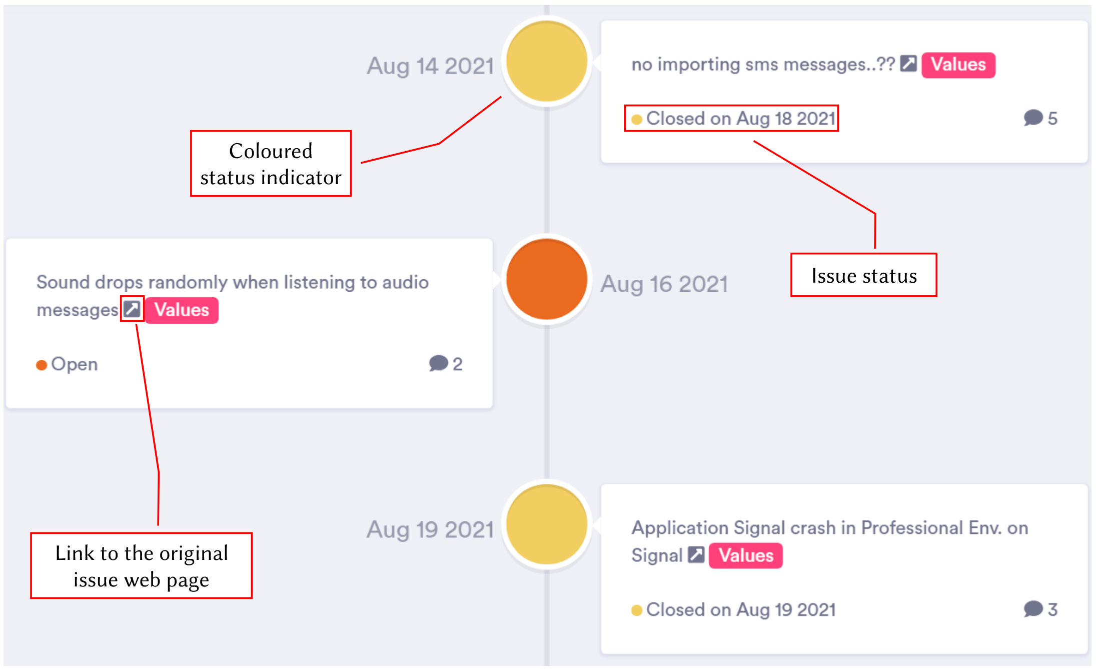
1. **Summarised values overview (OV)**. This view displays the numbers of issues where human values were present and not present in a pie chart. To allow for comparisons between projects, this view provides two of these pie charts side by side. This view also displays the number of issues where values are detected based on the status of the issues (i.e. open or closed). This view aims to provide insights on the number of values-labelled issues that need to be addressed. Figure 5.11 shows the implemented OV in the dashboard.

2. **Values-labelled list (LI)**. This view shows a list of issues similar to how issues are displayed in GitHub. This view displays a label as the result of the human values detector indicating the presence of human values in a particular issue. An issue is labelled with either the 'Values' label if the human values detector finds human values in that issue or the 'No Value' label if the human values detector does not find human values in that issue. This view also includes information on when the issue was opened, by whom, whether it is open or closed, and the number of posts. There is a filtering capability for the issues and a link to the original webpage of the issue, as suggested in the exploration stage (Section 5.3.5.4). The filtering feature allows software practitioners to view the issues wherein human values are identified. Figure 5.12 shows this view in the dashboard.

3. **Values-labelled timeline (TM)**. This view shows the issues chronologically based on the date the issues were opened. This view displays a bar chart showing the monthly number of open and closed issues where values are present (Figure 5.13a).



(A) Monthly issues


Aug 14 2021
Coloured status indicator
Aug 16 2021
Issue status
Aug 19 2021
Link to the original issue web page

no importing sms messages..?? Values
Closed on Aug 18 2021 5

Sound drops randomly when listening to audio messages Values
Open 2

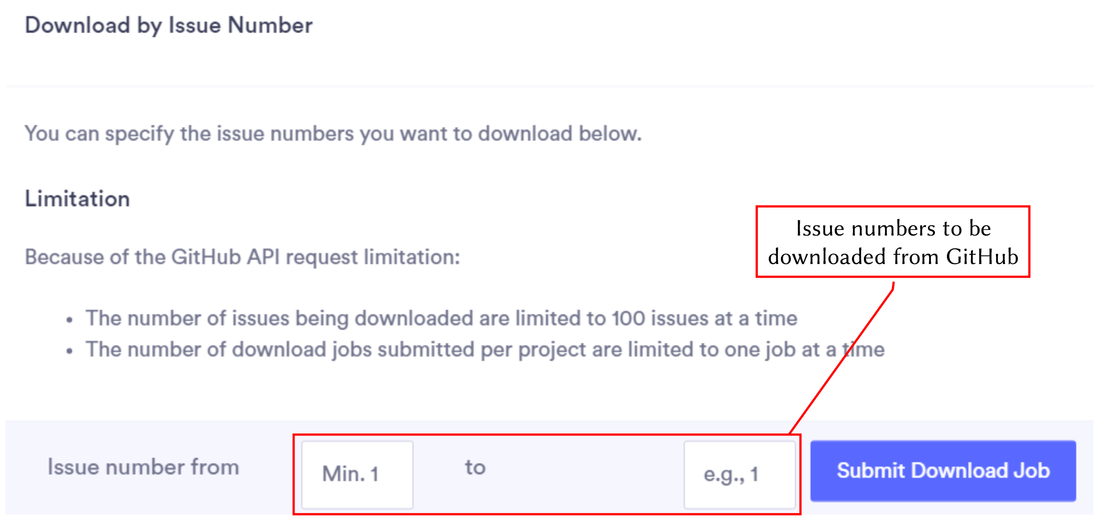
Application Signal crash in Professional Env. on Signal Values
Closed on Aug 19 2021 3

(B) Timeline view of the issues

FIGURE 5.13: Dashboard timeline (TM)
(Boxes in the red outline are not part of the dashboard)

At the bottom, this view presents a timeline of issues where values are present, with two different colours to indicate whether the values are open (orange) or closed (yellow) (Figure 5.13b). Figure 5.13 shows both visualisations in the timeline view.

In the back end, there are two interfaces used for the artefacts downloader and human values detector components. The first interface allows the development team to



Download by Issue Number

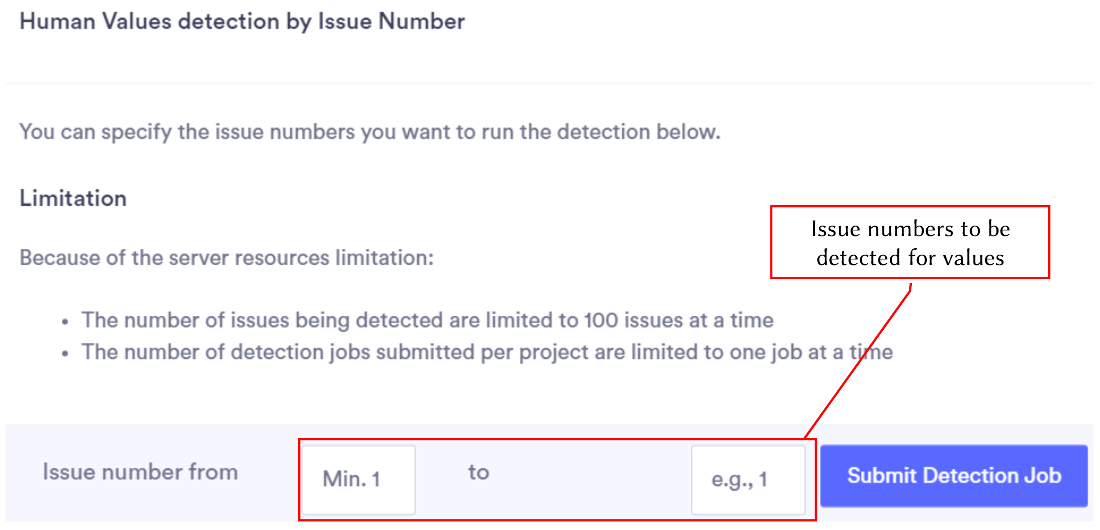
You can specify the issue numbers you want to download below.

Limitation

Because of the GitHub API request limitation:

- The number of issues being downloaded are limited to 100 issues at a time
- The number of download jobs submitted per project are limited to one job at a time

(A) Issue downloader



Human Values detection by Issue Number

You can specify the issue numbers you want to run the detection below.

Limitation

Because of the server resources limitation:

- The number of issues being detected are limited to 100 issues at a time
- The number of detection jobs submitted per project are limited to one job at a time

(B) Issue detector

FIGURE 5.14: User interfaces for the dashboard's back end

specify a range of issue numbers that they want to download from the project repository (Figure 5.14a). The issue downloader was implemented using GitHub API and the github3.py[5] library. The second interface enables the development team to specify a range of issue numbers to be detected by the human values detector (Figure 5.14b). This range of issues is then used as a parameter to run either the download or detection as a background task on the server. Figure 5.14 shows those interfaces for the dashboard back end. In this version, the developed dashboard limited download and detection to 100 issues at a time. Users could repeat the download and detection process. This limitation could be easily removed by subscribing to the GitHub API services and increasing the server resources used to analyse the issues. Two open-source

[5]https://github3.readthedocs.io/

projects, namely, Signal Android and K9 Mail, were used as examples in the feedback stage. This human values dashboard is available online[6].

5.5 Feedback Stage

This stage involved presenting the human values dashboard developed in the previous stage to gather feedback from the software practitioners. It focused on obtaining feedback from the practitioners and determining whether the human values dashboard is useful. Additionally, because the human values dashboard used an automated technique to detect human values (i.e. a human values detector – Figure 5.5), it is also necessary to understand the practitioners' opinions on the performance of the detector. Therefore, for this feedback stage, the following sub-research questions were defined:

RQ3.5 To what extent do practitioners find the human values dashboard useful?

RQ3.6 How do practitioners perceive the performance of the automated human values detection?

In addition to the answers to these sub-research questions, the participants' suggestions were also collected to improve the dashboard in the future.

5.5.1 Interview Guide Development

An interview guide was developed for the feedback interview to obtain the practitioners' feedback and suggestions on the human values dashboard. This semi-structured interview consisted of two parts. The first part of the interview asked for the demographic information of the participants, such as their roles and experiences. The second part of the interview started with an introduction on human values concepts and a demonstration of the values dashboard. Then, the second part continued by asking the practitioners' opinions regarding the dashboard and its usage. This part also asked questions related to the human values detector, e.g. a component to automatically detect the presence of values. This interview guide was discussed with the supervisory team and other group members, resulting in several suggestions. Several adjustments were made to the interview guide by incorporating these suggestions.

[6]https://dashboard.ovis-classification.cloud.edu.au

5.5.2 Data Collection

Participant selection criteria. The selection criteria used in this stage were similar to those used in the exploration stage (Section 5.3.3). This interview sought for practitioners who had been involved in a software development project and were familiar with artefacts from software repositories. This stage involved a new set of participants, i.e. practitioners who had not been involved in the exploration stage. This choice was made to investigate whether different practitioners viewed the human values dashboard as acceptable.

Participant recruitment. The recruitment of the participants was done by inviting contributors of open-source projects hosted in GitHub via email. The email addresses of these contributors were made available by them on their GitHub pages. Interested participants were asked to reply to the email invitation. An invitation to participate was also published on the group web page and LinkedIn. In addition, our colleagues were asked to broadcast the invitation to their networks. Interested practitioners were asked to inform us of their emails through our colleagues or fill out an online form on the group web page. These candidates were then contacted via email to request their consent and arrange an interview session.

Profile of the participants. Table 5.5 shows the profiles of the participants for the feedback interview. The participants mostly had developer roles. Most of them had less than 10 years of experience in software development; 4 had 10 or more years of experience. The participants were mostly located in Asia, with one participant located in Europe and another in Australia.

TABLE 5.5: Profile of the participants

Code	Role	Experience (years)	Location
P16	Developer	3	Asia
P17	Project Manager	16	Asia
P18	Developer	4	Asia
P19	Developer	8	Asia
P20	Developer	5	Asia
P21	Developer	6	Australia
P22	Software Architect	10	Asia
P23	Developer	12	Asia
P24	System Analyst	16	Europe
P25	Developer	6	Asia

Interview protocol. Before the interview session, the participants were requested to read the explanatory statement and fill out the interview consent form. The interview consisted of two parts. The first part focused on obtaining the participants' professional backgrounds. The second part started by explaining human values' concepts and the study. This was followed by a demonstration of the developed human values dashboard (Section 5.4.2). Then, the participants were asked for their perspectives on the usefulness of the dashboard (e.g. *'Would the dashboard be useful for you in software development?'*) and on the performance of the values detection (e.g. *'At what level is the dashboard accuracy tolerable for you?'*). This second part of the interview also asked for their suggestions and feedback for the dashboard (e.g. *'Does the information provided in the dashboard prototype sufficient to help you?'*). The interview questions for this study are available in Appendix C.

The interviews in this stage were recorded using a video conference system with the participants' permission. Similar to the the exploration phase, the number of interviews had not been set in advance. The recruitment and interviews were conducted in parallel with the data analysis until data saturation was reached [178, 179]. The convergence of answers and ideas became apparent in the data analysis after 10 interviews. The mean duration of the interviews was 30 minutes and 49 seconds. Professional transcription services transcribed all audio recordings of the interviews.

5.5.3 Data Analysis

The interview data were analysed using the thematic analysis approach [180]. Similar to the data analysis in this work's exploration stage and other previous studies [125, 181], the present author performed a large portion of the analysis, which was followed by reviews and discussions with other people. In this analysis process, the supervisory team was also consulted in the event of any doubts or difficulties. The present author started familiarising himself with the interview data by reading the transcriptions and listening to the audio recordings. Then, the present author generated codes and themes from the analysis of the transcriptions. Following this, the present author had several discussions with the supervisory team to review the identified codes and themes and determine their relations. The present author then assigned a name and definition to each theme. The resulting themes were presented to the supervisory team and the other group members for feedback. The themes were then adjusted by incorporating the feedback.

5.5.4 Results

This section presents the results of the feedback stage. First, this section describes the usefulness of the human values dashboard and the challenges of deploying it in a company. Second, this section presents the practitioners' perceptions of the human values detector. Finally, this section lists suggestions from the practitioners to improve the dashboard further.

5.5.4.1 Usefulness of the Dashboard (RQ3.5)

To understand the practitioners' perspectives on the extent to which the human values dashboard could be helpful, the interview started with presenting and providing the dashboard to the participants to explore. Then, the interview asked the participants about the usefulness of the dashboard to support their development activities. The analysis of the interviews suggested that all practitioners agreed that the dashboard could be useful for them. Some participants argued that there would be some potential challenges for the dashboard to be implemented in their company. These findings are described below.

The human values dashboard was considered useful. All participants agreed that the human values dashboard could be useful to support them in software development activities. Identifying the values present in issues would help the practitioners focus their attention on the issues, which, in turn, would ensure these issues were addressed. A developer mentioned:

💬 '*... Developers will pay their attention to that one [the **LI** view]. So, if we make sure that we have covered all possible scenarios in the issue list to take down [address] that human values in those tickets.*'. (P16 – Developer)

In addition to focusing the development team's attention, the participants believed that the dashboard's values labels would provide human values perspectives in addition to the well-known technical perspectives. A developer mentioned:

💬 '*When we look at an issue right now, so we do not think about any values aspect, like human values normally. We just think about it from a technical side usually. This would be helpful to understand there is another aspect for the ticket there.*'. (P20 – Developer)

The human values perspectives would subsequently help them prioritise their tasks. This usefulness would be apparent if there were a substantial backlog of issues:

> 💬 '*... Especially when there is a huge backlog of issues, I think it is very hard to kind of prioritise and a lot of issues get lost in the backlog, and we file it during one time and then it kind of gets lost and then it does not come up or it just that. So, if there is some sort lot of, let us say, a subjective value, let us say morals assigned to an issue. I think it would help to kind of prioritise it.* '. (P22 – Software Architect)

Some participants believed that the dashboard could also inform the team's performance. The dashboard summarised overview (Figure 5.11) could also be used to compare the progress between projects. A developer mentioned:

> 💬 '*I have a company, and I am running several projects. Okay. So, I can measure the team performance by this tool easier, and also the complexity of the project I can understand from this.*'. (P19 – Developer)

A project manager suggested that linking values-labelled issue posts to their contributors could help identify values champions. The participant referred to a values champion as '*anyone who aligns themselves with human values in the organisation*'. The participant mentioned:

> 💬 '*In this dashboard, you can see who is the champion of these values or maybe what is the level of "do not do evil" in the discussion, inside the repository and the issue tracker.* '. (P17 – Project Manager)

Potential challenges in adopting the dashboard. When the dashboard was presented, some participants reacted by suggesting potential challenges in adopting the dashboard in their environment. A developer mentioned that a contributor might not describe the issue correctly and that this could influence the result of the human values detector:

> 💬 '*Because, in my experience, I have gone through some issues that may be the QA developers, ... I mean, QA when raising these issues, but they are not correctly describing the issue in the field.* '. (P16 – Developer)

Another challenge concerned the willingness of a company to use the dashboard. Some participants suggested some reasons that could hinder the use of the dashboard in a company. First, a company may not be familiar with the concept of human values. This situation could lead to a lack of awareness of human values in the company and the company tending to focus on the financial aspects of the business:

💬 '*Although it has some significant impact while I am developing something or not, but sometimes the management or the [project] plan, and does not bother [with] that type of issues or that type of thing. They only think about money and business.* '. (P19 – Developer)

Even if a company is aware of human values, it must decide how to address conflicting values from different users. An additional effort may be necessary to determine what needs to be done:

💬 '*These are two issues that we need to prioritise. Are (users from) China our main priority or (users from) [the] US our main priority? The domain is specific. So, how can I prioritise these two issues by these two (users)? Is it possible?* '. (P19 – Developer)

Second, a corporation could argue that the consideration of human values is not required because it is unregulated. A project manager suggested that a company itself is in a position to decide whether it wants to support the consideration of values:

💬 '*This is [an] area where the company, right now within the US or maybe international law is not compulsory. It is more like the company does assessments on their intentions, on their diversity, and so on, as a public campaign, but not regulated.* '. (P17 – Project Manager)

All practitioners in the study considered the dashboard to be useful for focusing attention and prioritising issues. However, there are some potential challenges, such as the willingness and extra efforts required from a company in adopting the dashboard.

