Prioritisation of requests, bugs and enhancements pertaining to apps for remedial actions

Towards solving the problem of which app concerns to address initially for app developers

Saurabh
Malgaonkar

A thesis submitted for the degree of

Doctor of Philosophy

at the University of Otago, Dunedin,

New Zealand

11th September 2020

Abstract

Background: Useful app reviews contain information related to the bugs reported by the app's end-users along with the requests or enhancements (i.e., suggestions for improvement) pertaining to the app. App developers expend exhaustive manual efforts towards the identification of numerous useful reviews from a vast pool of reviews and converting such useful reviews into actionable knowledge by means of prioritisation. By doing so, app developers can resolve the critical bugs and simultaneously address the prominent requests or enhancements in short intervals of apps' maintenance and evolution cycles.

Research Problem: That said, the manual efforts towards the identification and prioritisation of useful reviews have limitations. The most common limitations are: high cognitive load required to perform manual analysis, lack of scalability associated with limited human resources to process voluminous reviews, extensive time requirements and error-proneness related to the manual efforts. While prior work from the app domain have proposed prioritisation approaches to convert reviews pertaining to an app into actionable knowledge, these studies have limitations and lack benchmarking of the prioritisation performance. Thus, the problem to prioritise numerous useful reviews still persists.

Research Method: In this study, initially, we conducted a systematic mapping study of the requirements prioritisation domain to explore the knowledge on prioritisation that exists and seek inspiration from the eminent empirical studies to solve the problem related to the prioritisation of numerous useful reviews. Findings of the systematic mapping study inspired us to develop automated approaches for filtering useful reviews, and then to facilitate their subsequent prioritisation. To filter useful reviews, this work developed six variants of the Multinomial Naïve Bayes method. Next, to prioritise the order in which useful reviews should be addressed, we proposed a group-based prioritisation method which initially classified the useful reviews into specific groups using an automatically generated taxonomy, and later prioritised these reviews using a multi-criteria heuristic function. Subsequently, we developed an individual prioritisation method that directly prioritised the useful reviews after filtering using the same multi-criteria heuristic function.

Results: Some of the findings of the conducted systematic mapping study not only provided the necessary inspiration towards the development of automated filtering and prioritisation approaches but also revealed crucial dimensions such as accuracy and time that could be utilised to benchmark the performance of a prioritisation method. With regards to the proposed automated filtering approach, we observed that the performance of the Multinomial Naïve Bayes variants varied based on their algorithmic structure and the nature of labelled reviews (i.e., balanced or imbalanced) that were made

available for training purposes. The outcome related to the automated taxonomy generation approach for classifying useful review into specific groups showed a substantial match with the manual taxonomy generated from domain knowledge. Finally, we validated the performance of the group-based prioritisation and individual prioritisation methods, where we found that the performance of the individual prioritisation method was superior to that of the group-based prioritisation method when outcomes were assessed for the accuracy and time dimensions. In addition, we performed a full-scale evaluation of the individual prioritisation method which showed promising results.

Conclusion: Given the outcomes, it is anticipated that our individual prioritisation method could assist app developers in filtering and prioritising numerous useful reviews to support app maintenance and evolution cycles. Beyond app reviews, the utility of our proposed prioritisation solution can be evaluated on software repositories tracking bugs and requests such as Jira, GitHub and so on.

Acknowledgement

I would like to convey my wholehearted gratitude to the people who supported me throughout the PhD course.

Firstly, I would like to convey my sincere thanks to my supervisors Sherlock A. Licorish and Bastin 'Tony' Roy Savarimuthu for their continuous encouragement, guidance and support during the PhD course. The mentorship offered by my supervisors was fundamental to shape the research conducted during the course of my PhD. With the supervisors' dedication and regular feedback, I was able to enhance my knowledge and improvise to complete the undertaken research work and thesis. I appreciate their beneficial advices and efforts, especially on making the research work robust which is essential to complete the PhD course.

I am very grateful to my parents (Ramakant Malgaonkar and Rhujuta Malgaonkar) and sister (Ketki Malgaonkar), who have been there for me always and motivated me during my PhD. I am also thankful to them for offering the essential financial and moral support to complete the PhD course.

I would also like to thank the members and participants of the Postgraduate Symposium organised by the Department of Information Science at University of Otago for their valuable feedback.

I would also like to thank my colleagues at University of Otago who made the PhD experience interesting and enjoyable and other staff members (Holger Regenbrecht, Gail Mercer, Heather Copper, Karen Bosworth, Brendon Sly, Steven Doig and Dean Huakau) for helping me out with several administrative and ICT support aspects.

I am also grateful for the financial support (hardship funds) I received from University of Otago and the grant funding from Russell Education Trust. I am thankful to the Department of Information Science at University of Otago for providing me with job opportunities such as teaching and research assistantships.

Special thanks to Daniel Alencar da Costa, Caitlin Amelia Owen and Abira Sengupta for offering their genuine moral support by being present at the final oral examination.

Finally, and importantly, I would like to thank God for helping me throughout this PhD journey and life in many mysterious ways.

List of Acronyms

- ACM - Association for Computing Machinery
- AHP - Analytical Hierarchical Process
- BERT - Bidirectional Encoder Representations from Transformers
- BoW - Bag of Words
- BST - Binary Search Tree
- CNN - Convolution Neural Network
- COALS - Correlated Occurrence Analogue to Lexical Semantics
- CV - Cumulative Voting
- CVA - Cost Value Approach
- GPS - Global Positioning System
- HAL - Hyperspace Analogue to Language
- HCV - Hierarchical Cumulative Voting
- IEEE - Institute of Electrical and Electronics Engineers
- KNN - K Nearest Neighbours
- LDA - Latent Dirichlet Allocation
- LSA - Latent Semantic Analysis
- MARA - Mobile App Repository Analyser
- MoSCoW - Must, Should, Could and Won't
- NA - Numerical Assignment
- NRP - Next Release Problem
- PG - Planning Game
- POS - Parts of Speech
- QFD - Quality Function Deployment
- SVD - Singular Value Decomposition
- SVM - Support Vector Machines
- TF-IDF - Term Frequency-Inverse Document Frequency
- UML - Unified Modelling Language
- VADER - Valence Aware Dictionary and sEntiment Reasoner