5.5.4.2 Perceptions of the Performance of the Human Values Detector (RQ3.6)

The use of an automated approach to identify human values in a dashboard has the possibility of leading to inaccuracies. This interview stage used the term 'accuracy' to simplify the communication with the participants regarding the correct or incorrect identification of values. To understand how the practitioners perceive the automated human values detector, the interview probed the extent to which the performance of the detector was tolerable to the participants.

The analysis of the interviews indicated that many practitioners understood the possibility of inaccuracies occurring in the identification of human values. However, the level of tolerance for the accuracy varied among the practitioners. One practitioner preferred to have 90% accuracy to trust the identification results:

💬 '*To have that kind of level of trust, I think at least 90% accuracy is needed. Less than 90%, usually we do not trust the tools, we do not put any action point on the tools.* '. (P17 – Project Manager)

Meanwhile, another practitioner considered 50% accuracy to still be tolerable:

💬 '*This is a machine learning thing, so there will be some issues. It cannot give an exact solution, so I think 50 is enough and it will develop after some time.* '. (P18 – Developer)

The analysis of the interviews also discovered that all participants preferred to have false positives on the detector than false negatives. This finding meant that it was acceptable to have the human values detector identify that an issue had values present even though that might not be correct. All participants agreed that false positives were better than missing critical issues because the detector was unable to detect the presence of the values. A developer mentioned:

💬 '*It says there is no value, but actually there is a value. We can neglect this since it notifies that this has no value, and we neglect it without further investigating the issue.* '. (P17 – Developer)

Practitioners have different level of tolerance for inaccuracies in the identification of values. However, false positives were preferred over false negatives.

5.5.4.3 Suggestions for Improving the Dashboard

To obtain feedback, each view in the dashboard (Figure 5.11, Figure 5.12, and Figure 5.13) was demonstrated to the participants. Then, the participants were probed for suggestions to improve the dashboard. The present author, as the main analyst of the interview, collected the participants' feedback and suggestions on each view and the overall dashboard.

TABLE 5.6: Suggestions for the overall dashboard

No.	Suggestion	Quotation
G1	Specific human values detection	'*If you can show these (values) categories in the dashboard, I guess, it would be helpful.*'. (P20 - Developer)
G2	Issue management	'*These are my plan[s] to address the issue. So, if we can enrich this system with our own way of planning, how to handle them, that would be good.*'. (P21 - Developer)
G3	Colour customisation for values labels	'*It is in there using the red or the blue, I think I would suggest a value that would be blue and no value, but I do not know; we tend to look to red as problem.*'. (P24 - System Analyst)
G4	Progress of an issue	'*I think the most important thing for us, in progress section, because we need to plan our delivery. So, by seeing this, we can predict when can we deliver this or not.*'. (P19 - Developer)
G5	Customisation for prioritised values	'*I think there should be some sort of weight to our value because privacy may not be that important to some application[s].*'. (P22 - Software Architect)
G6	Ranking of issues based on additional criteria	'*So, I think, basically, some sort of ranking for this. I do not know how urgent or popular this issue is in this view.*'. (P21 - Developer)
G7	Indication of values violation	'*And of course, as our earlier discussion, the violation of value that is also pulled in. It has value but in a negative way. It has violation of value, not just it has value. That's also important.*'. (P17 - Project Manager)

Table 5.6 shows the feedback from the participants on the dashboard. In **G1**, some participants wanted to have the dashboard display which specific values were detected in the issues. This would provide a development team with an opportunity to address issues based on their values priorities. One participant also wanted to have functionality, such as a to-do list or a planner, to manage the issues that a developer want to

address (**G2**). Some of these functions are provided in GitHub [94]. In **G3**, it was found that each team or practitioner had a preference on the label colour. A colour customisation feature could be developed to address this suggestion. Some practitioners also suggested that the dashboard display the progress of each issue (**G4**). This information could help them to plan or predict application delivery. Additionally, they wanted the dashboard to allow them to specify which values they wanted to prioritise (**G5**). This suggestion could be addressed by having the human values detector detect the presence of specific values (e.g. privacy or longevity). In **G6**, some practitioners suggested having additional criteria for ranking the issues on top of the presence of the human values. The urgency level or the popularity of an issue could be the indicators for this ranking. In **G7**, a practitioner proposed a suggestion to indicate not only the presence of human values but also the values violations in an issue. This practitioner stated that this indication would be helpful to prioritise the issues.

TABLE 5.7: Suggestions for the summarised overview in the dashboard

No.	**Suggestion**	**Quotation**
OV1	Provide additional categorisation (e.g. issue type)	'*I mean, for the QA person, we can add some categorising, (such) as user experience, user interfaces side, and we can add such things to one category.*'. (P16 - Developer)
OV2	Reporting	'*Maybe you have this on your future plans, maybe we can add some reporting here.*'. (P16 - Developer)

Table 5.7 lists the practitioners' suggestions for the summarised overview (OV) of the dashboard. In **OV1**, some participants suggested that the dashboard should have additional categorisation based on the type of issues. An example of this categorisation could be based on the types of roles in the team that could address the issues, such as UI issues for the UI designer. A practitioner also suggested that the OV have a reporting functionality to support the decision-makers of a software project (**OV2**).

For the list view of the dashboard, the practitioners had some suggestions, which are listed in Table 5.8. For example, a practitioner suggested that the issues' assignees be displayed in the dashboard (**LI1**). A practitioner mentioned that this information would help filter out the issues that still need someone to work on them. In **LI2**, a practitioner suggested including the topics of the issues. This information would provide the development team with a quick summary of what all of the issues are about. Related to the accuracy of the human values detector (Section 5.5.4.2), a participant mentioned that it would be helpful if the confidence level of the detection are

Table 5.8: Suggestions for the list view in the dashboard

No.	Suggestion	Quotation
LI1	Assignees' information (available in GitHub)	'*I think can it have something like assigned to kind of thing there? ... I think for me it might make more sense to look into it if it is not assigned or if nobody is looking into it.*'. (P25 - Developer)
LI2	Topic of the issues (e.g. word cloud)	'*So, I have just one suggestion, which is to cluster the issues. ... But I think let's say if I want to know what are these all value issues are discussing. They might be basically three or four topics, right. So, if I want, you know your points, you know word clouds, right?*'. (P21 - Developer)
LI3	Display of the detection confidence level	'*To include the rank of the classification as a values discussion in the view here so that you can filter depending on these. ... So, if value is one, let's say, for example, it might be 90% prediction or maybe 80%, or maybe even 51%, right. So, if you show this number here, ... I will be able to give a priority to the ones that are highly rank and then just leave those 51% to look at later.*'. (P21 - Developer)
LI4	Suggestions for solving the issues	'*In your automated tool, using (the) automated tool, can you search on Google about those issues? It is better [if] you can display some more information about this.*'. (P18 - Developer)

displayed on each issue (**LI3**). This suggestion could help the practitioners prioritise issues with a higher confidence level of detection. To further help practitioners in addressing issues, one participant also requested that the dashboard search and display relevant solutions from search engines (e.g. Google) or questions-and-answers forums (e.g. Stack Overflow) for each issue (**LI4**).

In the timeline view of the dashboard, the participants provided several suggestions related to the time perspective, as shown in Table 5.9. First, some practitioners requested a customisable time range (**TM1**). The practitioners felt that this customisation would highlight recent issues depending on the frequency of issues in a project. The remaining two suggestions were related to the duration of time that had passed since an issue was first reported (**TM2**) and the duration needed for an issue to be completed (**TM3**). The former suggestion (**TM2**) would highlight an issue that had not been addressed for a period of time, while the latter (**TM3**) would provide analytics of issue completion to the project managers.

TABLE 5.9: Suggestions for the timeline view in the dashboard

No.	Suggestion	Quotation
TM1	Customisable time range	'*Something like last one month or last few months kind of thing. So that it gives me a real idea on how things are looking at real. So you can always change, you can always select all to view everything. But so as a user, when I see it, if it gave me the latest data or latest strain over the week or the month, I think it would be useful.*'. (P25 - Developer)
TM2	Duration of time that had passed since an issue was first reported	'*And it's hard to figure out which was created when, and ... even it to take the immediate action or not? Since issues (were) created yesterday, (it) might wait for one or two or weeks, but if it is already one month, then you might want to take action and look into it, at least try to solve it.*'. (P25 - Developer)
TM3	Duration needed to fix the issues	'*Duration. Yeah, yeah, yeah. From open to close. What I am telling (you) is so, if I am an administrator or someone who manages everything, what I want to look is on average, how much time is it going?*'. (P25 - Developer)

Sixteen suggestions to improve the human values dashboard were collected. These suggestions are not only for each view of the dashboard but also for the overall dashboard.

5.6 Discussions

This section highlights and discusses the findings of the exploratory and feedback stages. First, this section discusses the findings on the awareness of values. Second, it discusses the possibility of using software artefacts as the source for identifying values for the dashboard. Third, this section discusses the possibility of providing a human values dashboard for users. Finally, it discusses the limitations and challenges of the dashboard.

5.6.1 Awareness of Values

The analysis in the exploration stage showed that most participants are familiar with only a limited set of values, such as *security* and *accessibility* (Section 5.3.5.1). This finding strengthened the findings from [10], which highlighted that only a few values have been discussed in recent academic software engineering publications. This lack of awareness was also found in the feedback stage as one potential challenge to the adoption of the human values dashboard in a company (Section 5.5.4.1). One possible reason for this stems from the fact that there is a lack of understanding of these values in the software engineering context. Furthermore, participants in the exploration stage thought that the values that they were familiar with were important and believed that they had already considered these values during software development. The other values that they were not familiar with became a 'nice to have' in an application. These findings showed the need to increase the awareness of values not only of practitioners but also of companies. A possible solution could be to provide a contextualised software engineering definition for each of these values [7, 10], as presented in Section 3.3. Additionally, as suggested in the findings, a tool, such as a human values dashboard, could be utilised to introduce and increase the values awareness of companies and their development teams. These findings were in line with a previous study that suggested that a dashboard has the benefit of increasing awareness [84].

5.6.2 Practitioners' Familiarity and the Presence of Values in Issue Discussions

The finding of familiarity of the participants with human values was not fully reflected in the presence of values in issue discussions (Section 3.3.2). Some values with which participants are familiar, such as privacy or security, were frequently found in issue discussions (see Figure 3.7). Some other values with which they are also familiar, such as accessibility, were found to be as frequent as other values with which they are not familiar, such as independence or wealth. This could happen for at least two reasons. First, the nature of software projects in the empirical study of issues is different from those developed by the participants. For example, the projects analysed in the empirical study are open source, while most of the participants developed projects from their company. Most companies have their own set of important values, while the values of an open source project usually depend on their contributors. Second, the issue discussions also include comments or suggestions from users. In this case, users may post a comment related to some values that are not necessarily known by the interview participants. Comparing the familiarity of the participants and the mapping process of the value themes also showed similar traits. Privacy and security are so well known that their mapping process is quite straightforward to the Schwartz values. Nevertheless, the familiar values are on a much lower level of abstraction (i.e. more technical) compared to those that are not.

To further investigate the relationship between the familiarity of values and their presence in issue discussions, a future study could investigate values in issue discussions of a company. This investigation could then be followed by interviewing the practitioners involved in the issue discussions under study. This kind of study could capture the perceptions and opinions of practitioners that could help validate and refine the values analysis criteria.

5.6.3 Artefacts as the Source for the Dashboard

The software practitioners in the exploration stage suggested that requirements documents and issue discussions are considered more suitable for mining human values. This thesis focused on one of these artefacts, namely issue discussions, as the source for the dashboard. Future work could extend this research by investigating the presence of human values in requirements documents. If such a study could find the presence of values in requirements documents, then it could be followed by incorporating the requirements documents as another artefact source in the human values dashboard.

Based on the results of the exploration stage, a human values dashboard was developed. This dashboard labels the presence of human values in issue discussions. The feedback stage found that the participants agreed that the dashboard could be helpful to focus their attention and prioritise issues. Nevertheless, it is still possible to enhance the developed human values dashboard by adding other artefacts. Therefore, future research could investigate the presence of human values in other artefacts to incorporate them in the dashboard.

5.6.4 A Human Values Dashboard for Users

The exploration stage results suggested that one of a human values dashboard's main benefits is promoting the awareness of values. This awareness of values could trigger discussions among stakeholders on what values must be considered in an application. Then, as suggested by the findings in the feedback stage, the development team could focus on the prioritised values and ensure these values are addressed during development. This study focused on software practitioners involved in software development. It did not include end users of an application as one of the stakeholders in software development. Application users are indirectly involved in application development by providing feedback. Giving them access to a human values dashboard would help users evaluate the values of an application [184], which in turn could guide them to choose their preferred application [5, 6, 185]. However, to understand the dashboard's usefulness for users, a future study involving users needs to be carried out.

5.6.5 Limitations and Challenges of the Dashboard

The human values dashboard (Figure 5.5) may have several limitations. First, the dashboard depends on the availability of artefacts (e.g. issue discussions). A project may not have all of the artefacts in Table 5.3 depending on how it is managed. Second, an automated approach has been chosen to identify the presence of values because it can reduce manual efforts. Identification using an automated approach has accuracy limitations. Although the tolerance level for inaccuracies varied among the participants, here the inaccuracies were understandable by the participants. The findings also revealed that all practitioners preferred false positives to false negatives. In comparison with this finding, the evaluation of the classification methods in Section 4.3.4 show higher recalls (i.e. lower false negatives) and lower precisions (i.e. higher false positives). This means that many practitioners could consider the classification methods evaluated in Section 4.3.4 that provide a recall score of 0.75 to be 'good enough' to indicate whether human values are present in an issue. Furthermore, evaluation metrics that emphasise false positives, such as the F2 score [117], could be used to evaluate the performance of the automated human values detector. Also, in the feedback stage, some practitioners (Section 5.5.4.3) suggested displaying the confidence level of the human values detection. This information could help practitioners prioritise issues with a higher level of confidence. Third, the automated approach used in the dashboard at this point is only able to detect whether any human values are present, without specifying which values. This approach could be improved in the future using several approaches discussed in Section 4.4.2. Another direction to improve the automated detection tool could be asking practitioners, as users of the dashboard, to add or correct the labels on the artefacts. These additions and corrections could then be incorporated as feedback to retrain the classification model to improve the identification over time.

Despite these limitations, the findings showed that the dashboard would be beneficial for software development. However, some practitioners highlighted two potential challenges in adopting the dashboard. The first challenge was an unclear or incorrect description of an issue provided by the reporter. One way to address this challenge is to provide issue reporting guidelines. Additionally, practitioners could ask for clarification in a post on that particular issue. The second challenge was related to the willingness of a company to adopt the dashboard. To address this challenge, essential efforts should be made to increase the awareness of human values. Providing regulations and standards (e.g. GDPR [186]) is one potential way to increase this awareness.

5.7 Threats to Validity

This section discusses the potential threats arising from the research method and the findings. This section uses the following validation criteria, which are considered suitable for qualitative research [187–189]:

Credibility: Possible threats to the credibility of this study could arise from the procedures used to collect data, develop interview questions, or select participants. Although the data collected came from only one source (interviews), the initial step of examining the literature before developing the dashboard prototype could increase the plausibility of our findings. To mitigate the threats from the interview questions, open-ended questions were used, and follow-up questions tailored to each participant's responses were asked. The use of issue discussions in the prototype of the exploration stage's interviews may have introduced bias into the participants' responses. This threat was mitigated by probing the participants to consider the possibilities of using other artefacts as the dashboard's source. To reduce possible threats stemming from the selection of participants, this study relied on the criteria for the recruitment of participants. The list of participants consisted of software practitioners with diverse roles, experiences, and work locations. Thus, the participants had the right competencies to provide insight for the study. To mitigate the uneven number of participants in each role, the interviews also asked them to share their opinions from other roles' perspectives. This approach allowed cross-validation of findings across different roles. The recruitment of participants by inviting practitioners from GitHub via email may introduce another threat to this study. Although the invitation was sent to practitioners using the email that were publicly available on GitHub, this approach may raise minor issues, e.g. a few of them complained about their email addresses being harvested.

Confirmability: A possible threat to the confirmability of this study could have been introduced by the definitions of human values, which have not been specifically developed for software engineering. To mitigate this, some examples were provided to the participants to describe what a value could possibly mean in software engineering contexts (e.g. '*A user who values privacy may not choose an application with a bad privacy reputation*'). Participants were also allowed to reflect and translate values into contextualised software engineering definitions based on their experiences. The data analysis could have introduced another possible threat to the confirmability, as it was primarily conducted by the present author. This threat was mitigated by having other authors review and validate the codes/themes in several discussions.

Transferability: This study accepted that the findings cannot be generalised to all software organisations and practitioners. Different results might have been discovered if another group of participants were included. However, this threat was reduced by involving a reasonable number of participants with various development roles and work locations. Additionally, the data reached saturation during the parallel work of interviews and data analysis. This study also accepted that the relative importance of some specific values to others cannot be generalised because the whole list of values was not presented to the participants. This threat was mitigated by concluding that some values are more important than others.

5.8 Concluding Remarks

This study envisioned a values-driven dashboard and investigated whether it would help software practitioners to address values during software development. The exploration stage of this study found that, in general, the practitioners acknowledged that a human values dashboard would be beneficial for them. The dashboard could raise awareness of values among development teams and inform values-based decision-making in project management. Supporting the idea of using artefacts as the dashboard source, the practitioners suggested requirements documents and issue discussions as the most suitable artefacts for values identification in the dashboard. This study also received suggestions as a set of requirements to develop the envisioned dashboard.

Based on these requirements, a human values dashboard was developed as a proof of concept. Then, feedback interviews were performed to obtain the practitioners' opinions on the dashboard. This study found that the human values dashboard could help focus attention and prioritise issues, in line with the findings from the exploration stage. The practitioners also suggested several potential challenges, such as the willingness and extra efforts required to deploying the dashboard in a company. Regarding the performance of the human values detector, the practitioners had different levels of tolerance, but all agreed that false positives were preferable to false negatives. The practitioners also made 16 suggestions to improve the dashboard.

The practitioners' suggestions and the results of this study could further improve the human values dashboard. Future work could focus on investigating other software artefacts to be detected in the dashboard. Another direction could be implementing the dashboard in a company and investigating its acceptance in a company setting.

Chapter 6

Conclusions and Future Work

This thesis presents a set of studies to support the integration of human values during software development. Specifically, this thesis aims to evaluate the presence of human values in issue discussions and provide a tool to make the presence of values useful in practice. For that purpose, first, an empirical study was conducted to investigate whether human values are present in issue discussions. Second, machine learning approaches were evaluated with various parameters to automate the detection of human values in issue discussions. Finally, a human values dashboard was developed, and qualitative studies with practitioners were performed to understand the dashboard's

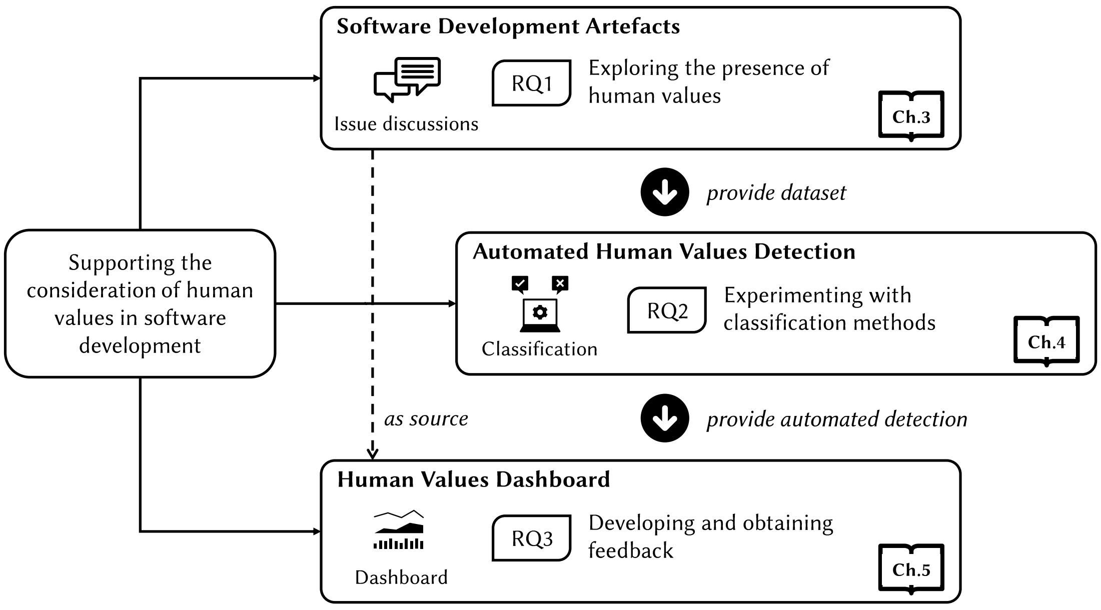

Software Development Artefacts
Issue discussions
RQ1
Exploring the presence of human values
Ch.3
provide dataset
Supporting the consideration of human values in software development
Automated Human Values Detection
Classification
RQ2
Experimenting with classification methods
Ch.4
as source
provide automated detection
Human Values Dashboard
Dashboard
RQ3
Developing and obtaining feedback
Ch.5

FIGURE 6.1: Overview of the research phases and their corresponding chapters

usefulness. Figure 6.1 shows an overview of the phases in the research and their corresponding chapters.

The remainder of this chapter is organised as follows: Section 6.1 presents the answers provided by this thesis for each research question. Section 6.2 discusses potential implications of the research presented in this thesis. Section 6.3 provides directions for potential future work based on this thesis.

6.1 Answers to the Research Questions

This thesis addresses the research questions proposed to guide the investigation of human values in issue discussions and determine how a dashboard that utilises these insights could support the consideration of human values in software engineering. The answers to each research question are presented below.

RQ1 To what extent are human values present in issue discussions?

- This research developed definitions to determine whether human values are present in issue discussions, which could support the analysis of human values in issue discussions.
- This research discovered 20 value themes that could be divided into two categories: human value themes and system value themes. For the human value themes, this research found that 10 themes could be mapped to 7 out of 19 of Schwartz's values. The remaining 10 themes were system value themes because they were important but could not easily be mapped to Schwartz's values. These value themes could provide human values perspectives on a software development project.
- This research proposed contextualised software engineering descriptions for each of the value themes that were found based on the empirical evidence from the issue discussions. These descriptions could better explain the abstract concepts of human values to software practitioners.
- This research provided empirical evidence that human values were present in one-third of the analysed issues. This research also identified a statement of supporting human values and the functionality of an application as two factors that influence the presence of values.

RQ2 To what extent can the detection of human values be automated?

- This research observed that the classification using the issue level classification unit performed better than the post level classification unit. The much smaller number of cases of posts compared to issues wherein human values were present was suspected as the cause for these results.
- This research found that using the multi-layer perceptron method with undersampling and using the TF-IDF feature performed the best for identifying human values.
- This research discovered that the effects of using resampling techniques varied with each classification method and feature set. However, the best performer in each classification method was found to use a resampling technique.
- This research found that the TF-IDF feature performed better than the BoW feature for the majority of the cases. Using the sentiment feature from SentiStrength in the classification had varying effects on the classification performance.

RQ3 How can the automated detection of human values be useful in software development?

- This research revealed that software practitioners agree that human values are important. However, the current understanding of values in software engineering is limited to well-known values.
- This research highlighted for different roles and in various phases of software development a human values could be useful. These findings could be used to introduce the human values dashboard in industry settings.
- This research identified two software development artefacts, namely, requirements documents and issue discussions, as the most suitable sources for detecting human values. These findings could be incorporated into the development of the dashboard.
- This research elicited 6 high-level requirements suggested by software practitioners to a human values dashboard.
- This research developed a human values dashboard as a proof of concept based on the requirements suggested by the software practitioners.

- This research revealed that the dashboard could be useful, especially to focus attention on and prioritise the issues during software development. Potential challenges in deploying the dashboard in industry settings were discovered, such as the willingness and extra efforts by a company when deploying the dashboard.
- This research revealed that software practitioners vary in their level of tolerance regarding the performance of the detection of human values. The practitioners preferred to have false positives over false negatives.
- This research listed 16 suggestions to improve the human values dashboard, which could help software practitioners integrate human values into software development.

6.2 Implications

From an empirical point of view, this research aimed to assess whether human values are present in GitHub issue discussions. Developing a human values dashboard to highlight these values could promote the awareness and consideration of values during software development. This research has several implications for industry practice and researchers, which are described in the following subsections.

6.2.1 Implications for Industry Practice

This research provides empirical evidence that human values are present in issue discussions. The dashboard presented in this research could be used as a tool to support the consideration of human values during software development. Because the human values dashboard uses GitHub Issues, the dashboard could be used during the development, maintenance, and release phases of software development. This means any software project that has issues in their repository could utilise the dashboard even if their software or application has been released.

The use of GitHub in the human values dashboard allows for immediate deployment and utilisation of the dashboard for software projects hosted in GitHub. The dashboard enables project stakeholders to assess values at any point in the software development. Moreover, the cross-referencing between issues and other artefacts (e.g. pull requests and source code commits) of GitHub [95] could also indirectly indicate the presence of values in those artefacts. Including other artefacts available through GitHub in the dashboard would allow project stakeholders to have a complementary view of values to the technical aspects of application development.

A human values dashboard can promote the awareness of human values within a software development team. This will encourage the team members to discuss values and their considerations during software development. Additionally, if the discussion's participants are recorded, the organisation could identify **values champions** (i.e. *'team members who actively advocate for values inclusion'* [190]).

An organisation may have more than one project hosted on other repositories, such as GitLab or BitBucket. These projects could also benefit from the human values dashboard. These collaborative repositories are known to have similar functionalities to GitHub. Thus, implementing a human values dashboard for these repositories is possible with some modifications in the developed dashboard. The dashboard is designed to accommodate numerous projects (Section 5.4.1). An organisation could specify its projects to be included in the dashboard. This feature allows for comparison of projects and could potentially benefit higher-level management, who oversee many software projects within an organisation.

Organisations or project owners need to adhere to the following practices to reap the benefits of a human values dashboard. First, they should allow and encourage discussion within a project's repository. Although not compulsory, involving end users of a project in issue discussions will complement the human values dashboard with the users' perspectives of the application. Second, they should use project repositories to store artefacts generated during software development. These artefacts include requirements specifications, design documents, and the like. Having all these artefacts assessed for values and displayed in the dashboard will provide a holistic view of values in the software project.

From an end user point of view, a human values dashboard allows users to assess how a software project addresses values-related issues. Furthermore, the dashboard can be used as a base for human values auditing and certification. This could enable users to make informed decisions on the application that they want to use.

6.2.2 Implications for Researchers

The findings of this research could enable further research to obtain a deeper understanding of human values in software engineering. For instance, the value themes descriptions (Section 3.3.1) could be used as a base to perform qualitative and quantitative analyses to develop and refine human values definitions in software engineering contexts. These descriptions could also facilitate bridging the gap in understanding of software practitioners by involving them in evaluating and improving the human values descriptions for software engineering.

The automated human values detection in this research could facilitate further research of automated detection for other software development artefacts. The criteria developed to identify the presence of values could enable researchers to build a larger dataset to accommodate each value in Schwartz's model. This larger dataset would allow further performance improvement of the human values detection.

The feedback from software practitioners regarding the human values dashboard could enable further studies to improve the dashboard. The human values dashboard itself could allow for a study involving software practitioners to evaluate the dashboard in a real development environment. Such a follow-up study could provide insights into how useful the dashboard is in actual software development practice.

6.3 Potential Future Work

The research in this thesis presents a step towards supporting the integration of human values in software development. Although this thesis offers significant contributions, some areas and challenges require further work. The following subsections present potential future work to address these challenges.

6.3.1 Extending the Empirical Study of Values in Software Artefacts

The findings of this research (Section 3.3) suggest an opportunity for researchers to investigate software development artefacts other than issue discussions. Pull requests and source code commits on GitHub are two artefacts with a similar discussion feature.

GitHub Issues and these two artefacts are commonly used in software maintenance and evolution research (e.g. [191, 192]). Thus, these two artefacts would provide a dataset that allows human values research from software maintenance and evolution perspectives. Furthermore, investigating other software development artefacts, such as requirements, design, or code of conduct documents, could potentially lead to a more comprehensive study of human values throughout the software development life cycle.

The findings of this research (Figure 3.7) also show that the presence of specific values, such as dignity or wealth, in the three applications is very low. Further work could examine other categories of applications on a larger scale, with many projects to extend the dataset for particular values. In addition, examining a specific category of applications, such as computer games or educational applications, might lead to the discovery of new value themes. This direction could also contribute to the incremental refinement and development of human values definitions in software engineering contexts.

The analysis of software development artefacts and other application categories could be performed using the thematic analysis method. Future research in this direction could use and refine the criteria presented in Section 3.2.2.2 to better capture the presence of human values in natural language-based software development artefacts. To support the analysis of human values on a larger scale, future studies could develop and use keywords related to human values to shortlist artefacts before the following manual analysis.

6.3.2 Improving and Expanding the Detection of Human Values

The performance of detecting human values could be improved by expanding the dataset and experimenting with other detection methods. Future studies could use the cost-effective approaches proposed in recent studies [63, 167] to reduce efforts to manually expand the dataset. Alternatively, another study could examine the use or refine other available datasets for specific values, such as security (e.g. [70, 72]) or diversity (e.g. [193, 194]), to improve classification performance. For detection methods, future studies could evaluate other approaches, such as deep learning or keyword- and knowledge-based methods, or newer classification features, such as word embedding, to determine whether they could improve the classification.

Another possible line of research involves the development of classifiers for specific values, such as face or hedonism. This direction could be achieved by implementing multilabel classification, as demonstrated in previous studies on text documents (e.g. [61, 64]). Alternatively, each classifier could be developed for each specific value. An interview study could be conducted involving software professionals to help prioritise the specific values classifier to be developed. This research direction may require an extension to the dataset to detect particular values. Therefore, this research direction could be a continuation of studies that expand the investigation of human values in software development artefacts (Section 6.3.1).

6.3.3 Evaluating the Human Values Dashboard in Practise

This thesis presents the evaluation of the human values dashboard through interviews with software practitioners. This evaluation results in feedback to improve the dashboard. Future research could use this feedback before evaluating the dashboard in industry settings. An evaluation study could compare the development process before and after deploying the dashboard. For example, whether it is useful for software development and how the development team interacts and uses the dashboard. This evaluation could be performed using an observational study or a controlled experiment. In an observational study, the human values dashboard could be injected into the software development of a project. The study then observes how the team uses and interacts with the dashboard. Meanwhile, a controlled experiment may involve providing the dashboard to some members of the development team, while others remain as usual. These studies can then be followed up by interviews with members of the development team. For example, whether the dashboard is useful or whether the dashboard is ready for practical use in the industry. Furthermore, this direction could investigate the possible challenges of adopting the dashboard and the possible uses of the dashboard in a company. For example, how software development teams could include the use of the dashboard in their development activities, what the challenges are, and what other possible uses of the dashboard are for management.

6.3.4 Investigating the Relationship between Human Values in Issues and the Software Development Process

Analysis of issue discussions shows that human values are discussed in a software project. A potential future direction of this study is to investigate whether the presence of human values affects how the software is developed. For example, an analysis could examine whether the development team solved issues with values faster than those that were not present. Another empirical study could analyse whether values-related issues attract more attention, resulting in a higher level of activities (e.g. comments, pull requests, or commits) related to those issues. This analysis could be followed by an interview study to explore how the software development team addressed these issues. For example, how they make decisions regarding these issues and whether they handle them differently from any other issue. This research direction could potentially provide the level of awareness and importance given by the development team toward human values during software development.

This study also shows that having human values in the discussion of issues can mean that values are present or absent in the software. Future studies could examine whether values-related issues result in code changes (e.g. commits or pull requests) and software releases. Furthermore, further studies could reveal which features of the software corresponded to human values. This study could be followed by an interview with software practitioners and users to validate the findings and develop a list of mappings between values and features.

6.3.5 Investigating the Relationship between Human Values and the Development Team Dynamics and Composition

This thesis presents human values perspectives in software development. Meanwhile, previous work has investigated other social aspects of software development. For example, recent studies have investigated the impacts of team dynamics (e.g. politeness [195, 196]) and composition (e.g. diversity [197] or gender [194]) on the development process and team productivity. Therefore, further studies could investigate the relationship between human values and these social aspects of software engineering. Quantitative analysis of software repository artefacts could examine if there is co-occurrence between the presence of values and the team dynamics and composition. For example, whether the diversity of the team in a project co-occurs with presence of inclusiveness value in the issue discussions. The following qualitative analysis can investigate whether there is a connection between them. For example, a team of different nationalities may initiate conversations about cultural considerations during software development. The results of both analyses could be complemented with an observational and interview study involving members of the development team.