Publications

Journal

- Saurabh Malgaonkar, Sherlock A. Licorish, Bastin Tony Roy Savarimuthu, "A Survey and Critical Evaluation of Requirements Prioritization", IET Software. https://digital-library.theiet.org/content/journals/10.1049/iet-sen.2019.0215

Conference

- Saurabh Malgaonkar, Sherlock A. Licorish, Bastin Tony Roy Savarimuthu (2020) Towards Automated Taxonomy Generation for Grouping App Reviews: A Preliminary Empirical Study. In: Shepperd M., Brito e Abreu F., Rodrigues da Silva A., Pérez-Castillo R. (eds) Quality of Information and Communications Technology. QUATIC 2020. Communications in Computer and Information Science, Vol. 1266. Springer, Cham. http://doi-org-443.webvpn.fjmu.edu.cn/10.1007/978-3-030-58793-2

Paper under consideration

- Extended version of published paper "Towards Automated Taxonomy Generation for Grouping App Reviews: A Preliminary Empirical Study" has been submitted to the Software Quality Journal.
- Paper titled "Evaluating Possible Variants of Naïve Bayes Towards Filtering Useful App Reviews" which is based on the filtering of useful reviews phase documented in this thesis has been submitted to the Software: Evolution and Process Journal.
- Paper titled "Prioritizing User Concerns in App Reviews – A Study of Requests for New Features, Enhancements and Bug Fixes" which is based on the prioritisation of useful reviews phase documented in this thesis has been submitted to Information and Software Technology Journal.

Table of Contents

List of Figures

List of Tables

1 Introduction

An app is a software product that is the outcome of software engineering and undergoes several re-engineering phases for its maintenance and evolution (Goul et al., 2012; Maalej et al., 2016a). The app market has become a multibillion dollar industry[1] with millions of apps hosted on commonly known app distribution platforms such as Google Play Store[2] or Apple App Store[3]. This suggests that the modern society is strongly reliant on apps to fulfil their application specific requirements (Pagano & Maalej, 2013). The prospective end-users of these apps download and install the apps on their app compatible devices such as smartphones, tablets, notebooks, and so on. As the app distribution platforms facilitate the provision of end-users' feedback regarding their experience with an app, usually the majority of the end-users log their feedback in the form of reviews (Pagano & Maalej, 2013). Apart from a star rating that is expressed on a scale of 1-5, or a general compliment or criticism, the reviews usually indicate the request for features, bugs present in the app, or enhancements (i.e., suggestions for improvements) (Maalej et al., 2016a; Pagano & Maalej, 2013). Figure 1 illustrates an example depicting a star rating of 3 and a review indicating a bug related to My Tracks[4] app made available on Google Play Store. My Tracks app allows its end-users to set and track possible travelling routes. The app also allows its end-users to check statistics of their travelling activities with regards to the distance travelled, speed attained, ground elevation levels, exercise routines, and so on.

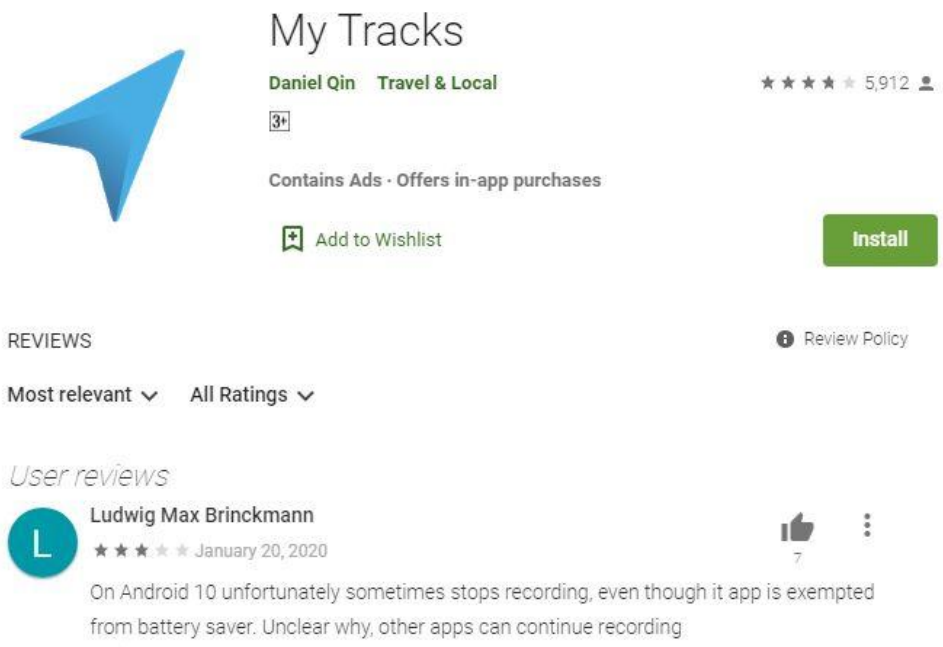


Figure 1. Example of end-user rating and review for My Tracks app

[1] https://www.businessofapps.com/data/app-revenues/
[2] https://play.google.com/store
[3] https://www.apple.com/ios/app-store/
[4] https://play.google.com/store/apps/details?id=com.zihua.android.mytracks

Table 1.1 shows review examples that reflect a request, a bug and an enhancement pertaining to the My Tracks app. In the request example, a feature is requested that would allow the end-user to save the travelled routes for comparison purpose and find the shortest route among them. The bug example indicates a flaw in the tracking functionality of the app that generates inaccurate distance statistics. On the other hand, an end-user suggests an enhancement that would probably motivate end-users for regular exercising.