Appendix A

Automated Human Values Detection Results

Table A.1: Best performers of the evaluation experiments using issue level dataset without any sampling techniques

No.	Resampling tech.	Feature set	Method	Precision	Recall	F1	MCC
1	No Sampling	BoW	SVM	0.607±0.078	0.365±0.067	0.453±0.068	0.288±0.084
2	No Sampling	BoW	RF	0.653±0.083	0.477±0.065	0.548±0.059	0.381±0.084
3	No Sampling	BoW	MLP	0.635±0.046	0.597±0.076	0.612±0.048	0.431±0.063
4	No Sampling	BoW	LR	0.445±0.040	0.885±0.077	0.589±0.017	0.327±0.045
5	No Sampling	TF-IDF	SVM	0.573±0.038	0.646±0.069	0.606±0.044	0.393±0.066
6	No Sampling	TF-IDF	RF	0.684±0.068	0.499±0.075	0.573±0.063	0.419±0.078
7	No Sampling	TF-IDF	MLP	0.647±0.037	0.581±0.071	0.610±0.043	0.433±0.054
8	No Sampling	TF-IDF	LR	0.603±0.056	0.638±0.071	0.616±0.042	0.418±0.066
9	No Sampling	BoW+Sentiment	SVM	0.626±0.082	0.406±0.069	0.490±0.070	0.324±0.093
10	No Sampling	BoW+Sentiment	RF	0.649±0.087	0.517±0.067	0.572±0.057	0.397±0.092
11	No Sampling	BoW+Sentiment	MLP	0.633±0.056	0.583±0.091	0.605±0.065	0.423±0.087
12	No Sampling	BoW+Sentiment	LR	0.485±0.054	0.803±0.098	0.597±0.018	0.348±0.037
13	No Sampling	TF-IDF+Sentiment	SVM	0.514±0.036	0.578±0.064	0.543±0.040	0.295±0.056
14	No Sampling	TF-IDF+Sentiment	RF	0.658±0.095	0.531±0.085	0.585±0.081	0.415±0.113
15	No Sampling	TF-IDF+Sentiment	MLP	0.665±0.056	0.578±0.063	0.615±0.041	0.447±0.057
16	No Sampling	TF-IDF+Sentiment	LR	0.603±0.059	0.681±0.069	0.637±0.050	0.442±0.074

Table A.2: Best performers of the evaluation experiments using issue level dataset and an oversampling technique

No.	Resampling tech.	Feature set	Method	Precision	Recall	F1	MCC
17	Oversampling	BoW	SVM	0.556±0.078	0.554±0.084	0.546±0.042	0.322±0.071
18	Oversampling	BoW	RF	0.620±0.052	0.618±0.095	0.615±0.067	0.428±0.079
19	Oversampling	BoW	MLP	0.591±0.068	0.599±0.040	0.593±0.042	0.385±0.074
20	Oversampling	BoW	LR	0.477±0.029	0.738±0.100	0.575±0.037	0.312±0.052
21	Oversampling	TF-IDF	SVM	0.606±0.071	0.580±0.082	0.590±0.063	0.393±0.091
22	Oversampling	TF-IDF	RF	0.604±0.073	0.395±0.088	0.473±0.080	0.302±0.090
23	Oversampling	TF-IDF	MLP	0.649±0.043	0.586±0.077	0.613±0.051	0.438±0.063
24	Oversampling	TF-IDF	LR	0.622±0.054	0.602±0.071	0.608±0.046	0.419±0.064
25	Oversampling	BoW+Sentiment	SVM	0.552±0.083	0.545±0.085	0.539±0.045	0.313±0.079
26	Oversampling	BoW+Sentiment	RF	0.637±0.063	0.610±0.085	0.619±0.058	0.438±0.078
27	Oversampling	BoW+Sentiment	MLP	0.579±0.052	0.597±0.034	0.586±0.033	0.374±0.059
28	Oversampling	BoW+Sentiment	LR	0.492±0.039	0.705±0.073	0.577±0.038	0.318±0.063
29	Oversampling	TF-IDF+Sentiment	SVM	0.562±0.054	0.578±0.085	0.566±0.059	0.348±0.079
30	Oversampling	TF-IDF+Sentiment	RF	0.631±0.067	0.422±0.084	0.503±0.080	0.338±0.088
31	Oversampling	TF-IDF+Sentiment	MLP	0.654±0.053	0.597±0.072	0.623±0.056	0.449±0.076
32	Oversampling	TF-IDF+Sentiment	LR	0.613±0.045	0.643±0.077	0.625±0.049	0.434±0.067

Table A.3: Best performers of the evaluation experiments using issue level dataset and an undersampling technique

No.	Resampling tech.	Feature set	Method	Precision	Recall	F1	MCC
33	Undersampling	BoW	SVM	0.597±0.085	0.414±0.067	0.487±0.068	0.305±0.091
34	Undersampling	BoW	RF	0.581±0.049	0.602±0.064	0.589±0.041	0.378±0.063
35	Undersampling	BoW	MLP	0.558±0.038	0.739±0.084	0.633±0.045	0.422±0.073
36	Undersampling	BoW	LR	0.584±0.062	0.643±0.061	0.608±0.042	0.399±0.072
37	Undersampling	TF-IDF	SVM	0.575±0.051	0.676±0.059	0.619±0.038	0.407±0.064
38	Undersampling	TF-IDF	RF	0.584±0.037	0.640±0.076	0.609±0.048	0.402±0.065
39	Undersampling	TF-IDF	MLP	0.582±0.043	0.741±0.062	0.650±0.032	0.451±0.053
40	Undersampling	TF-IDF	LR	0.575±0.044	0.709±0.062	0.632±0.032	0.424±0.054
41	Undersampling	BoW+Sentiment	SVM	0.605±0.087	0.422±0.064	0.495±0.067	0.315±0.092
42	Undersampling	BoW+Sentiment	RF	0.574±0.050	0.615±0.073	0.592±0.052	0.379±0.074
43	Undersampling	BoW+Sentiment	MLP	0.557±0.041	0.711±0.086	0.622±0.039	0.407±0.061
44	Undersampling	BoW+Sentiment	LR	0.584±0.062	0.654±0.063	0.615±0.048	0.406±0.081
45	Undersampling	TF-IDF+Sentiment	SVM	0.509±0.036	0.591±0.059	0.546±0.038	0.295±0.056
46	Undersampling	TF-IDF+Sentiment	RF	0.574±0.046	0.640±0.082	0.603±0.055	0.392±0.078
47	Undersampling	TF-IDF+Sentiment	MLP	0.575±0.055	0.738±0.062	0.644±0.045	0.439±0.074
48	Undersampling	TF-IDF+Sentiment	LR	0.570±0.045	0.757±0.047	0.649±0.035	0.445±0.059

Table A.4: Best performers of the evaluation experiments using post level dataset without any sampling techniques

No.	Resampling tech.	Feature set	Method	Precision	Recall	F1	MCC
49	No Sampling	BoW	SVM	0.390±0.042	0.559±0.102	0.458±0.062	0.401±0.072
50	No Sampling	BoW	RF	0.408±0.031	0.490±0.050	0.445±0.038	0.384±0.042
51	No Sampling	BoW	MLP	0.455±0.073	0.350±0.081	0.394±0.075	0.345±0.079
52	No Sampling	BoW	LR	0.406±0.074	0.430±0.085	0.416±0.075	0.355±0.082
53	No Sampling	TF-IDF	SVM	0.284±0.023	0.637±0.061	0.393±0.032	0.337±0.043
54	No Sampling	TF-IDF	RF	0.423±0.025	0.490±0.054	0.454±0.035	0.395±0.038
55	No Sampling	TF-IDF	MLP	0.462±0.069	0.341±0.068	0.391±0.067	0.344±0.070
56	No Sampling	TF-IDF	LR	0.345±0.030	0.623±0.076	0.444±0.043	0.389±0.053
57	No Sampling	BoW+Sentiment	SVM	0.381±0.035	0.580±0.097	0.458±0.055	0.402±0.065
58	No Sampling	BoW+Sentiment	RF	0.385±0.029	0.527±0.065	0.444±0.042	0.383±0.047
59	No Sampling	BoW+Sentiment	MLP	0.476±0.086	0.326±0.062	0.387±0.070	0.344±0.077
60	No Sampling	BoW+Sentiment	LR	0.416±0.069	0.443±0.087	0.427±0.075	0.368±0.081
61	No Sampling	TF-IDF+Sentiment	SVM	0.346±0.025	0.675±0.053	0.457±0.033	0.410±0.041
62	No Sampling	TF-IDF+Sentiment	RF	0.397±0.024	0.525±0.039	0.452±0.027	0.392±0.031
63	No Sampling	TF-IDF+Sentiment	MLP	0.472±0.065	0.363±0.070	0.408±0.061	0.361±0.064
64	No Sampling	TF-IDF+Sentiment	LR	0.359±0.036	0.653±0.063	0.463±0.045	0.413±0.054

Table A.5: Best performers of the evaluation experiments using post level dataset and an oversampling technique

No.	Resampling tech.	Feature set	Method	Precision	Recall	F1	MCC
65	Oversampling	BoW	SVM	0.104±0.020	0.473±0.070	0.171±0.030	0.030±0.052
66	Oversampling	BoW	RF	0.087±0.010	0.498±0.063	0.148±0.015	-0.027±0.038
67	Oversampling	BoW	MLP	0.244±0.043	0.330±0.059	0.280±0.049	0.196±0.055
68	Oversampling	BoW	LR	0.202±0.034	0.303±0.051	0.241±0.039	0.149±0.045
69	Oversampling	TF-IDF	SVM	0.313±0.034	0.503±0.083	0.385±0.048	0.317±0.056
70	Oversampling	TF-IDF	RF	0.393±0.027	0.376±0.030	0.384±0.028	0.322±0.030
71	Oversampling	TF-IDF	MLP	0.301±0.027	0.648±0.069	0.410±0.038	0.357±0.049
72	Oversampling	TF-IDF	LR	0.369±0.051	0.573±0.089	0.448±0.062	0.390±0.072
73	Oversampling	BoW+Sentiment	SVM	0.498±0.230	0.151±0.101	0.187±0.066	0.194±0.073
74	Oversampling	BoW+Sentiment	RF	0.081±0.013	0.446±0.084	0.136±0.022	-0.045±0.043
75	Oversampling	BoW+Sentiment	MLP	0.229±0.038	0.334±0.056	0.272±0.043	0.185±0.050
76	Oversampling	BoW+Sentiment	LR	0.155±0.024	0.371±0.069	0.218±0.036	0.113±0.046
77	Oversampling	TF-IDF+Sentiment	SVM	0.472±0.041	0.441±0.065	0.454±0.045	0.402±0.047
78	Oversampling	TF-IDF+Sentiment	RF	0.386±0.046	0.382±0.049	0.384±0.046	0.321±0.050
79	Oversampling	TF-IDF+Sentiment	MLP	0.304±0.023	0.672±0.065	0.418±0.032	0.369±0.042
80	Oversampling	TF-IDF+Sentiment	LR	0.385±0.040	0.597±0.074	0.467±0.050	0.412±0.058

Table A.6: Best performers of the evaluation experiments using post level dataset and an undersampling technique

No.	Resampling tech.	Feature set	Method	Precision	Recall	F1	MCC
81	Undersampling	BoW	SVM	0.332±0.039	0.517±0.089	0.397±0.034	0.334±0.036
82	Undersampling	BoW	RF	0.295±0.023	0.659±0.052	0.407±0.030	0.356±0.039
83	Undersampling	BoW	MLP	0.296±0.027	0.667±0.052	0.410±0.032	0.359±0.039
84	Undersampling	BoW	LR	0.280±0.019	0.694±0.056	0.399±0.026	0.352±0.035
85	Undersampling	TF-IDF	SVM	0.234±0.012	0.758±0.048	0.357±0.018	0.318±0.027
86	Undersampling	TF-IDF	RF	0.294±0.023	0.693±0.058	0.412±0.032	0.366±0.041
87	Undersampling	TF-IDF	MLP	0.224±0.012	0.753±0.039	0.345±0.015	0.303±0.022
88	Undersampling	TF-IDF	LR	0.238±0.012	0.764±0.041	0.362±0.017	0.324±0.024
89	Undersampling	BoW+Sentiment	SVM	0.316±0.025	0.621±0.084	0.416±0.027	0.361±0.035
90	Undersampling	BoW+Sentiment	RF	0.294±0.023	0.659±0.063	0.407±0.032	0.355±0.042
91	Undersampling	BoW+Sentiment	MLP	0.297±0.025	0.713±0.066	0.419±0.033	0.376±0.042
92	Undersampling	BoW+Sentiment	LR	0.286±0.016	0.718±0.040	0.409±0.021	0.366±0.027
93	Undersampling	TF-IDF+Sentiment	SVM	0.322±0.035	0.521±0.047	0.398±0.038	0.331±0.044
94	Undersampling	TF-IDF+Sentiment	RF	0.291±0.025	0.678±0.059	0.407±0.033	0.358±0.044
95	Undersampling	TF-IDF+Sentiment	MLP	0.242±0.018	0.739±0.051	0.364±0.025	0.321±0.034
96	Undersampling	TF-IDF+Sentiment	LR	0.245±0.020	0.753±0.050	0.369±0.028	0.330±0.038

Appendix B

Interview Questions for the Dashboard's Exploratory Study

Topics for Interviews – Human Values Dashboard

Software Practitioners

Estimated time: 40 - 60 minutes

Introductions: Introduce myself, my research area and the topic of 'human values'.

Part 1 – Demographic Information

Questions:

1. Are you involved in a software project? Which project are you working on (open source/industrial/proprietary)?
2. What is your role in that project?
3. How long have you been involved in that project? How long is your experience in software development?
4. Are you familiar with software development artefacts? Are you familiar with GitHub or other software repositories?

Part 2 - Human Values Perceptions and Experiences

The next scenarios are some examples of how values may play a role in the development or selection of software.

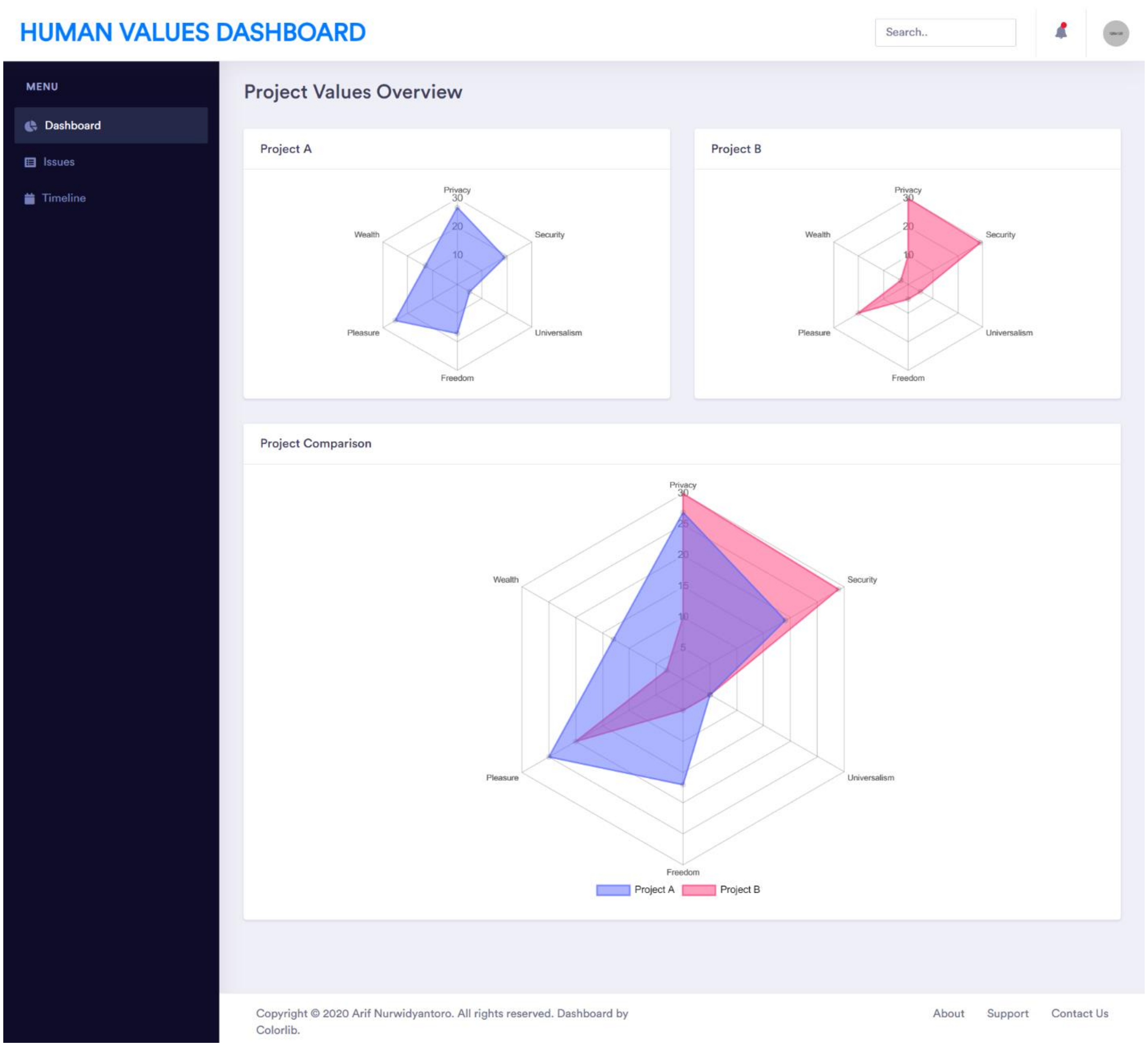
Triggering Case Scenario

1. App-selection/User perspective:
   A user who does not want to install or use Facebook because of its poor reputation (e.g., Cambridge-Analytica case) in protecting the privacy of its users. (Privacy)
   a. A user wants to **check** the way an app addresses privacy before installing an app, or for choosing between apps.
2. Developer perspective:
   a. A software designer designs a language selection feature for the user interface in a system used by multinational users. (Universalism)
   b. A software developer carefully chooses background and font colours for a mobile application to support colour-blind users. (Accessibility)

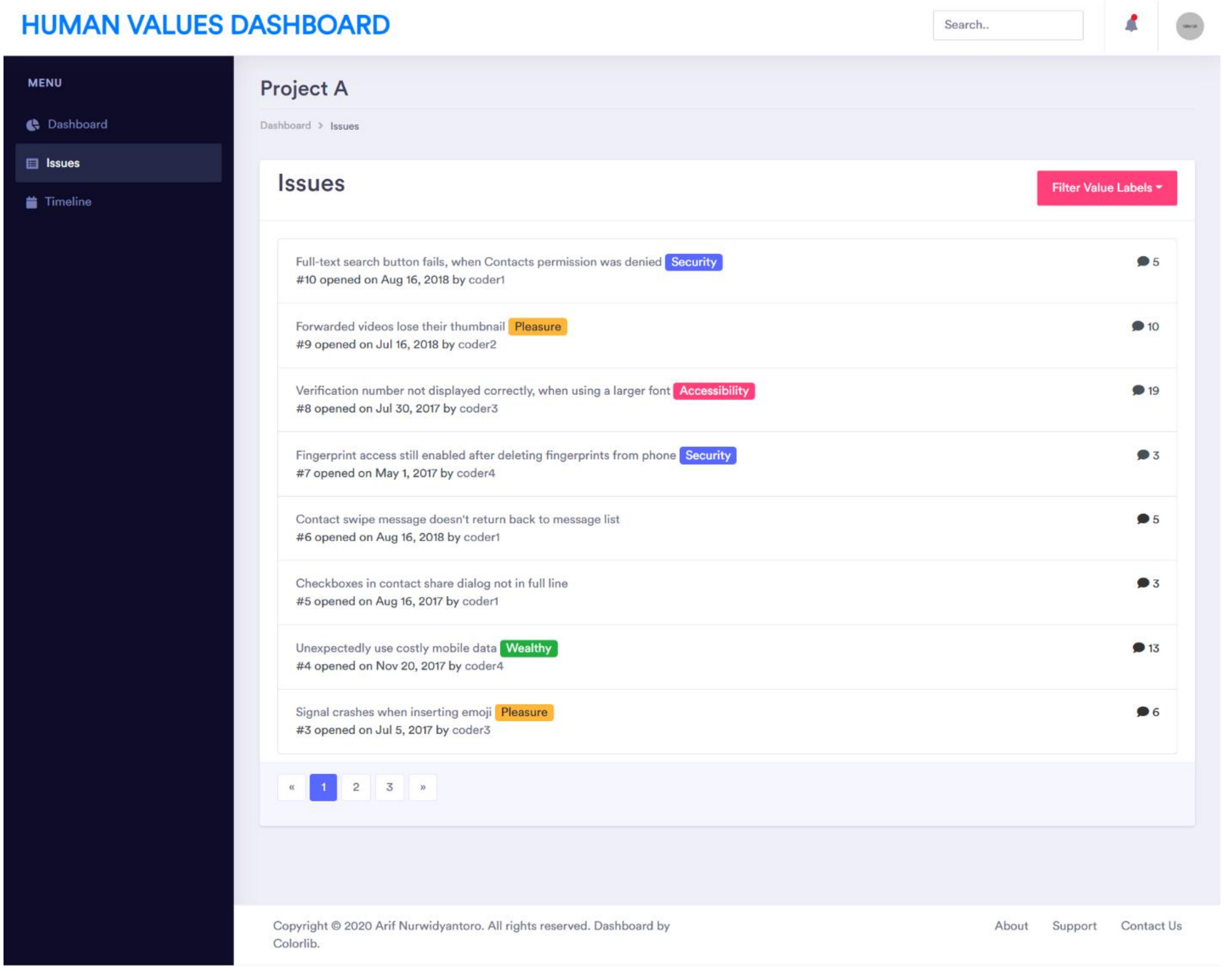
Questions:

1. Do you have any similar experience during developing software?
   a. Do you consider any ***<u>particular</u>*** value(s) important?
   b. How do you consider/deal with ***<u>that particular</u>*** value(s) when you develop/use software?
   c. How does your company view values in software development?
2. Human values are something that people consider important and guide their decision. From a model here (show the participant a human value model), which ones raise your interest?

Part 3 –Human Values Dashboard for Software Development



Summarised Values Overview


HUMAN VALUES DASHBOARD

Search..

MENU

Dashboard

Issues

Timeline

Project A

Dashboard > Issues

Issues

Filter Value Labels

Full-text search button fails, when Contacts permission was denied Security 5
#10 opened on Aug 16, 2018 by coder1

Forwarded videos lose their thumbnail Pleasure 10
#9 opened on Jul 16, 2018 by coder2

Verification number not displayed correctly, when using a larger font Accessibility 19
#8 opened on Jul 30, 2017 by coder3

Fingerprint access still enabled after deleting fingerprints from phone Security 3
#7 opened on May 1, 2017 by coder4

Contact swipe message doesn't return back to message list 5
#6 opened on Aug 16, 2018 by coder1

Checkboxes in contact share dialog not in full line 3
#5 opened on Aug 16, 2017 by coder1

Unexpectedly use costly mobile data Wealthy 13
#4 opened on Nov 20, 2017 by coder4

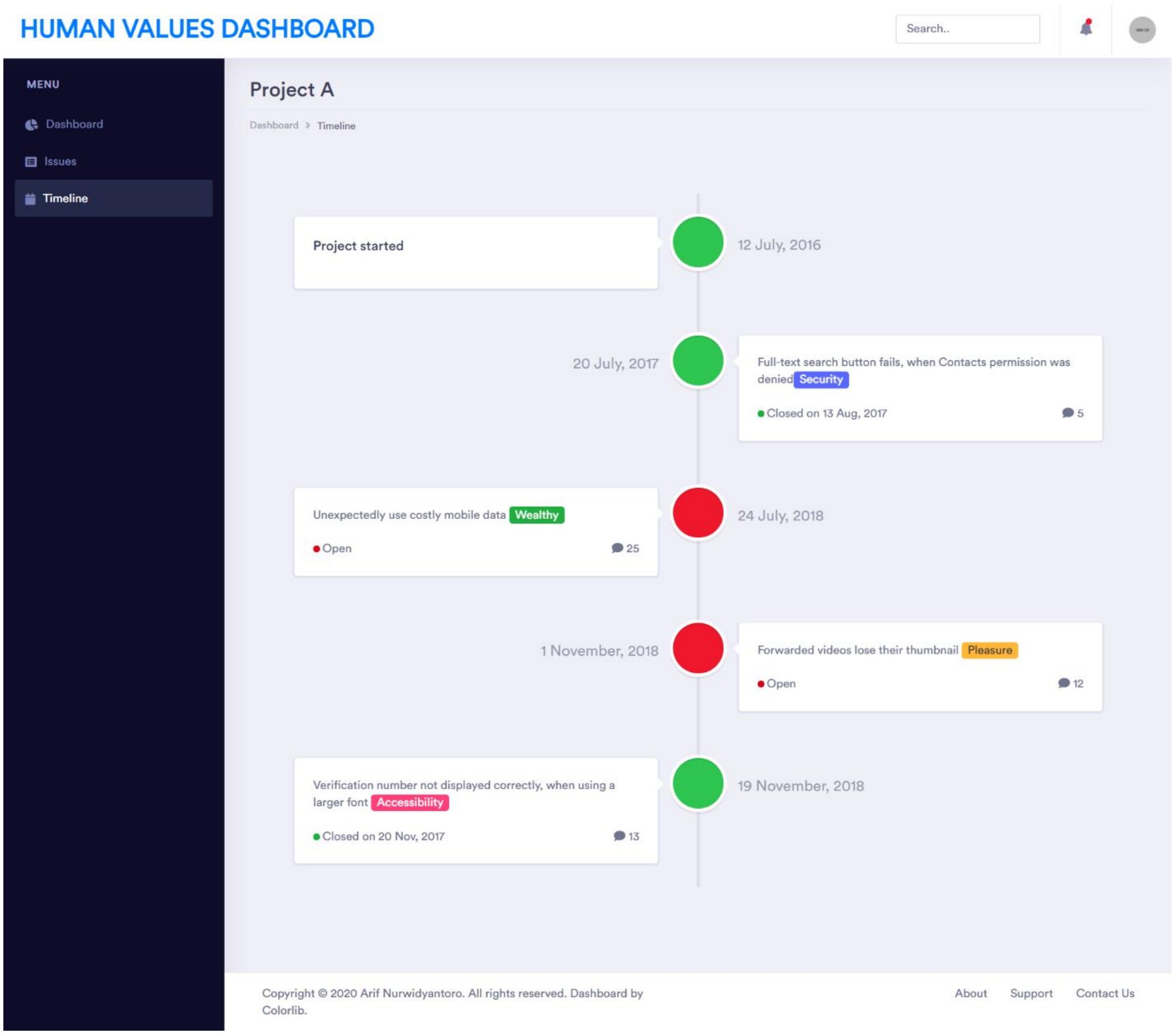
Signal crashes when inserting emoji Pleasure 6
#3 opened on Jul 5, 2017 by coder3

« 1 2 3 »

Copyright © 2020 Arif Nurwidyantoro. All rights reserved. Dashboard by Colorlib.

About Support Contact Us

Values-labelled List


HUMAN VALUES DASHBOARD

Search..

MENU

Dashboard

Issues

Timeline

Project A

Dashboard > Timeline

Project started — 12 July, 2016

20 July, 2017 — Full-text search button fails, when Contacts permission was denied Security
● Closed on 13 Aug, 2017 — 5

24 July, 2018 — Unexpectedly use costly mobile data Wealthy
● Open — 25

1 November, 2018 — Forwarded videos lose their thumbnail Pleasure
● Open — 12

19 November, 2018 — Verification number not displayed correctly, when using a larger font Accessibility
● Closed on 20 Nov, 2017 — 13

Copyright © 2020 Arif Nurwidyantoro. All rights reserved. Dashboard by Colorlib.

About Support Contact Us

Values-labelled Timeline

Questions:

1. From the cases above, which one do you prefer? Why?
2. Will this dashboard be useful in software development?
    a. What role in software development do you think will benefit from this dashboard? How/Why? (users – developer – designer – project manager – product owner)
    b. Which software development task do you think will benefit from this dashboard? How/Why? (requirements – design – implementation – evaluation)
    c. How will you use this dashboard?
    d. How should the identification of values be embedded into the dashboard?
3. To/In which artefacts should the identification values carried out?
4. Are there anything you want to have on the dashboard to make it more useful to you?

Appendix C

Interview Questions for the Dashboard's Feedback Study

Topics for the Feedback Interview

Software Practitioners

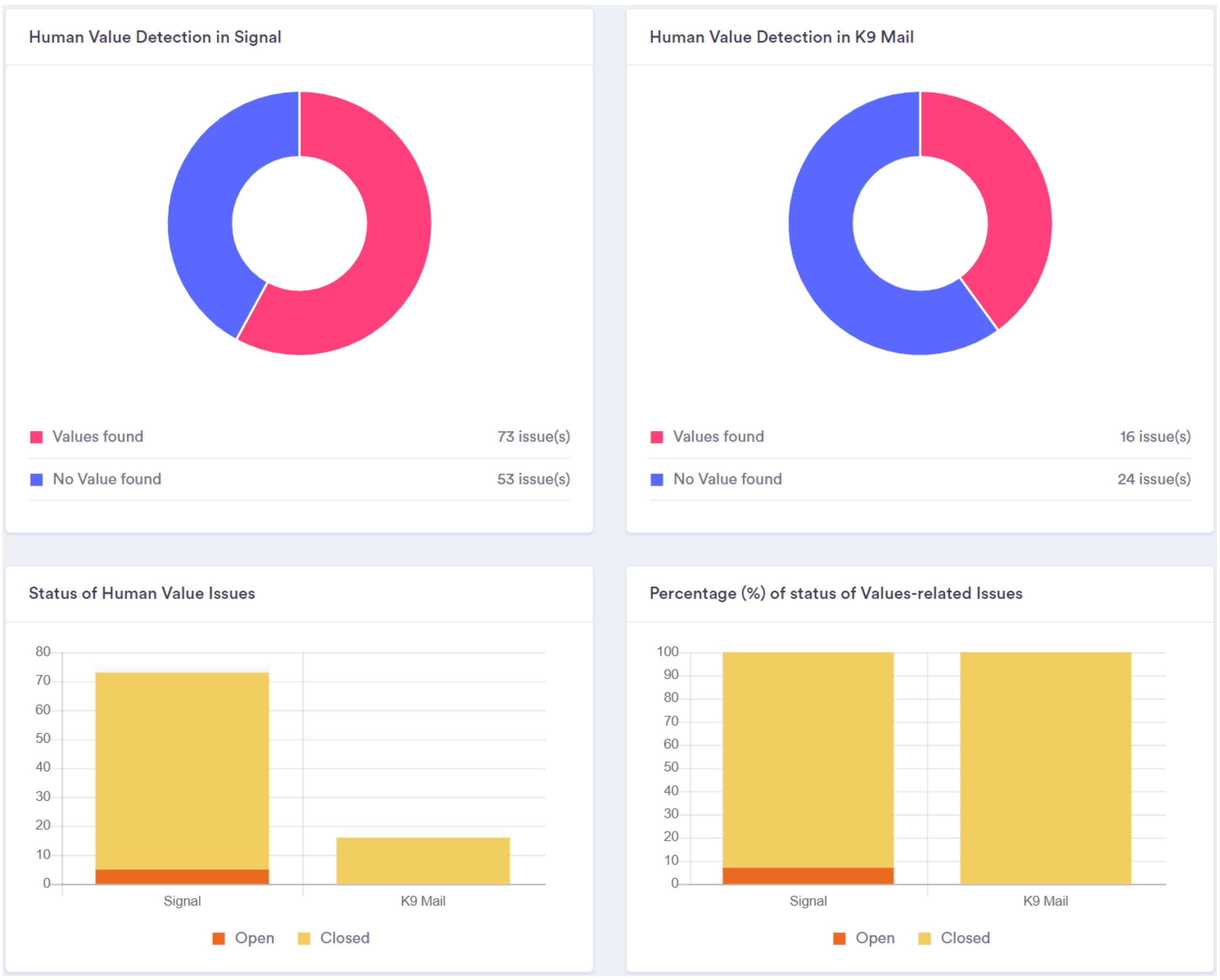
Estimated time: 30 minutes

Part 0 – Professional Backgrounds

1. How long have you been involved in software development projects?
2. What is your usual role in software development projects?
3. Do you have any experience using software repositories in the projects (e.g., GitHub, Bitbucket, etc.)?

Part 1 – Human Values Dashboard Demo

Demonstration of the Human Values Dashboard to the participants


Human Value Detection in Signal
Values found 73 issue(s)
No Value found 53 issue(s)
Human Value Detection in K9 Mail
Values found 16 issue(s)
No Value found 24 issue(s)
Status of Human Value Issues
80 70 60 50 40 30 20 10 0
Signal
K9 Mail
Open
Closed
Percentage (%) of status of Values-related Issues
100 90 80 70 60 50 40 30 20 10 0
Signal
K9 Mail
Open
Closed

Return to Project List

Filter Value Labels

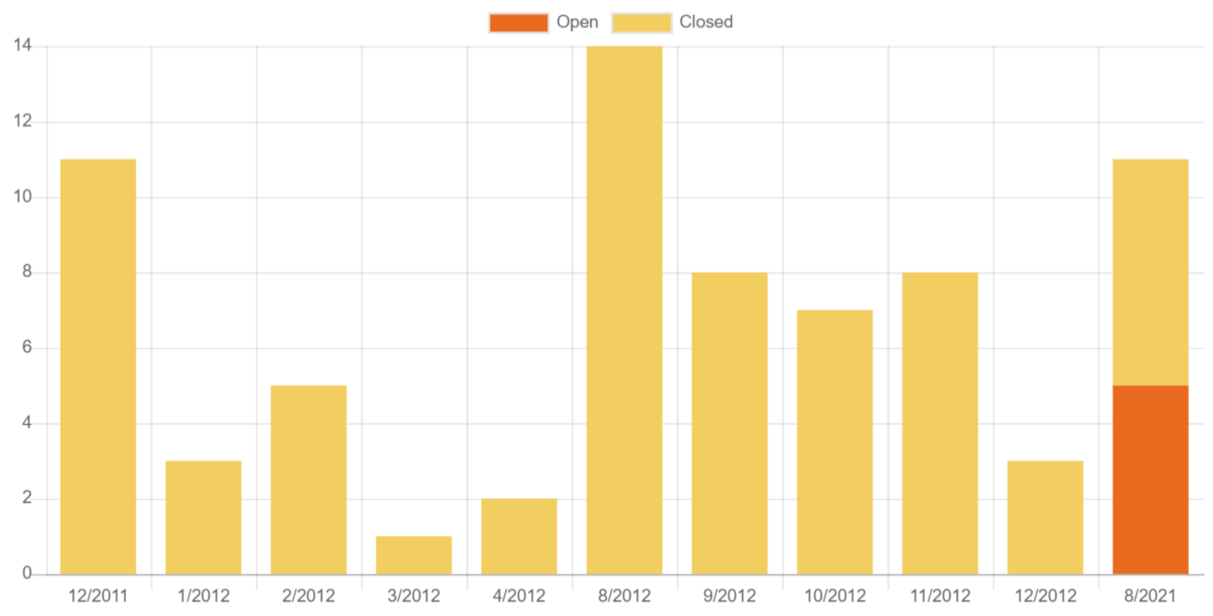
Message gets lost if user has signal account but not actively using it Values 1
#11568 opened on Aug 25 2021 by sarunluitel

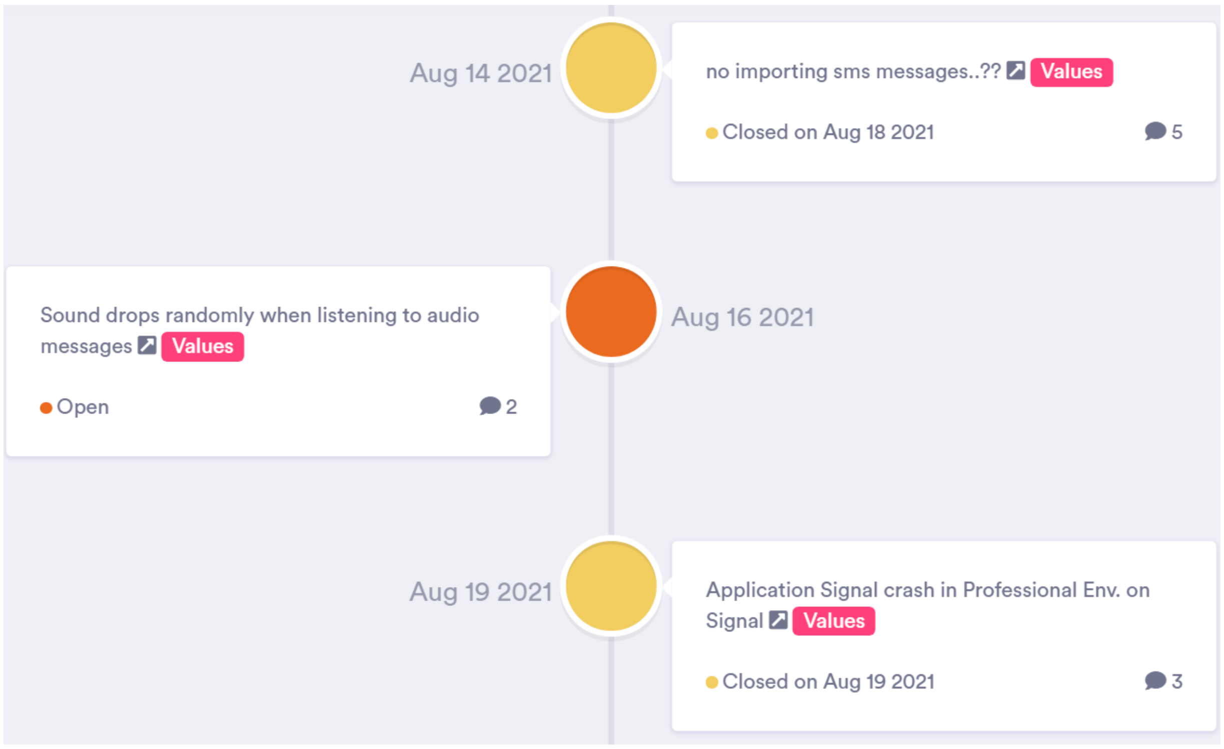
Fixes #11547 Update exoplayer v2.15.0 No Value 2
#11562 opened on Aug 22 2021 by makp9k

Verification code never received [Chinese phone number, previously working] Values 2
#11561 opened on Aug 22 2021 by ksl1989

Self-note : can't "delete for all" on phone Values 7
#11560 opened on Aug 22 2021 by Its-Just-Nans

Some animated stickers play really quickly in android app No Value 1
#11559 opened on Aug 22 2021 by parrajustin




Aug 14 2021

no importing sms messages..?? Values

● Closed on Aug 18 2021 5

Aug 16 2021

Sound drops randomly when listening to audio messages Values

● Open 2

Aug 19 2021

Application Signal crash in Professional Env. on Signal Values

● Closed on Aug 19 2021 3

Part 2 – Interviews for Feedback

Questions about the Dashboard

1. Does the dashboard show the information that you expect?

2. In the context of your current software development project, would this dashboard be useful?

3. Does the accuracy of the automated human values detection matter for practical use?

    a. Do you prefer to have an issue labelled as value-theme, but it is not (false positives) or do you prefer to have issue labelled as no value present, but it actually values are present (false negatives)?
    b. At what point the results is tolerable to you?