Table 1.1 Examples indicating a request, bug or enhancement review

Review Type	Example
Request	I go walk from one place to one destination, From different routes, I want to save and COMPARE and find shortest route….please add this option.
Bug	Tracking is inaccurate...known 3 mile walk tracked at over 5 miles. 2 out of 4 tracks were inaccurate. Frustrating... can't rely on the data.
Enhancement	I would suggest that a reward system for regular exercising would be an awesome addon to this app..

From the above-mentioned examples, it is evident that the reviews indicate aspects related to the app that are significant to the end-users. Thus, addressing reviews indicating key requests, bugs or enhancements logged by the end-users is of foremost importance to app developers as it allows the app developers to launch a new version of the app reflecting the addressed requests, bugs or enhancements in the form of app updates (Licorish et al., 2017). Simultaneously, this supports the app maintenance and evolution cycles and improves the quality of the app (Maalej et al., 2016a; Pagano & Maalej, 2013). Furthermore, this contributes towards the app enterprise's monetary gains and potentially increases the popularity of the app in the competitive app market (Goul et al., 2012; Maalej et al., 2016b). Throughout this study, the reviews that indicate requests, bugs or enhancements are termed as 'useful reviews' as these reviews indicate the useful information for app developers which is necessary towards improving the quality and market performance of the app (Panichella et al., 2015; Roma & Ragaglia, 2016).

1.1 Problem Statement

Several studies on app reviews mining have been conducted such as understanding end-users' sentiments regarding app usage, and identification of keywords of interest from reviews (Fu et al., 2013; Iacob & Harrison, 2013). As such studies only attempt to draw out application specific meaningful insights from the reviews, and app developers are constantly on the lookout for reliable automated approaches that convert the innumerable useful reviews into actionable knowledge as they endlessly face the dilemma of '*Which useful reviews to address initially during the short intervals of app maintenance and evolution cycle?*' (Maalej et al., 2016a; Pagano & Maalej, 2013). App developers usually prefer automated approaches over manual ones to lessen errors, reduce the overall processing time, avert the need for high levels of cognitive load, and to establish scalability (Pagano & Maalej,

2013). Thus, classification is a popular approach utilised towards the resolution of the dilemma (Ciurumelea et al., 2018; Maalej et al. 2016a). However, the outcome of classification tends to provide only a generalised view of the actionable knowledge, and hence, is suited only when the reviews are in manageable numbers (Yang & Liang, 2015). In case of numerous reviews, classification fails to answer the question '*Which are the important reviews to address and in what order they need to be addressed?*', and in most scenarios does not resolve the redundant information issue occurring as a result of duplicate instances of similar reviews being classified into multiple groups (or classes) of interests (Aly, 2005; Ciurumelea et al., 2018; Maalej et al., 2016a). On the contrary, prioritisation approaches have shown promise towards the conversion of numerous reviews into actionable knowledge, as they perform ranking of certain aspects (e.g., app features) mentioned in the reviews, or reviews themselves based on their importance or severity through the use of particular method(s) (Chen et al., 2014; Licorish et al., 2017). However, the methods that are used for prioritisation are far from perfect, and the problem to prioritise numerous reviews still persists which is the primary research gap that this study aims to address (Licorish et al., 2017).

1.2 Research Aim

The problem to prioritise numerous useful reviews is similar to the NRP (Next Release Problem) encountered by software developers (Bagnall et al., 2001; Sureka, 2014). The NRP states that the software developers are unable to decide on which software requirements to address for the next release version of the software during the requirements engineering phase. For instance, the order in which the end-users' requirements need to be addressed to release the next version of the software (Sureka, 2014). Therefore, the overall aim of this research is to engineer a prioritisation method to generate actionable knowledge for app developers through the prioritisation of numerous useful reviews and simultaneously provide understandings for the software engineering community in terms of how a method can be developed to prioritise requests, bugs or enhancements pertaining to a software product logged by users. Table 1.2 provides an example of actionable knowledge generated in the form of prioritisation that this study aims to provide for the app developers.

Table 1.2 Example of prioritised useful reviews

Review	Priority
I go walk from one place to one destination, from different routes, I want to save and COMPARE and find shortest route…please add this option	Medium
Tracking is inaccurate...known 3 mile walk tracked at over 5 miles. 2 out of 4 tracks were inaccurate. Frustrating... can't rely on the data.	High
I would suggest that a reward system for regular exercising would be an awesome addon to this app..	Low

In Table 1.2 the Priority column indicates the order in which the app developers can address the concerns related to the app conveyed through useful reviews. Based on the computed priorities, the app developers can initially fix the bug related to the tracking functionality of the app that displays inaccurate distance, and then later can add the option in the app that allows the end-user to save and compare the travelled routes and find the shortest route among them. Finally, the app developers can work towards the reward system. We believe that the app developers with the assistance of such actionable knowledge can decide on which useful reviews to address first during the limited intervals of app maintenance and evolution cycle given the constraints (e.g., budget, technical, time, resource, feasibility, and so on) that are imposed on the app developers.

1.3 Research Outline

To our best knowledge there are limited prior studies in the app domain that deal with prioritisation. As we could not inherit essential guidelines towards developing an automated prioritisation method for useful reviews from the limited prior studies in the app domain, we reviewed studies from the requirements prioritisation domain in different disciplines that address the prioritisation problem. Next, based on our findings from the conducted assessments we accordingly framed the relevant initial research question to drive our research. The research question lead to the initiation of the first phase of the study. In the first phase we conducted a comprehensive systematic mapping study of the requirements prioritisation domain to explore and critique prioritisation studies belonging to various disciplines such as software engineering, product manufacturing, education, finance, real estate and law to seek inspiration and derive essential guidelines towards the development and empirical evaluation of our proposed prioritisation method. The outcome of the systematic mapping study lead to the initiation of the next subsequent three phases guided by appropriate research questions. In the second phase we carried out a pilot study which deals with the filtering of useful reviews and benchmarked the performance of six different variants of the same filtering method. The motive behind the filtering approach was to avoid non-useful reviews that did not convey significant information necessary towards the remedial actions for the particular app. Phase three presents a pilot study that experimented with a classification approach which comprised of automatically generating a taxonomy to classify useful reviews into groups of interest. Phase four presents a pilot study in which we experimented with a group-based prioritisation method and an individual prioritisation method. The group-based prioritisation method utilised the outcome generated by our proposed classification approach (i.e., classified useful reviews) from phase 3 to compute priorities of the useful reviews and their associated groups. However, the method did not produce promising results in the internal validation stage because of which we had to develop the individual prioritisation method. Thus, we developed the individual prioritisation method that outperformed the group-based prioritisation method based on the generated results. Next, we performed a full-scale evaluation of the individual prioritisation method. As the results

generated by the individual prioritisation method showed promise in the interval validation stage, we subjected the same subset of results for external validation and found similar reassuring outcomes.

1.4 Research Questions

Based on the research outline mentioned above, Figure 2 illustrates the research questions (RQs) addressed in this thesis and the relationship shared among the overarching RQs (Dillon, 1984; Potts, 1993). The brief elaboration of the RQs portrayed in Figure 2 is as follows; in this study, RQ1 (What is the state-of-the art of requirements prioritisation?) is concerned with obtaining a comprehensive understanding of the requirements prioritisation domain and we answered RQ1 through the means of a systematic mapping study (Petersen et al., 2008). Because of the systematic mapping study, we were able to perform a critical evaluation of studies that have provided requirements prioritisation methods across all disciplines. We found out that the current methods from the software engineering discipline do not consider the strengths of the requirements prioritisation methods available from other disciplines (e.g., product manufacturing) or vice-versa, a gap that opens new research opportunities. Among the other findings, we observed that while many prioritisation methods are targeted, often researchers have proposed prioritisation methods that were not evaluated. Most prioritisation methods were only validated as being operational, and the attributes studied had limited effects on performance outcomes. In addition, performance trade-offs are to be expected of such methods, depending on their performance targets. Overall, the evidence obtained from the systematic mapping study suggests that emerging methods may address the requirements prioritisation challenge if they are inspired by hybrid prioritisation methods. The explicit details of our mapping study with regards to RQ1.1 to RQ1.6 are presented from Chapter 3 onwards. That said, we found one empirical study in phase 1 that proposed a prioritisation method which prioritised numerous requirements assuring its scalability (Peng et al., 2012). Even though the prioritisation method performed the prioritisation of requirements at a group level (i.e., generating only the priorities of the pre-defined groups (or classes) in which the requirements were classified into) and was dependent on the availability of the domain knowledge along with the priority preferences of stakeholders to generate the priorities of the groups, the model followed by the method for prioritisation provided the essential inspiration and shaped RQ2, RQ3 and RQ4 respectively.