4. Do you have any suggestions to improve the dashboard?
    a. in terms of functionality?
    b. in terms of layout?
    c. any other ….

Bibliography

[1] Shalom H. Schwartz. An overview of the Schwartz Theory of Basic Values. *Online Readings in Psychology and Culture*, 2(1):12–13, 2012. ISSN 2307-0919. doi: 10.9707/2307-0919.1116.

[2] Shivali Best. Whatsapp loses millions of users to rivals Telegram and Signal amid fears of increased data sharing with Facebook, 2021. URL https://www.dailymail.co.uk/sciencetech/article-9183553/WhatsApp-loses-MILLIONS-users-rivals-Telegram-Signal-ahead-privacy-policy-update.html. Accessed: April 28, 2021.

[3] Nicholas Confessore. Cambridge Analytica and Facebook: The scandal and the fallout so far, 2018. URL https://www.nytimes.com/2018/04/04/us/politics/cambridge-analytica-scandal-fallout.html. Accessed: April 28, 2021.

[4] Avi-Asher Schapiro and Umberto Bacchi. U.S. protests fuel calls for ban on racially biased facial recognition tools, 2020. URL https://www.reuters.com/article/us-usa-protests-tech-trfn-idUSKBN23B3B5. Accessed: April 28, 2021.

[5] Hsiu-Yu Wang, Chechen Liao, and Ling-Hui Yang. What affects mobile application use? the roles of consumption values. *International Journal of Marketing Studies*, 5(2), 2013. ISSN 1918-719X. doi: 10.5539/ijms.v5n2p11.

[6] Bin Fu, Jialiu Lin, Lei Liy, Christos Faloutsos, Jason Hong, and Norman Sadeh. Why people hate your App - Making sense of user feedback in a mobile app store. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1276–1284, 2013. ISBN 9781450321747. doi: 10.1145/2487575.2488202.

[7] Davoud Mougouei, Harsha Perera, Waqar Hussain, Rifat Shams, and Jon Whittle. Operationalizing human values in software: a research roadmap. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering*

Conference and Symposium on the Foundations of Software Engineering, pages 780–784, 2018. ISBN 9781450355735. doi: 10.1145/3236024.3264843.

[8] Maria Angela Ferrario, Will Simm, Stephen Forshaw, Adrian Gradinar, Marcia Tavares Smith, and Ian Smith. Values-first SE. In *Proceedings of the 38th International Conference on Software Engineering Companion - ICSE '16*, pages 553–562, 2016. ISBN 9781450342056. doi: 10.1145/2889160.2889219.

[9] Harsha Perera, Waqar Hussain, Davoud Mougouei, Rifat Ara Shams, Arif Nurwidyantoro, and Jon Whittle. Towards integrating human values into software: Mapping principles and rights of GDPR to values. In *Proceedings of the IEEE 27th International Requirements Engineering Conference*, pages 404–409, 2019. ISBN 978-1-7281-3913-5. doi: 10.1109/RE.2019.00053.

[10] Harsha Perera, Waqar Hussain, Jon Whittle, Arif Nurwidyantoro, Davoud Mougouei, Rifat Ara Shams, and Gillian Oliver. A study on the prevalence of human values in software engineering publications, 2015 – 2018. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 409–420, 2020. ISBN 9781450371216. doi: 10.1145/3377811.3380393.

[11] Harsha Perera, Gunter Mussbacher, Waqar Hussain, Rifat Ara Shams, Arif Nurwidyantoro, and Jon Whittle. Continual Human Value Analysis in Software Development: A Goal Model Based Approach. In *Proceedings of the IEEE International Conference on Requirements Engineering*, pages 192–203, 2020. ISBN 9781728174389. doi: 10.1109/RE48521.2020.00030.

[12] Rifat Ara Shams, Waqar Hussain, Gillian Oliver, Arif Nurwidyantoro, Harsha Perera, and Jon Whittle. Society-oriented applications development: Investigating users' values from Bangladeshi agriculture mobile applications. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering: Software Engineering in Society*, ICSE-SEIS '20, page 53–62, 2020. ISBN 9781450371244. doi: 10.1145/3377815.3381382.

[13] Batya Friedman, Peter H. Kahn, Jr., and Peter H. Kahn Jr. Human values, ethics, and design. In *The Human-Computer Interaction Handbook*, pages 1177–1209. 2003. ISBN 0-8058-3838-4.

[14] Batya Friedman, Peter H. Kahn, Jr., and Alan Borning. Value sensitive design and information systems. In *The Handbook of Information and Computer Ethics*, pages 69–101. 2008.

[15] Sarah Thew and Alistair Sutcliffe. Value-based requirements engineering: method and experience. *Requirements Engineering*, 23(4):443–464, 2018. ISSN 1432010X. doi: 10.1007/s00766-017-0273-y.

[16] Huib Aldewereld, Virginia Dignum, and Yao Hua Tan. Design for values in software development. In *Handbook of Ethics, Values, and Technological Design: Sources, Theory, Values and Application Domains*, pages 831–845. 2015. ISBN 9789400769700. doi: 10.1007/978-94-007-6970-0_26.

[17] Langdon Winner. Do artifacts have politics? *Daedalus*, pages 121–136, 1980.

[18] Seil Kim, Jae Ik Cho, Hee Won Myeong, and Dong Hoon Lee. A study on static analysis model of mobile application for privacy protection. In *Computer Science and Convergence*, pages 529–540. Springer, 2012. ISBN 978-94-007-2792-2.

[19] Clint Gibler, Jonathan Crussell, Jeremy Erickson, and Hao Chen. Androidleaks: Automatically detecting potential privacy leaks in Android applications on a large scale. In *Proceedings of the International Conference on Trust and Trustworthy Computing*, volume 7344 LNCS, pages 291–307, 2012. ISBN 9783642309205. doi: 10.1007/978-3-642-30921-2_17.

[20] Sultan S Alqahtani and Juergen Rilling. An ontology-based approach to automate tagging of software artifacts. In *Proceedings of the 2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 169–174, 2017. ISBN 9781509040391. doi: 10.1109/ESEM.2017.25.

[21] Yaqin Zhou and Asankhaya Sharma. Automated identification of security issues from commit messages and bug reports. In *Proceedings of the 11th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2017, page 914–919, 2017. ISBN 9781450351058. doi: 10.1145/3106237.3117771.

[22] John Viega, J. T. Bloch, Tadayoshi Kohno, and Gary McGraw. Token-based scanning of source code for security problems. *ACM Transactions on Information and System Security*, 5(3):238–261, 2002. ISSN 10949224. doi: 10.1145/545186.545188.

[23] Felix Fischer, Konstantin Bottinger, Huang Xiao, Christian Stransky, Yasemin Acar, Michael Backes, and Sascha Fahl. Stack Overflow considered harmful? the impact of copy & paste on Android application security. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 121–136, 2017. ISBN 9781509055326. doi: 10.1109/SP.2017.31.

[24] Mohammad Naseri, Nataniel P. Borges, Andreas Zeller, and Romain Rouvoy. Accessileaks: Investigating privacy leaks exposed by the Android accessibility service. In *Proceedings on Privacy Enhancing Technologies*, pages 291–305, 2019. doi: 10.2478/popets-2019-0031.

[25] Lingfeng Bao, David Lo, Xin Xia, Xinyu Wang, and Cong Tian. How android app developers manage power consumption? In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 37–48. ACM, 2016. ISBN 9781450341868. doi: 10.1145/2901739.2901748.

[26] Rui Pereira, Tiago Carcao, Marco Couto, Jacome Cunha, Joao Paulo Fernandes, and Joao Saraiva. Helping programmers improve the energy efficiency of source code. In *Proceedings of the 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, pages 238–240, 2017. ISBN 9781538615898. doi: 10.1109/ICSE-C.2017.80.

[27] Shalom H Schwartz. Are there universal aspects in the structure and contents of human values? *Journal of Social Issues*, 50(4):19–45, 1994. ISSN 15404560. doi: 10.1111/j.1540-4560.1994.tb01196.x.

[28] Shalom H. Schwartz. The refined theory of basic values. In *Values and Behavior: Taking a Cross Cultural Perspective*, pages 51–72. 2017. ISBN 9783319563527. doi: 10.1007/978-3-319-56352-7_3.

[29] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M German, and Daniela Damian. The promises and perils of mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 92–101, 2014. ISBN 9781450328630. doi: 10.1145/2597073.2597074.

[30] GitHub. About issues, . URL https://docs.github.com/en/issues/tracking-your-work-with-issues/about-issues. Accessed: October 1, 2021.

[31] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. Social coding in GitHub: Transparency and collaboration in an open software repository. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work, CSCW*, pages 1277–1286, 2012. ISBN 9781450310864. doi: 10.1145/2145204.2145396.

[32] Tegawende F. Bissyande, David Lo, Lingxiao Jiang, Laurent Reveillere, Jacques Klein, and Yves Le Traon. Got issues? who cares about it? a large scale investigation of issue trackers from GitHub. In *Proceedings of the IEEE 24th International*

Symposium on Software Reliability Engineering, ISSRE 2013, pages 188–197, 2013. ISBN 9781479923663. doi: 10.1109/ISSRE.2013.6698918.

[33] Hideaki Hata, Nicole Novielli, Sebastian Baltes, Raula Gaikovina Kula, and Christoph Treude. Github discussions: An exploratory study of early adoption. *Empirical Software Engineering*, 27(1):1–32, 2022.

[34] Tao Zhang, Jiachi Chen, Xiapu Luo, and Tao Li. Bug reports for desktop software and mobile apps in GitHub: What's the difference? *IEEE Software*, 36(1):63–71, 2019. ISSN 19374194. doi: 10.1109/MS.2017.377142400.

[35] Risto Salo, Timo Poranen, and Zheying Zhang. Requirements management in GitHub with a lean approach. In *Proceedings of the 14th Symposium on Programming Languages and Software Tools (SPLST'15)*, volume 1525, pages 164–178, 2015.

[36] João Brunet, Gail C. Murphy, Ricardo Terra, Jorge Figueiredo, and Dalton Serey. Do developers discuss design? In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 340–343, 2014. doi: 10.1145/2597073.2597115.

[37] Giovanni Viviani, Calahan Janik-Jones, Michalis Famelis, and Gail C. Murphy. The structure of software design discussions. In *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 104–107. ACM, 2018. ISBN 9781450357258. doi: 10.1145/3195836.3195841.

[38] An Shou Cheng and Kenneth R. Fleischmann. Developing a meta-inventory of human values. In *Proceedings of the ASIST Annual Meeting*, volume 47, 2010. ISBN 1550-8390. doi: 10.1002/meet.14504701232.

[39] Milton Rokeach. *The Nature of Human Values.* Free press, 1973.

[40] Robin M. Williams Jr. Change and stability in values and value systems: A sociological perspective. In *Understanding Human Values*, pages 15–46. 1979. ISBN 9781439118887.

[41] Pat Duffy Hutcheon. Value theory: Towards conceptual clarification. *The British Journal of Sociology*, 23(2):172–187, 1972. ISSN 00071315, 14684446.

[42] Valerie Braithwaite. Consensus, stability and meaning in abstract social values. *Australian Journal of Political Science*, 33(3):363–380, 1998. doi: 10.1080/10361149850525.

[43] Paul H.P. Hanel, Gregory R. Maio, Ana K.S. Soares, Katia C. Vione, Gabriel L.de Holanda Coelho, Valdiney V. Gouveia, Appasaheb C. Patil, Shanmukh V. Kamble, and Antony S.R. Manstead. Cross-cultural differences and similarities in human value instantiation. *Frontiers in Psychology*, 9:1–13, 2018. ISSN 16641078. doi: 10.3389/fpsyg.2018.00849.

[44] Tim Holmes, Elena Blackmore, Richard Hawkins, and Tom Wakeford. *The Common Cause Handbook.* Public Interest Research Center, 2011.

[45] Shalom H. Schwartz and Wolfgang Bilsky. Toward a theory of the universal content and structure of values: Extensions and cross-cultural replications. *Journal of Personality and Social Psychology*, 58(5):878–891, 1990.

[46] Juan A Barceló, Florencia Del Castillo Bernal, Ricardo Del Olmo, Laura Mameli, FJ Miguel Quesada, David Poza, and Xavier Vilà. Social interaction in hunter-gatherer societies: simulating the consequences of cooperation and social aggregation. *Social Science Computer Review*, 32(3):417–436, 2014. doi: 10.1177/0894439313511943.

[47] Md Saddam Hossain Mukta, Euna Mehnaz Khan, Mohammed Eunus Ali, and Jalal Mahmud. Predicting movie genre preferences from personality and values of social media users. In *Proceedings of the Eleventh International AAAI Conference on Web and Social Media*, 2017.

[48] Barry Boehm and Li Guo Huang. Value-based software engineering: A case study. *Computer*, (3):33–41, 2003. doi: 10.1109/MC.2003.1185215.

[49] Hourieh Khalajzadeh, Mojtaba Shahin, Humphrey O. Obie, and John Grundy. How are Diverse End-user Human-centric Issues Discussed on GitHub? In *Proceedings of the 44th IEEE/ACM International Conference on Software Engineering - Software Engineering in Society*, 2022.

[50] Emily Winter, Stephen Forshaw, Lucy Hunt, and Maria Angela Ferrario. Advancing the study of human values in software engineering. In *Proceedings of the 2019 IEEE/ACM 12th International Workshop on Cooperative and Human Aspects of Software Engineering*, pages 19–26, 2019. ISBN 9781728122397. doi: 10.1109/CHASE.2019.00012.

[51] Waqar Hussain, Harsha Perera, Jon Whittle, Arif Nurwidyantoro, Rashina Hoda, Rifat Shams, and Gillian Oliver. Human values in software engineering: Contrasting case studies of practice. *IEEE Transactions on Software Engineering*, (1): 1–1, Nov 2020. ISSN 1939-3520. doi: 10.1109/TSE.2020.3038802.

[52] Batya Friedman, Peter H. Kahn, Alan Borning, and Alina Huldtgren. Value sensitive design and information systems. In *Early Engagement and New Technologies: Opening Up the Laboratory*, pages 55–95. 2013. doi: 10.1007/978-94-007-7844-3_4.

[53] Dewi Mairiza, Didar Zowghi, and Nurie Nurmuliani. An investigation into the notion of non-functional requirements. In *Proceedings of the ACM Symposium on Applied Computing*, pages 311–317, 2010. ISBN 9781605586380. doi: 10.1145/1774088.1774153.

[54] Martin Glinz. On non-functional requirements. In *Proceedings of the 15th IEEE International Requirements Engineering Conference*, pages 21–26, 2007. ISBN 0769529356. doi: 10.1109/RE.2007.45.

[55] Balbir S. Barn. Do you own a volkswagen? values as non-functional requirements. In *Proceedings of the Joint 6th International Working Conference on Human-Centred Software Engineering and 8th International Working Conference on Human Error, Safety, and System Development*, 2016. ISBN 9783319449012. doi: 10.1007/978-3-319-44902-9_10.

[56] Daniel Pletea, Bogdan Vasilescu, and Alexander Serebrenik. Security and emotion: sentiment analysis of security discussions on GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 348–351, 2014. ISBN 9781450328630. doi: 10.1145/2597073.2597117.

[57] Li Li, Alexandre Bartel, Tegawendé F Bissyandé, Jacques Klein, Yves Le Traon, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Damien Octeau, and Patrick McDaniel. IccTA: Detecting inter-component privacy leaks in Android apps. In *Proceedings of the International Conference on Software Engineering*, pages 280–291, 2015. ISBN 9781479919345. doi: 10.1109/ICSE.2015.48.

[58] Konstantin Kuznetsov, Vitalii Avdiienko, Alessandra Gorla, and Andreas Zeller. Checking app user interfaces against app descriptions. In *Proceedings of the International Workshop on App Market Analytics*, pages 1–7, 2016. ISBN 9781450343985. doi: 10.1145/2993259.2993265.

[59] Vibhu Saujanya Sharma, Roshni R Ramnani, and Shubhashis Sengupta. A framework for identifying and analyzing non-functional requirements from text. In *Proceedings of the 4th International Workshop on Twin Peaks of Requirements and Architecture*, pages 1–8, 2014. ISBN 9781450328487. doi: 10.1145/2593861.2593862.

[60] Emi Ishita, Douglas W. Oard, Kenneth R. Fleischmann, and Thomas Clay Templeton. Investigating multi-label classification for human values. In *Proceedings of the 73rd ASIS&T Annual Meeting on Navigating Streams in an Information Ecosystem*, 2010. ISBN 0020-6539.

[61] Yasuhiro Takayama, Yoichi Tomiura, Emi Ishita, Douglas W. Oard, Kenneth R. Fleischmann, and An-Shou Cheng. A word-scale probabilistic latent variable model for detecting human values. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, pages 1489–1498, 2014. ISBN 9781450325981. doi: 10.1145/2661829.2661966.

[62] Emi Ishita, Toru Oga, An Shou Cheng, Kenneth R. Fleischmann, Takayama Yasuhiro, Douglas W. Oard, and Yoichi Tomiura. Toward automating detection of human values in the nuclear power debate. *Proceedings of the Association for Information Science and Technology*, 54(1):714–715, 2017. ISSN 23739231. doi: 10.1002/pra2.2017.14505401127.

[63] Emi Ishita, Satoshi Fukuda, Toru Oga, Douglas W. Oard, Kenneth R. Fleischmann, Yoichi Tomiura, and An Shou Cheng. Toward three-stage automation of annotation for human values. In *Information in Contemporary Society*, pages 188–199, 2019. ISBN 9783030157418. doi: 10.1007/978-3-030-15742-5_18.

[64] Yasuhiro Takayama, Yoichi Tomiura, Emi Ishita, Zheng Wang, Douglas W. Oard, Kenneth R. Fleischmann, and An-Shou Cheng. Improving automatic sentence-level annotation of human values using augmented feature vectors. In *Proceedings of the Conference of the Pacific Association for Computational Linguistics (PACLING)*, 2013.

[65] Emi Ishita, Douglas W. Oard, Kenneth R. Fleischmann, Yoichi Tomiura, Yasuhiro Takayama, and An Shou Cheng. Learning curves for automating content analysis: How much human annotation is needed? In *Proceedings of the 2015 IIAI 4th International Congress on Advanced Applied Informatics*, pages 171–176, 2015. ISBN 9781479999583. doi: 10.1109/IIAI-AAI.2015.295.

[66] Yasuhiro Takayama, Yoichi Tomiura, Kenneth R. Fleischmann, An-Shou Cheng, Douglas W. Oard, and Emi Ishita. Automatic dictionary extraction and content analysis associated with human values. *Information Engineering Express*, 1(4): 107–118, 2015. doi: doi.org/10.52731/iee.v1.i4.34.

[67] Yasuhiro Takayama, Yoichi Tomiura, Kenneth R. Fleischmann, An Shou Cheng, Douglas W. Oard, and Emi Ishita. An automatic dictionary extraction and annotation method using simulated annealing for detecting human values. In *Proceedings of the 2015 IIAI 4th International Congress on Advanced Applied Informatics*, pages 177–182, 2016. ISBN 9781479999583. doi: 10.1109/IIAI-AAI.2015.268.

[68] Rocky Slavin, Xiaoyin Wang, Mitra Bokaei Hosseini, James Hester, Ram Krishnan, Jaspreet Bhatia, Travis D Breaux, and Jianwei Niu. Toward a framework for detecting privacy policy violations in Android application code. In *Proceedings of the 38th IEEE International Conference on Software Engineering*, pages 25–36, 2016. ISBN 9781450339001. doi: 10.1145/2884781.2884855.

[69] Seifeddine Bettaieb, Seung Yeob Shin, Mehrdad Sabetzadeh, Lionel C. Briand, Michael Garceau, and Antoine Meyers. Using machine learning to assist with the selection of security controls during security assessment. *Empirical Software Engineering*, 25(4):2550–2582, 2020. ISSN 15737616. doi: 10.1007/s10664-020-09814-x.

[70] Mansooreh Zahedi, Muhammad Ali Babar, and Christoph Treude. An empirical study of security issues posted in open source projects. In *Proceedings of the 51st Hawaii International Conference on System Sciences*, 2018. ISBN 10125/50575. doi: 10.24251/hicss.2018.686.

[71] David N. Palacio, Daniel McCrystal, Kevin Moran, Carlos Bernal-Cardenas, Denys Poshyvanyk, and Chris Shenefiel. Learning to identify security-related issues using convolutional neural networks. In *Proceedings of the 2019 IEEE International Conference on Software Maintenance and Evolution*, pages 140–144, 2019. ISBN 9781728130941. doi: 10.1109/ICSME.2019.00024.

[72] Ali Rezaei Nasab, Mojtaba Shahin, Peng Liang, Mohammad Ehsan Basiri, Seyed Ali Hoseyni Raviz, Hourieh Khalajzadeh, Muhammad Waseem, and Amineh Naseri. Automated identification of security discussions in microservices systems: Industrial surveys and experiments. *Journal of Systems and Software*, 181, 2021. ISSN 01641212. doi: 10.1016/j.jss.2021.111046.

[73] Marco Couto, Jácome Cunha, João Paulo Fernandes, Rui Pereira, and João Saraiva. GreenDroid: A tool for analysing power consumption in the Android ecosystem. In *Proceedings of the 2015 IEEE 13th International Scientific Conference on Informatics*, pages 73–78, 2015. ISBN 9781467398688. doi: 10.1109/Informatics.2015.7377811.

[74] Conghui Li, Humphrey O. Obie, and Hourieh Khalajzadeh. A first step towards detecting human values-violating defects in Android APIs. In *Proceedings of the 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pages 53–58. IEEE, 2021. ISBN 9781665435833. doi: 10.1109/asew52652.2021.00022.

[75] Steve Wexler, Jeffrey Shaffer, and Andy Cotgreave. *The Big Book of Dashboards: Visualizing Your Data Using Real World Business Scenarios.* 2017.

[76] Andrea Janes, Alberto Sillitti, and Giancarlo Succi. Effective dashboard design. *Cutter IT Journal*, 26(1):17–24, 2013. ISSN 15227383.

[77] Vladimir Ivanov, Alan Rogers, Giancarlo Succi, Jooyong Yi, and Vasiii Zorin. Precooked developer dashboards: What to show and how to use. In *International Conference on Software Engineering: Companion*, pages 402–403, 2018. ISBN 9781450356633. doi: 10.1145/3183440.3195028.

[78] Vladimir Ivanov, Vladislav Pischulin, Alan Rogers, Giancarlo Succi, Jooyong Yi, and Vasilii Zorin. Design and validation of precooked developer dashboards. In *Proceedings of the 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 821–826. ACM, 2018. ISBN 9781450355735. doi: 10.1145/3236024.3275530.

[79] Christoph Treude and Margaret Anne Storey. Awareness 2.0: Staying aware of projects, developers and tasks using dashboards and feeds. In *Proceedings of the 32nd International Conference on Software Engineering*, pages 365–374, 2010. ISBN 9781605587196. doi: 10.1145/1806799.1806854.

[80] Olga Baysal, Reid Holmes, and Michael W. Godfrey. Developer dashboards: The need for qualitative analytics. *IEEE Software*, 30(4):46–52, 2013. ISSN 07407459. doi: 10.1109/MS.2013.66.

[81] L. López, M. Manzano, C. Gómez, M. Oriol, C. Farré, X. Franch, S. Martínez-Fernández, and A. M. Vollmer. QaSD: A quality-aware strategic dashboard for

supporting decision makers in agile software development. *Science of Computer Programming*, 202:102568, 2021. ISSN 01676423. doi: 10.1016/j.scico.2020.102568.

[82] George K. Thiruvathukal, Shilpika, Nicholas J. Hayward, and Konstantin Läufer. Metrics dashboard: A hosted platform for software quality metrics, 2018.

[83] Larissa Leite, Christoph Treude, and Fernando Figueira Filho. UE dashboard: Awareness of unusual events in commit histories. In *Proceedings of the 10th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering*, pages 978–981, 2015. ISBN 9781450336758. doi: 10.1145/2786805.2803184.

[84] Christoph Treude and Margaret-anne Storey. Concernlines: A timeline view of co-occurring concerns. In *Proceedings of the 31st International Conference on Software Engineering*, pages 575–578, 2009. ISBN 9781424434527.

[85] Cauldron. Level up software development analytics. URL https://cauldron.io/explore. Accessed: April 18, 2021.

[86] Mautic. Mautic community dashboard. URL https://dashboard.mautic.org/. Accessed: April 18, 2021.

[87] GitHub. About your personal dashboard, . URL https://docs.github.com/en/github/setting-up-and-managing-your-github-user-account/about-your-personal-dashboard}accessing-your-personal-dashboard. Accessed: April 18, 2021.

[88] GitHub. About your organization dashboard, . URL https://docs.github.com/en/organizations/collaborating-with-groups-in-organizations/about-your-organization-dashboard. Accessed: April 18, 2021.

[89] Tamara Lopez, Thein T. Tun, Arosha Bandara, Mark Levine, Bashar Nuseibeh, and Helen Sharp. An anatomy of security conversations in Stack Overflow. In *2019 International Conference on Software Engineering: Software Engineering in Society*, pages 31–40, 2019. ISBN 9781728117621. doi: 10.1145/3194707.3194713.

[90] Kenneth R Fleischmann, Douglas W Oard, An Shou Cheng, Ping Wang, and Emi Ishita. Automatic classification of human values: Applying computational thinking to information ethics. In *Proceedings of the 72nd Annual Meeting of the American Society for Information Science and Technology*, pages 6–11, 2009. doi: 10.1002/meet.2009.1450460345. URL https://onlinelibrary-wiley-com.ezproxy.lib.monash.edu.au/doi/pdf/10.1002/meet.2009.1450460345.

[91] Marvin Wyrich, Andreas Preikschat, Daniel Graziotin, and Stefan Wagner. The mind is a powerful place: How showing code comprehensibility metrics influences code understanding. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, pages 512–523. IEEE, 2021.

[92] Jil Klünder, Nils Prenner, Ann-Kathrin Windmann, Marek Stess, Michael Nolting, Fabian Kortum, Lisa Handke, Kurt Schneider, and Simone Kauffeld. Do you just discuss or do you solve? meeting analysis in a software project at early stages. In *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops*, pages 557–562, 2020.

[93] Parastou Tourani, Bram Adams, and Alexander Serebrenik. Code of conduct in open source projects. In *2017 IEEE 24th international conference on software analysis, evolution and reengineering (SANER)*, pages 24–33. IEEE, 2017.

[94] GitHub. Github issues - project planning for developers, . URL https://github.com/features/issues/. Accessed: October 1, 2021.

[95] GitHub. Autolinked references and urls, . URL https://docs.github.com/en/github/writing-on-github/working-with-advanced-formatting/autolinked-references-and-urls. Accessed: September 23, 2021.

[96] Dataset of human values in Github Issues. URL https://github.com/ovislabmonash/values-issues-dataset. Accessed: December 31, 2020.

[97] Giovanni Viviani, Calahan Janik-Jones, Michalis Famelis, Xin Xia, and Gail C. Murphy. What design topics do developers discuss? In *Proceedings of the 26th Conference on Program Comprehension*, pages 328–331. ACM, 2018. ISBN 9781450357142. doi: 10.1145/3196321.3196357.

[98] Giovanni Viviani, Michalis Famelis, Xin Xia, Calahan Janik-Jones, and Gail C. Murphy. Locating latent design information in developer discussions: A study on pull requests. *IEEE Transactions on Software Engineering*, 2019. ISSN 19393520. doi: 10.1109/TSE.2019.2924006.

[99] Oskar Jarczyk, Blazej Gruszka, Szymon Jaroszewicz, Leszek Bukowski, and Adam Wierzbicki. Github projects. quality analysis of open-source software. In *Proceedings of the International Conference on Social Informatics*, pages 80–94, 2014. ISBN 978-3-319-13734-6. doi: 10.1007/978-3-319-13734-6_6.

[100] Giuseppe Destefanis, Marco Ortu, David Bowes, Michele Marchesi, and Roberto Tonelli. On measuring affects of Github issues' commenters. In *Proceedings of the 3rd International Workshop on Emotion Awareness in Software Engineering*, pages 14–19, 2018. ISBN 9781450357517. doi: 10.1145/3194932.3194936.

[101] Jason Tsay, Laura Dabbish, and James Herbsleb. Let's talk about it: Evaluating contributions through discussion in GitHub. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 144–154, 2014. ISBN 9781450330565. doi: 10.1145/2635868.2635882.

[102] StatCounter. Mobile operating system market share worldwide. URL https://gs.statcounter.com/os-market-share/mobile/worldwide. Accessed: August 15, 2021.

[103] M. Hosseinkhani Loorak, Philip W.L. Fong, and Sheelagh Carpendale. Papilio: Visualizing Android application permissions. In *Computer Graphics Forum*, volume 33, pages 391–400. Wiley Online Library, 2014. doi: 10.1111/cgf.12395.

[104] Karina Sokolova, Charles Perez, and Marc Lemercier. Android application classification and anomaly detection with graph-based permission patterns. *Decision Support Systems*, 93:62–76, 2017. doi: 10.1016/j.dss.2016.09.006.

[105] Open Whisper Systems. Announcing the public beta, 2010. URL https://web.archive.org/web/20100530011131/http://www.whispersys.com/updates.html. Accessed: January 1, 2020.

[106] Moxie Marlinspike. Saying goodbye to encrypted sms/mms, 2015. URL https://signal.org/blog/goodbye-encrypted-sms/. Accessed: January 1, 2020.

[107] Signal. Signal. URL https://signal.org/en/. Accessed: December 1, 2021.

[108] K9 Mail. PGP/MIME encryption, . URL https://k9mail.github.io/documentation/security/pgpmime.html. Accessed: December 31, 2020.

[109] Vincent Breitmoser. OpenPGP considerations, Part III: Autocrypt and encryption by default, Feb 2018. URL https://k9mail.github.io/2018/02/26/OpenPGP-Considerations-Part-III-Autocrypt.html. Accessed: January 1, 2020.

[110] K9 Mail. K9 mail, . URL https://k9mail.app. Accessed: December 1, 2021.

[111] Alice Wyman and et al. What is Firefox Focus?, Sep 2020. URL http://mzl.la/1NDD2IB. Accessed: September 30, 2020.

[112] Firefox Focus. Simply private mobile browsing. URL https://www.mozilla.org/en-US/firefox/browsers/mobile/focus/. Accessed: December 1, 2021.

[113] Emily Winter, Steve Forshaw, and Maria Angela Ferrario. Measuring Human Values in Software Engineering. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 8–11, 2018. ISBN 9781450358231. doi: 10.1145/3239235.3267427.

[114] Gregory R Maio. Mental representations of social values. In *Advances in Experimental Social Psychology*, volume 42, pages 1–43. Elsevier, 2010. doi: 10.1016/S0065-2601(10/42001-8.

[115] Steve Adolph, Wendy Hall, and Philippe Kruchten. A methodological leg to stand on: Lessons learned using grounded theory to study software development. In *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds*, pages 13:166–13:178, 2008. doi: 10.1007/s10664-010-9152-6.

[116] Paul H.P. Hanel, Katia C. Vione, Ulrike Hahn, and Gregory R. Maio. Value instantiations: The missing link between values and behavior? In *Values and Behavior: Taking a Cross Cultural Perspective*, pages 175–190. 2017. ISBN 9783319563527. doi: 10.1007/978-3-319-56352-7_8.

[117] Nishant Jha and Anas Mahmoud. Mining non-functional requirements from App store reviews. *Empirical Software Engineering*, 24(6):3659–3695, 2019. ISSN 15737616. doi: 10.1007/s10664-019-09716-7.

[118] Serdar Aslan and Osman Balci. GAMED: Digital educational game development methodology. *Simulation*, 91(4):307–319, 2015. ISSN 17413133. doi: 10.1177/0037549715572673.

[119] Hojjat Mahmoudi, Mohsen Koushafar, Javad Amani Saribagloo, and Ghasem Pashavi. The effect of computer games on speed, attention and consistency of learning mathematics among students. *Procedia - Social and Behavioral Sciences*, 176:419–424, feb 2015. ISSN 18770428. doi: 10.1016/j.sbspro.2015.01.491.

[120] Chia Hui Tsai, Ching Hsue Cheng, Duen Yian Yeh, and Shih Yun Lin. Can learning motivation predict learning achievement? A case study of a mobile game-based English learning approach. *Education and Information Technologies*, 22(5): 2159–2173, oct 2017. ISSN 15737608. doi: 10.1007/s10639-016-9542-5.

[121] Qiao Huang, Xin Xia, David Lo, and Gail C Murphy. Automating intention mining. *IEEE Transactions on Software Engineering*, 2018. ISSN 19393520. doi: 10.1109/TSE.2018.2876340.

[122] Marco Ortu, Alessandro Murgia, Giuseppe Destefanis, Parastou Tourani, Roberto Tonelli, Michele Marchesi, and Bram Adams. The emotional side of software developers in JIRA. In *Proceedings of the 13th International Conference on Mining Software Repositories*, pages 480–483, 2016. ISBN 9781450341868. doi: 10.1145/2901739.2903505.

[123] Karl Werder and Sjaak Brinkkemper. MEME – Toward a method for emotions extraction from GitHub. In *Proceedings of the ACM/IEEE 3rd International Workshop on Emotion Awareness in Software Engineering*, pages 20–24, 2018. ISBN 9781450357517. doi: 10.2307/j.ctvct0023.22.

[124] Jin Ding, Hailong Sun, Xu Wang, and Xudong Liu. Entity-level sentiment analysis of issue comments. In *Proceedings of the IEEE/ACM 3rd International Workshop on Emotion Awareness in Software Engineering*, volume 18, pages 7–13. ACM, 2018. ISBN 9781450357517. doi: 10.1145/3194932.3194935.