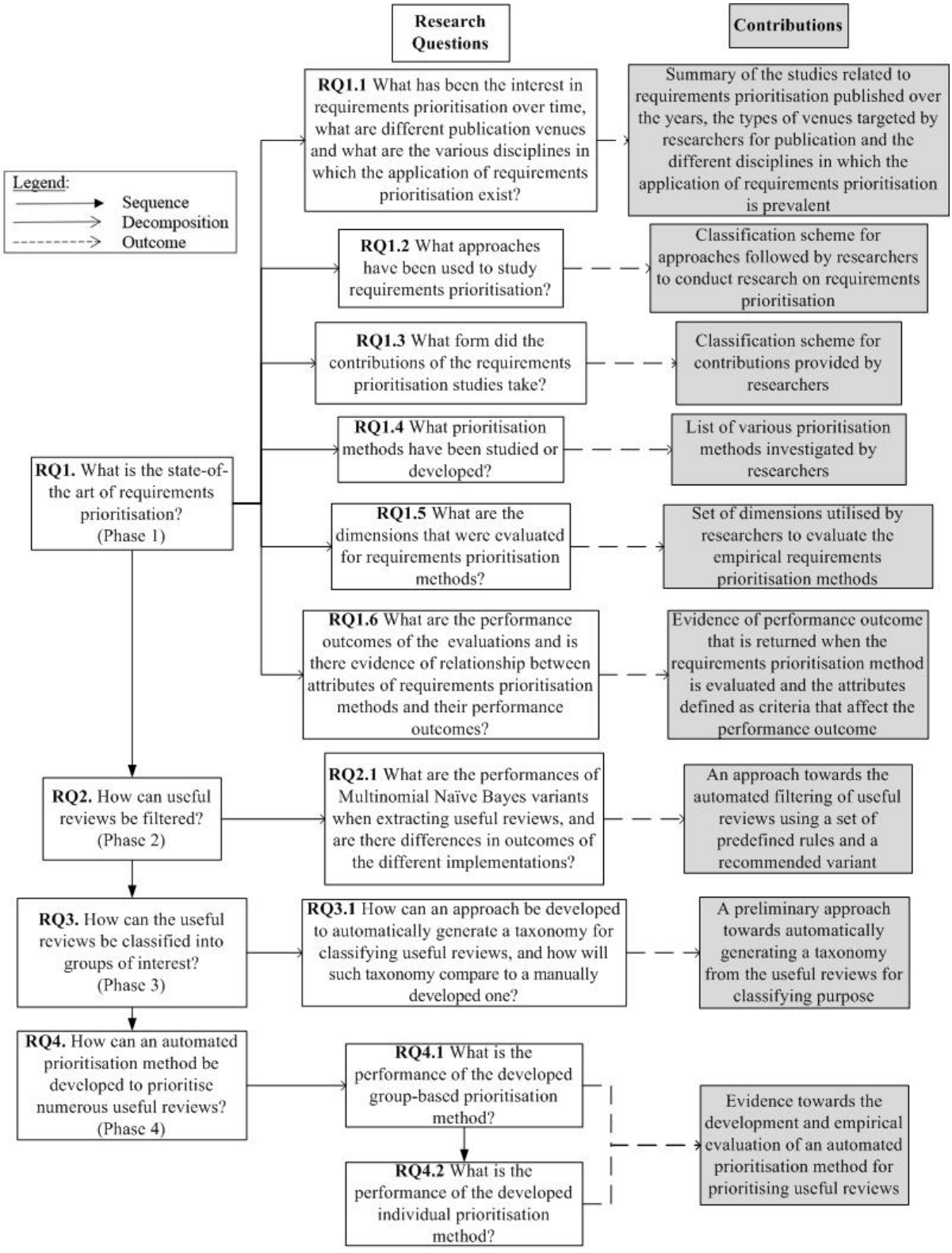


Figure 2. Research questions

To answer RQ2 (How can useful reviews be filtered?), we explored the information retrieval approaches that provided insights towards the filtering (or extraction) of useful reviews from a vast pool of reviews. Our investigation led to the discovery and empirical evaluation of six Multinomial Naïve Bayes variants specialised in filtering of useful reviews based on predefined rules. As an outcome of this we could

conclude that the selection of a specific Multinomial Naïve Bayes variant for useful reviews filtering is dependent on the nature of the information retrieval application (i.e., number of reviews and the ratio of useful to non-useful reviews that are made available for learning purpose) (Nigam et al., 2003). We cover all the details revolving around RQ2 from Chapters 3 onwards. To answer RQ3 (How can the useful reviews be classified into groups of interest?), we went beyond the utilisation of a manual taxonomy to classify useful reviews into groups of interest by developing a preliminary approach that automatically generates a taxonomy independent of the availability of domain knowledge. Finally, to answer RQ4 (How can an automated prioritisation method be developed to prioritise numerous useful reviews?) we took inspiration from studies from domains such as feature engineering, information theory, information retrieval and marketing to identify prominent methods specialised in prioritisation and develop a hybrid automated prioritisation methods (i.e., group-based prioritisation method and individual prioritisation method) to prioritise useful reviews (Chea et al., 2009; Fang & Zhan, 2015; Filcek et al., 2017; Htay & Lynn, 2013; Ko et al., 2000; Sundaram et al., 2005). We benchmarked the performance of the developed prioritisation methods using the two dimensions (accuracy and time) identified in phase 1. Similar to RQ1 and RQ2 all the evidence related to the answering of RQ3 and RQ4 is explicitly elaborated from Chapter 3 onwards.

1.5 Thesis Structure

The further chapters of this thesis are organised as follows. Chapter 2 (Background) presents a review highlighting a brief assessment of the existing prioritisation studies that provide the foundation related to the problem investigated in this thesis, and describes the basics related to the actionable knowledge generated in terms of prioritisation. In Chapter 3, we present the undertaken systematic mapping study on requirements prioritisation corresponding to RQ1 and its associated decomposed RQs (i.e., RQs 1.1 to RQ1.6). Chapter 4 presents the phase which deals with filtering of useful reviews (i.e., phase 2) corresponding to RQ2 and its associated decomposed RQ2.1. Chapters 5 and 6 present the phases that deal with classification of useful reviews (i.e., phase 3) and prioritisation of useful reviews (i.e., phase 4), respectively. RQ3 and RQ4 and their associated decomposed RQs (i.e., RQ3.1, RQ4.1 to RQ4.2) are covered in Chapters 5 and 6 accordingly. The results pertaining to each decomposed RQs are covered within the relevant chapters, and Chapter 6 mentions a link to the web tool that shows the operational demonstration of phases 2, 3 and 4 of the research project. Furthermore, the discussions and implications related to the RQs are documented within the relevant chapters along with the threats to validity associated with the respective phase. Finally, we provide the summary of the outcomes corresponding to the RQs, contributions and potential future work in Chapter 7 (Conclusions).

In the next chapter, we present the background related to the undertaken study.

2 Background

A successful app thriving in the competitive market constantly demands software maintenance and evolution cycles as the usefulness of the app to its end-users depends on the features the app provides and the quality it aims to assure by addressing the end-users' requests, bugs or enhancements (Bennett & Rajlich, 2000; Maalej et al., 2016a; Pagano & Maalej, 2013). Thus, the app's maintenance and evolution cycles are initiated after the app is released in the market, and the cycles typically involve the addressing of end-users' requests, bugs or enhancements. This leads to the relevant transformations in the app's software architecture which causes a new release version of the app in the form of an update (Bennett & Rajlich, 2000; Maalej et al., 2016a; Pagano & Maalej, 2013). Similar to the traditional software repositories such as logs comprising of bug reports or requests that are often seen beneficial for software maintenance, app developers primarily rely on the app's reviews as most of these are seen as trusted source of insights and provide the necessary information to drive the app's maintenance and evolution cycles (Goul et al., 2012; Iacob et al., 2014; Tian et al., 2004). Furthermore, as app developers are aware that end-users' satisfaction is central to the app gaining positive popularity to guarantee prolong usage of the app, app developers find it necessary to address reviews reflecting end-users' requests, bugs or enhancements to provide substantial contribution towards the app's market value (Fabio et al., 2015; Roma & Ragaglia, 2016). As the stream of reviews are logged by the end-users at regular intervals (i.e., after app or update release), the app developers are constantly engaged in the app's post-delivery activities to identify the necessary information (i.e., useful reviews) from the reviews and later convert the information into actionable knowledge to address the end-users' requests, bugs or enhancements pertaining to the app and expedite the necessary app updates (Pagano & Maalej, 2013).

As the reviews logged by the end-users usually tend to be voluminous, app developers face limitations when utilising manual efforts towards the identification and conversion of the useful reviews into actionable knowledge (Pagano & Maalej, 2013). Some of the serious limitations point towards the demand for high cognitive loads for manual analysis, error-proneness, time constraints and lack of scalability of the manual efforts due to limited human resources (Maalej et al., 2016a). Therefore, the app developers are on the lookout for automated approaches that allow them to accomplish the same objective by incurring less overheads but at the same time providing a substantial level of precise information because of which they can work towards the essential updates required for the app (Ciurumelea et al., 2017; Maalej et al., 2016a).

We review the studies from the app domain in section 2.1 that utilise different research approaches to examine app reviews to gain meaningful insights, and those that attempt to transfer such insights into actionable knowledge, along with the limitations of these studies. This is followed by a brief

introduction of the NRP in section 2.2 and studies from the requirements prioritisation domain in section 2.3, which ultimately lead towards the generation of RQ1.

2.1 App Domain Studies

Many studies have made attempts towards obtaining meaningful insights from reviews through the means of app reviews mining. For instance, Kim et al. (2012) have studied the relationship between app's ratings assigned by several end-users and the market price of the app to investigate if the price of the app was suitable according to its market performance. This involved investigating the end-users' satisfaction of purchasing the app based on the number of properly functioning app features. Similarly, Fu et al. (2013) study the variations in the number of reviews logged over time and have attempted to uncover the reasons behind the sudden logging of a large set of reviews at specific intervals (e.g., after app update release). The authors also perform sentiment analysis of the reviews to understand the end-users satisfaction levels associated with the app usage and found out that such analysis assisted in identifying the crucial aspects (i.e., requests, bugs or enhancements) of the app that needed immediate attention. In another study, Iacob and Harrison (2013) have developed a prototype tool named MARA (Mobile App Repository Analyser) that uses text mining for automatically identifying and extracting app features from reviews. However, these existing studies mainly focus on gaining meaningful insights from the app reviews (i.e., restricted only towards certain semantics of the app reviews) and ignore the aspect that the uncovered insights must be converted into some form of actionable knowledge so that app developers can initiate the necessary remedial actions for the app. For example, a prior study developed a method to identify the critical app features and indicated the order in which they need to be addressed by the app developers, but did not offer a tool to bring the undertaken research at application level nor benchmarked the performance of the prioritisation method to determine its suitability (Licorish et al., 2017).

Classification approaches have been widely used by researchers with the intent to automatically convert the reviews into actionable knowledge. For instance, Pagano and Maalej (2013) have classified reviews into four categories; rating, requirements, community reviews and user experience. Reviews referencing other reviews or apps are classified into the community reviews category, whereas the requirements category covers end-user requests, bugs related to the apps or suggestions for improvements (i.e., enhancements). Reviews expressing end-user sentiments (e.g., happy) are classified into rating category and end-user experiences indicating helpful information aiding towards increasing the quality of the app are classified into the user experience category. Maalej et al. (2016a) have extended the scope of a study conducted earlier to utilise and empirically evaluate the performance of several classification methods that classify reviews into four categories; user experience, bug reports, end-user requests and ratings. Reviews reflecting end-user experience regarding the app usage are classified into the user experience category, reviews indicating issues associated with the apps are contained in the bug reports

category while requests pertaining to the apps are classified into the end-user requests category, and reviews expressing the end-user sentiments are classified into ratings category. McIlroy et al. (2016) have classified reviews into the following categories; user interface problem, crash report, cost, update issue, optimization problem, removal requests, functionality problem, privacy concern, compatibility issue, network issue and uninteresting content. In another study, Di Sorbo et al. (2017) have performed the summarisation of app reviews by initially classifying the reviews into manually derived topics from domain knowledge such as App, GUI, Contents, Pricing, Feature or Functionality, Improvement, Updates/Versions, Resources, Security, Download, Model and Company. Similarly, Ciurumelea et al. (2018) have classified reviews restricted to Android apps into the following self-explanatory categories; Price, Performance, Complaint, Device, Hardware, Licensing, Privacy, UI, Security, App Usability, Android Version, Memory and Battery. That said, Vu et al. (2019) have utilised the manual taxonomy developed by Di Sorbo et al. (2017) to classify app reviews and respond to these reviews automatically using system generated responses. While such studies summarise the information conveyed by the reviews into specific categories of interest through classification approaches, *these approaches fail to identify the important categories or reviews that reside in those categories and does not indicate the order in which they need to be addressed as a part of app maintenance and evolution.* Moreover, *the majority of the approaches do not eliminate the duplicate instances of the same reviews getting classified into multiple categories constituting towards unwanted redundant information* (Maalej et al., 2016a; McIlroy et al., 2016). Both these limitations of the works employing a classification approach are of concern as app developers have to eventually traverse through numerous classified reviews to uncover their specifics (e.g., which important app feature has problem or is being requested), which requires time and demands strenuous efforts from the app developers for performing manual analysis. Such a scenario demonstrates the need for a prioritisation approach.

That said, to our best knowledge, only few preliminary prior studies (with various limitations) from the app domain have researched on prioritisation approach for converting reviews into actionable knowledge. For instance, a study specific to app reviews mining has classified numerous reviews using topic modelling and an unsupervised algorithm (i.e., LDA - Latent Dirichlet Allocation) into various categories. Later, the study generates priorities of the categories and the reviews present in those categories using a prioritisation method that was developed as a result of the incorporation of multiple unjustified criteria (Chen et al., 2014). However, *the study does not thoroughly assess the performance of the prioritisation method (i.e., study did not conduct the validation of the generated priorities of all the categories and its reviews)*. Furthermore, to generate the priorities of each category the method considers criteria such as the number of reviews present inside a category, the priorities computed for a category over time and the average rating of the reviews within a category. Subsequently, the priority of each review is computed based on criteria such as the proportion of a review with reference to the other reviews within a category, the similar types of reviews within a category (i.e., duplicate reviews),