[125] Kristin Fjola Tomasdottir, Mauricio Aniche, and Arie Van Deursen. Why and how JavaScript developers use linters. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 578–589. APA, 2017. ISBN 9781538626849. doi: 10.1109/ASE.2017.8115668.

[126] Gede Artha Azriadi Prana, Christoph Treude, Ferdian Thung, Thushari Atapattu, and David Lo. Categorizing the content of GitHub README files. *Empirical Software Engineering*, 24(3):1296–1327, 2019. ISSN 15737616. doi: 10.1007/s10664-018-9660-3.

[127] Yuzhan Ma, Sarah Fakhoury, Michael Christensen, Venera Arnaoudova, Waleed Zogaan, and Mehdi Mirakhorli. Automatic classification of software artifacts in open-source applications. In *Proceedings of the IEEE/ACM 15th International Conference on Mining Software Repositories*, pages 414–425, 2018. ISBN 9781450357166. doi: 10.1145/3196398.3196446.

[128] Preetha Chatterjee, Kostadin Damevski, Lori Pollock, Vinay Augustine, and Nicholas A. Kraft. Exploratory study of Slack Q&A chats as a mining source for software engineering tools. pages 490–501, 2019. ISBN 9781728134123. doi: 10.1109/MSR.2019.00075.

[129] Shaohua Wang, Nhathai Phan, Yan Wang, and Yong Zhao. Extracting API tips from developer question and answer websites. In *Proceedings of the ACM/IEEE 16th International Working Conference on Mining Software Repositories*, pages 321–332. IEEE, 2019. ISBN 9781728134123. doi: 10.1109/MSR.2019.00058.

[130] Rafael Kallis, Andrea Di Sorbo, Gerardo Canfora, and Sebastiano Panichella. Predicting issue types on GitHub. *Science of Computer Programming*, 205, 2021. ISSN 01676423. doi: 10.1016/j.scico.2020.102598.

[131] Qiang Fan, Yue Yu, Gang Yin, Tao Wang, and Huaimin Wang. Where is the road for issue reports classification based on text mining? In *Proceedings of the ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 121–130, 2017. ISBN 9781509040391. doi: 10.1109/ESEM.2017.19.

[132] Jordi Cabot, Javier Luis Canovas Izquierdo, Valerio Cosentino, and Belen Rolandi. Exploring the use of labels to categorize issues in open-source software projects. In *Proceedings of the IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering*, pages 550–554, 2015. ISBN 9781479984695. doi: 10.1109/SANER.2015.7081875.

[133] GitHub. Managing labels, . URL https://docs.github.com/en/issues/using-labels-and-milestones-to-track-work/managing-labels. Accessed: October 1, 2021.

[134] Steven Bird, Ewan Klein, and Edward Loper. Natural language processing with Python. URL https://nltk.org/book. Accessed: November 26, 2021.

[135] GitHub. Mentioning people and teams, . URL https://docs.github.com/en/github/writing-on-github/getting-started-with-writing-and-formatting-on-github/basic-writing-and-formatting-syntax}mentioning-people-and-teams. Accessed: November 26, 2021.

[136] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan. *Introduction to Information Retrieval.* Cambridge University Press Cambridge, 2008.

[137] Bing Liu. *Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*. Cambridge University Press, 2020.

[138] Mike Thelwall, Kevan Buckley, Georgios Paltoglou, Di Cai, and Arvid Kappas. Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, 61(12):2544–2558, 2010. doi: 10.1002/asi.21416.

[139] Gias Uddin and Foutse Khomh. Automatic mining of opinions expressed about APIs in Stack Overflow. *IEEE Transactions on Software Engineering*, 47:522–559, 2019. ISSN 1939-3520. doi: 10.1109/TSE.2019.2900245.

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

[141] Nicole Novielli, Fabio Calefato, Filippo Lanubile, and Alexander Serebrenik. Assessment of off-the-shelf SE-specific sentiment analysis tools: An extended replication study. *Empirical Software Engineering*, 26(4):1–29, 2021.

[142] Mehdi Golzadeh, Alexandre Decan, Damien Legay, and Tom Mens. A ground-truth dataset and classification model for detecting bots in GitHub issue and PR comments. *Journal of Systems and Software*, 175, 2021. ISSN 01641212. doi: 10.1016/j.jss.2021.110911.

[143] Deeksha Arya, Wenting Wang, Jin L.C. Guo, and Jinghui Cheng. Analysis and detection of information types of open-source software issue discussions. In *Proceedings of the IEEE/ACM 41st International Conference on Software Engineering*, pages 454–464, 2019. ISBN 9781728108698. doi: 10.1109/ICSE.2019.00058.

[144] Vijaya Kumar Eluri, Shahram Sarkani, and Thomas A. Mazzuchi. Open-source software survivability prediction using multi layer perceptron. *EPiC Series in Computing*, 64:148–157, 2019. ISSN 23987340. doi: 10.29007/cmc6.

[145] Asher Trockman, Rijnard Van Tonder, and Bogdan Vasilescu. Striking gold in software repositories? An econometric study of cryptocurrencies on GitHub. In *Proceedings of the ACM/IEEE 16th International Working Conference on Mining Software Repositories*, volume 2019-May, pages 181–185, 2019. ISBN 9781728134123. doi: 10.1109/MSR.2019.00036.

[146] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. Curating github for engineered software projects. *Empirical Software Engineering*, 22(6):3219–3253, 2017. ISSN 15737616. doi: 10.1007/s10664-017-9512-6.

[147] Riivo Kikas, Marlon Dumas, and Dietmar Pfahl. Using dynamic and contextual features to predict issue lifetime in GitHub projects. In *Proceedings of the ACM/IEEE 13th Working Conference on Mining Software Repositories*, pages 291–302, 2016. ISBN 9781450341868. doi: 10.1145/2901739.2901751.

[148] Yang Song and Oscar Chaparro. BEE: A tool for structuring and analyzing bug reports. In *Proceedings of the 28th ACM Joint Meeting European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1551–1555, 2020. ISBN 9781450370431. doi: 10.1145/3368089.3417928.

[149] Fabian Pedregosa, Gael Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Matthieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[150] Armin Shmilovici. Support vector machines. In *Data Mining and Knowledge Discovery Handbook*, pages 231–247. Springer US, 2010. ISBN 978-0-387-09823-4. doi: 10.1007/978-0-387-09823-4_12.

[151] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

[152] Wasiur Rhmann, Babita Pandey, Gufran Ansari, and D. K. Pandey. Software fault prediction based on change metrics using hybrid algorithms: An empirical study. *Journal of King Saud University - Computer and Information Sciences*, 32 (4):419–424, may 2020. ISSN 22131248. doi: 10.1016/j.jksuci.2019.03.006.

[153] David G Kleinbaum, Mitchel Klein, and Erica Rihl Pryor. *Logistic Regression: A Self-Learning Text*. Springer, 2002.

[154] Jingxiu Yao and Martin Shepperd. The impact of using biased performance metrics on software defect prediction research. *Information and Software Technology*, 139:106664, 2021. ISSN 09505849. doi: 10.1016/j.infsof.2021.106664.

[155] Jingxiu Yao and Martin Shepperd. Assessing software defection prediction performance: Why using the Matthews correlation coefficient matters. In *Proceedings of the Evaluation and Assessment in Software Engineering*, pages 120–129, 2020. ISBN 9781450377317. doi: 10.1145/3383219.3383232.

[156] Jacob Cohen. *Statistical Power Analysis for the Behavioral Sciences*. Routledge, 2013.

[157] Huiwei Zhou, Shixian Ning, Yunlong Yang, Zhuang Liu, and Junli Xu. Chinese hedge scope detection based on phrase semantic representation. In *Proceedings of the 2017 International Conference on Asian Language Processing*, pages 285–288, 2017. ISBN 978-1-5386-1982-7. doi: 10.1109/IALP.2017.8300599.

[158] Samuel K. Akpatsa, Xiaoyu Li, and Hang Lei. A survey and future perspectives of hybrid deep learning models for text classification. In *Proceedings of the International Conference on Artificial Intelligence and Security*, pages 358–369, 2021. ISBN 978-3-030-78609-0.

[159] Cristian Padurariu and Mihaela Elena Breaban. Dealing with data imbalance in text classification. In *Procedia Computer Science*, volume 159, pages 736–745, 2019. doi: 10.1016/j.procs.2019.09.229.

[160] Roweida Mohammed, Jumanah Rawashdeh, and Malak Abdullah. Machine learning with oversampling and undersampling techniques: Overview study and experimental results. In *Proceedings of the 11th International Conference on Information and Communication Systems*, pages 243–248, 2020. ISBN 9781728162270. doi: 10.1109/ICICS49469.2020.239556.

[161] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. doi: 10.1613/jair.953.

[162] Stefanie Beyer, Christian Macho, Massimiliano Di Penta, and Martin Pinzger. What kind of questions do developers ask on Stack Overflow? A comparison of automated approaches to classify posts into question categories. *Empirical Software Engineering*, 25(3):2258–2301, 2020. ISSN 15737616. doi: 10.1007/s10664-019-09758-x.

[163] Gemma Catolino, Fabio Palomba, Andy Zaidman, and Filomena Ferrucci. Not all bugs are the same: Understanding, characterizing, and classifying bug types.

Journal of Systems and Software, 152:165–181, 2019. ISSN 01641212. doi: 10.1016/j.jss.2019.03.002.

[164] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. *Journal of Machine Learning Research*, 18:559–563, 2017.

[165] Eeshita Biswas, K. Vijay-Shanker, and Lori Pollock. Exploring word embedding techniques to improve sentiment analysis of software engineering texts. In *Proceedings of the ACM/IEEE 16th International Working Conference on Mining Software Repositories*, pages 68–78, 2019. ISBN 9781728134123. doi: 10.1109/MSR.2019.00020.

[166] Edna Dias Canedo, Rodrigo Bonifácio, Márcio Vinicius Okimoto, Alexander Serebrenik, Gustavo Pinto, and Eduardo Monteiro. Work practices and perceptions from women core developers in OSS communities. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2020. ISBN 9781450375801. doi: 10.1145/3382494.3410682.

[167] Emi Ishita, Satoshi Fukuda, Toru Oga, Yoichi Tomiura, Douglas W. Oard, and Kenneth R. Fleischmann. Cost-effective learning for classifying human values. In *iConference Poster*, 2020.

[168] Javier Luis Canovas Izquierdo, Valerio Cosentino, Belen Rolandi, Alexandre Bergel, and Jordi Cabot. GiLA: GitHub label analyzer. In *IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015 - Proceedings*, pages 479–483, 2015. ISBN 9781479984695. doi: 10.1109/SANER.2015.7081860.

[169] Siw Elisabeth Hove and Bente Anda. Experiences from conducting semi-structured interviews in empirical software engineering research. In *Proceedings of the 11th IEEE International Software Metrics Symposium*, volume 2005, pages 203–212, 2005. ISBN 0769523714. doi: 10.1109/METRICS.2005.24.

[170] João M. Fernandes and Ricardo J. Machado. Requirements elicitation. In *Requirements Engineering in Projects*, pages 85–117. Springer, Cham, 2016. doi: 10.1007/978-3-319-18597-2_5.

[171] Marko Gasparic, Andrea Janes, Francesco Ricci, Gail C. Murphy, and Tural Gurbanov. A graphical user interface for presenting integrated development environment command recommendations: Design, evaluation, and implementation.

Information and Software Technology, 92:236–255, dec 2017. ISSN 09505849. doi: 10.1016/j.infsof.2017.08.006.

[172] Miguel Pestana, Rúben Pereira, and Sérgio Moro. A productivity dashboard for hospitals: An empirical study. In *Information Systems: Research, Development, Applications, Education*, pages 184–199, 2018. ISBN 9783030000592. doi: 10.1007/978-3-030-00060-8_14.

[173] Miguel Pestana, Ruben Pereira, and Sérgio Moro. Improving health care management in hospitals through a productivity dashboard. *Journal of Medical Systems*, 44, 2020. ISSN 1573689X. doi: 10.1007/s10916-020-01546-1.

[174] Enrico Bunde. AI-Assisted and explainable hate speech detection for social media moderators - A design science approach. In *Proceedings of the 54th Hawaii International Conference on System Sciences*, 2021. ISBN 9780998133140. doi: 10.24251/hicss.2021.154.

[175] Vladimir Ivanov, Daria Larionova, Dragos Strugar, and Giancarlo Succi. Design of a dashboard of software metrics for adaptable, energy efficient applications. In *Proceedings of the 25th International DMS Conference on Visualization and Visual Languages*, pages 75–82, 2019. ISBN 1891706497. doi: 10.18293/jvlc2019-n2-009.

[176] GitLab. Issues. URL https://docs.gitlab.com/ee/user/project/issues/. Accessed: April 18, 2021.

[177] Antonio Gonz, Roberto Ther, and Francisco J Garc. Maleku: An evolutionary visual software analytics tool for providing insights into software evolution. In *Proceedings of the 27th IEEE International Conference on Software Maintenance*, pages 594–597, 2011. ISBN 9781457706646.

[178] Ben K Beitin. Interview and sampling. In *The SAGE Handbook of Interview Research: The Complexity of the Craft*, pages 243–254. Sage Thousand Oaks, CA, 2012.

[179] Zakaria Ournani, Romain Rouvoy, Pierre Rust, and Joel Penhoat. On reducing the energy consumption of software: From hurdles to requirements. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–12, 2020. ISBN 9781450375801. doi: 10.1145/3382494.3410678.

[180] Virginia Braun and Victoria Clarke. Thematic analysis. In *APA Handbook of Research Methods in Psychology: Vol.2. Research Designs*, pages 57–71. 2012. doi: 10.1037/13620-004.

[181] Kristín Fjóla Tómasdóttir, Maurício Aniche, and Arie Van Deursen. The adoption of JavaScript linters in practice: A case study on ESLint. *IEEE Transactions on Software Engineering*, 46(8):863–891, 2020. doi: 10.1109/TSE.2018.2871058.

[182] Theodore Hammer, Lenore Huffman, H. Rosenberg Linda, William Wilson, and Lawrence E. Hyatt. Doing requirements right the first time!, 1998.

[183] Perdita Stevens and Rob J. Pooley. *Using UML: Software Engineering with Objects and Components*. Pearson Education, 2006.

[184] Sari Kujala and Kaisa Väänänen-Vainio-Mattila. Value of information systems and products: Understanding the users' perspective and values. *JITTA: Journal of Information Technology Theory and Application*, 9(4):23, 2009. ISSN 1532-4516.

[185] Mark A. Harris, Robert Brookshire, and Amita Goyal Chin. Identifying factors influencing consumers' intent to install mobile applications. *International Journal of Information Management*, 36(3):441–450, 2016. ISSN 02684012. doi: 10.1016/j.ijinfomgt.2016.02.004.

[186] EU. What is GDPR, the EU's new data protection law? URL https://gdpr.eu/what-is-gdpr/. Accessed: October 1, 2021.

[187] Egon G. Guba. Criteria for assessing the trustworthiness of naturalistic inquiries. *Educational Communication & Technology Journal*, 29(2):75–91, 1981. ISSN 01485806. doi: 10.1007/BF02766777.

[188] Klaas Jan Stol, Paris Avgeriou, Muhammad Ali Babar, Yan Lucas, and Brian Fitzgerald. Key factors for adopting inner source. *ACM Transactions on Software Engineering and Methodology*, 23(2), 2014. ISSN 15577392. doi: 10.1145/2533685.

[189] Daniela S. Cruzes and Tore Dybå. Recommended steps for thematic synthesis in software engineering. In *Proceedings of the International Symposium on Empirical Software Engineering and Measurement*, pages 275–284. IEEE, 2011. ISBN 9780769546049. doi: 10.1109/esem.2011.36.

[190] Waqar Hussain, Mojtaba Shahin, Rashina Hoda, Jon Whittle, Harsha Perera, Arif Nurwidyantoro, Rifat Ara Shams, and Gillian Oliver. How can human values be

addressed in agile methods? A case study on SAFe. *IEEE Transactions on Software Engineering*, Early access, 2022. doi: 10.1109/TSE.2022.3140230.

[191] Jailton Coelho, Marco Tulio Valente, Luciano Milen, and Luciana L Silva. Is this GitHub project maintained? Measuring the level of maintenance activity of open-source projects. *Information and Software Technology*, 122:106274, 2020. doi: 10.1016/j.infsof.2020.106274.

[192] Fiorella Zampetti, Luca Ponzanelli, Gabriele Bavota, Andrea Mocci, Massimiliano Di Penta, and Michele Lanza. How developers document pull requests with external references. In *Proceedings of the IEEE/ACM 25th International Conference on Program Comprehension (ICPC)*, pages 23–33, 2017. doi: 10.1109/ICPC.2017.30.

[193] Nasif Imtiaz, Justin Middleton, Joymallya Chakraborty, Neill Robson, Gina Bai, and Emerson Murphy-Hill. Investigating the effects of gender bias on github. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*, pages 700–711. IEEE, 2019.

[194] Bogdan Vasilescu, Alexander Serebrenik, and Vladimir Filkov. A data set for social diversity studies of github teams. In *2015 IEEE/ACM 12th working conference on mining software repositories*, pages 514–517. IEEE, 2015.

[195] Marco Ortu, Giuseppe Destefanis, Bram Adams, Alessandro Murgia, Michele Marchesi, and Roberto Tonelli. The jira repository dataset: Understanding social aspects of software development. In *Proceedings of the 11th international conference on predictive models and data analytics in software engineering*, pages 1–4, 2015.

[196] Marco Ortu, Giuseppe Destefanis, Mohamad Kassab, Steve Counsell, Michele Marchesi, and Roberto Tonelli. Would you mind fixing this issue? In *International conference on Agile software development*, pages 129–140. Springer, 2015.

[197] Marco Ortu, Giuseppe Destefanis, Steve Counsell, Stephen Swift, Roberto Tonelli, and Michele Marchesi. How diverse is your team? investigating gender and nationality diversity in github teams. *Journal of Software Engineering Research and Development*, 5(1):1–18, 2017.