the posterior probability of the review, the rating and timestamp of a review (i.e., the time and date the review was logged). Thus, as observed from the above-mentioned criteria, it is obvious that the method tends to generate higher priorities of the categories that hold more reviews than the others but certain categories holding lesser reviews might be equally or more important. For instance, consider a category holding 100 unique reviews and another category holding 25 unique reviews. Based on the category-based prioritisation criteria, the prioritisation method would always generate a higher priority of the first category than the second category as it holds greater number of reviews. Secondly, the method does not eliminate the duplicate instances of the same reviews classified into different categories (due to unsupervised classification) creating redundant information to act upon that leads to the question '*what is the correct priority of a review having different priorities across multiple prioritised categories?'* More to this, studies show that there are discrepancies between the ratings assigned and the reviews logged by the end-users, thus questioning the judgement to utilise rating as a criteria to prioritise reviews or categories in the way it was used (Aral, 2014; Fu et al., 2013; Ganu et al., 2009; Rodrigues et al., 2017). For instance, Pagano and Maalej (2013) have found out that the reviews falling under different ratings categories (i.e., 1-5) highlighted feature requests, shortcomings of the app or provided helpful information towards improving the quality of the app. In addition, based on the timestamp criteria Chen et al.'s (2014) method assigns higher priorities to the reviews that are the latest ones, thus making the method bias towards new incoming reviews. This suggests that there might be scenarios where the old reviews and their respective categories of actual importance might never be brought to the notice of the app developers. In another study, Licorish et al. (2017) have filtered reviews that had ratings less than or equal to three to identify and prioritise app features (e.g., interface) present in those reviews that might need attention. However, as the reviews were filtered using the unreliable rating criteria, some of the useful reviews could have been potentially left out which could cause a loss of significant information required towards app improvement. Furthermore, as the generalised priorities of the app features (i.e., an average priority score of each app feature) are computed, the details regarding requests, bugs or enhancements associated with the app features (e.g., "the interface fails to load properly on my Samsung s7") stay hidden and their importance for planning the necessary remedial actions remains undetermined. Similarly, Gao et al., (2018) have developed an approach that prioritises app reviews using different criteria. However, this approach was validated based on its usefulness i.e., how useful did the app developers find the approach towards its utilisation for app maintenance and evolution tasks, but the authors did not benchmark the performance of the approach based on standard metrics such as accuracy and time.

That said, both studies (Chen et al., 2014; Licorish et al., 2017) lacked the vital and standard dimensions (e.g., time required for prioritisation, accuracy of the generated priorities, and so on (refer to Chapter 4, sub-section 4.1.5) based on which the performance of the prioritisation method could be benchmarked. Benchmarking the performance of a prioritisation method is crucial as it helps to determine the

effectiveness of the method and allows the researchers to investigate the method so that it can be optimised or tuned for an improved performance (Achimugu et al. 2014b).

In addition, Gao et al., (2015) have developed a tool that prioritises app reviews based on semantics and sentiment of those reviews. However, this tool attempts to identify and prioritise bugs (issues) associated with the app and does cover the requests or enhancements pertaining to the app. Similarly, Jiang et al., (2019) have proposed a new prioritisation approach that identifies new feature requests mentioned in app reviews and later prioritises these feature requests. However, this approach is restricted to new feature requests and does not cover prioritisation of existing features, bugs or enhancements.

2.2 Next Release Problem (NRP)

As the studies from the app domain suffer from several limitations as discussed above and did not provide adequate guidelines towards the solving of the problem related to the prioritisation of numerous reviews reflecting end-users' requests, bugs or enhancement, our further investigation lead towards the NRP. NRP in some cases is termed an NP-hard problem and in the software engineering community is commonly known as the problem to determine the optimal next release of a software product as the enterprise creating and maintaining a software product faces a steady stream of incoming requirements over software product evolution (Bagnall et al., 2001; Sureka, 2014). The cases in which NRP is termed as NP-hard is when the number of feasible prioritisation solutions increase exponentially with the increase in the number of requirements to prioritise (Sureka, 2014). In such cases, the number of potential prioritisation solutions for 'N' requirements is '2^N' and as the number of requirements increase it becomes practically extensive or unlikely for software developers to conduct an exact search to compute or identify an optimal prioritisation solution (Sureka, 2014).

As the enterprises face the challenge of deciding which requirements to address considering the imposed constraints (e.g., feasibility, time, budget, etc.) for the next release of the software product towards meeting the needs of their stakeholders, this makes prioritisation of the requirements inevitable and there has been a demand for requirements prioritisation methods (Bagnall et al., 2001; Sureka, 2014). It should be noted that the NRP problem is similar to the problem to prioritise numerous useful reviews pertaining to an app i.e., '*which end-users' requests, bugs or enhancements to address before launching the next release version of the app?'* (Bagnall et al., 2001; McIlroy et al., 2015). Figure 3 visualises the generally observed relationship between the stakeholders (i.e., end-users and software team) and the software product development, maintenance and evolution process from the software product enterprise's perspective (Wnuk et al., 2009). The end-users state their requirements and often provide feedback which indicates their experience regarding the software product's access or usage. It is then the primary task of the software team to address the prominent requirements or crucial pointers (i.e., to

software feature requests, reported bugs or enhancements) present in the feedback. Software development, maintenance and evolution cycles typically follow the requirements engineering phase before software testing is done towards the product or its update release. During such cycles, the question most often at the forefront of the software team is '*which requirements or feedback should be addressed and in what order?'* End-users may answer this question during development or post release of the product, especially when there is limited information (i.e., requirements or feedback) to convey. However, as the information scales upwards in abundance because of voluminous feedback from end-users, the software developers face challenges in manually processing and deciding which aspects of the information need to be actioned before the next release of the product. Hence, prioritisation becomes an important component in the requirements engineering phase, as it plays a crucial part in significantly assisting the software product enterprise to deliver continuous value to the end-users (Lehtola & Kauppinen, 2006).

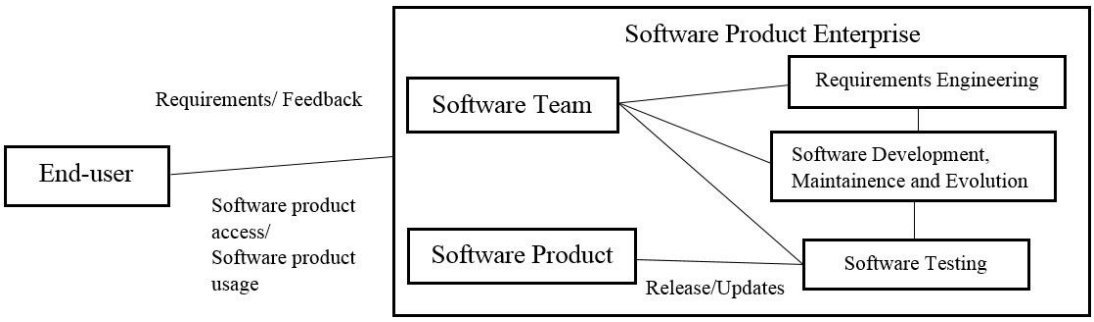


Figure 3. Relationship between an end-user and the software product development, maintenance and evolution process

2.3 Requirements Prioritisation

Requirements prioritisation is a domain that has diverse studies dedicated towards solving of the NRP, and is often scrutinised by researchers in the requirements engineering phase of the software engineering discipline (Berander & Andrews, 2005). Requirements prioritisation deals with the ranking or classification of software product requirements based on their severity or importance. This is because, identifying and addressing prioritised requirements assists valuably in releasing the software product with most prominent features in the market or launching essential software updates. Irrespective of the software development model followed by the software team (e.g., Waterfall, Agile, Spiral, and so on), a suitable requirements prioritisation method helps with the identification and fulfilment of the requirements. This in general contributes towards enhancing the performance and quality of the software product with regards to the changing market conditions (Achimugu et al., 2014b). Prioritisation is particularly crucial when there are numerous requirements and feedback in the form of crowdsourced information such as useful reviews, and we believe that methods from the requirements prioritisation domain may influence the development of a reliable prioritisation approach for app

reviews (i.e., to determine which reviews are important and indicate the order in which they could be addressed) (Hosseini et al., 2015; Licorish et al., 2017; Maalej et al., 2016a; Pagano & Maalej, 2013).

Several requirements prioritisation methods have been developed to prioritise requirements. The common ones are AHP (Analytical Hierarchical Process), MoSCoW (Must, Should, Could, Won't), CV (Cost Value), QFD (Quality Function Deployment), CVA (Cost Value Analysis) and NA (Numerical Assignment) (Achimugu et al., 2014b). Among these, the AHP method is the most popular and is commonly used to perform requirements prioritisation (Achimugu et al., 2014b). In fact, we noticed that most of the prioritisation methods have some methodological concepts inherited from the AHP method. The AHP method is a mathematical framework augmented by components that support decision making aspects of human beings required towards the prioritisation of requirements. Figure 4 presents the structure of the AHP method where the alternatives are a set of possible priorities that can be generated based on the specific criteria to achieve a particular goal. Alternatives are set to certain values such as low, medium, or high. Criteria would determine the various factors that influence the prioritisation process such as cost, risk, feasibility, scope, importance, complexity, and so on. The goal is based on alternatives and criteria which are then used to generate the final outcome such as setting the priority of a requirement.

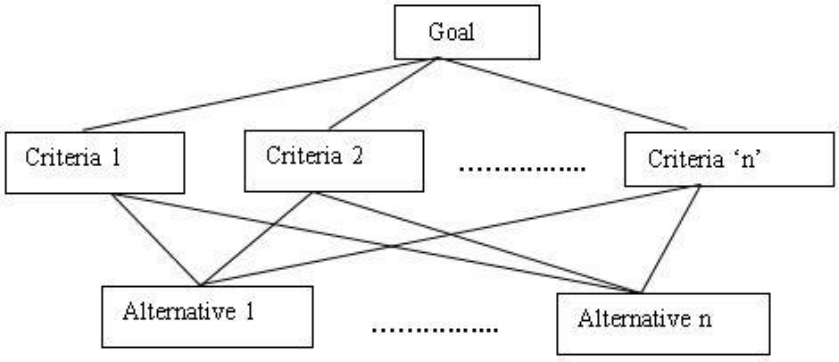


Figure 4. AHP working network

While AHP computes the priorities of each requirement, the MoSCoW method is one of those requirements prioritisation methods that determines the priority of a requirement based on the category the requirements gets classified into (i.e., classification based requirement priority generation). This method classifies each requirement in any one of the following four categories; MUST – points towards a requirement which is of utmost important and should be addressed first. SHOULD – indicates a requirement of a high priority and needs to be addressed on a regular basis. COULD – suggests a requirement of less importance in comparison to MUST and can be addressed later. WON'T – highlights a requirement which can be addressed last or may not be addressed at all as it does not have importance. In the requirements engineering phase, all the requirements from the MUST category are given attention by product developers followed by the requirements of the SHOULD, COULD and

WON'T categories. Despite of being a classification-based prioritisation method, the application of MoSCoW is best suited when the number of requirements are limited. For instance, Kravchenko and Sergey (2017) have used the MoSCoW method to prioritise eighteen requirements of six stakeholders that played a crucial role in a bank' communication management process. Other traditional methods such as QFD, PG (Planning Game), BST (Binary Search Tree), HCV (Hierarchical Cumulative Voting), Weiger's method and Win-Win method also attempt to address the requirements prioritisation problem (Achimugu et al., 2014b). Reviewing these methods, it was concluded that the utilisation of the particular requirements prioritisation method is dependent on the type of requirements prioritisation problem that is to be solved and these methods have their individual advantages and disadvantages (Achimugu et al., 2014b). In addition, software teams often use an appropriate prioritisation method that suits their prioritisation application requirement (Ryan & Karlsson, 1997). For instance, the AHP method performs well in accurately prioritising a small set of requirements but suffers from scalability issue (i.e., takes more time to prioritise with extensive use of computing resources) when dealing with numerous requirements due to its pairwise comparison mechanism, especially if the number of stakeholders' inputs increase (Achimugu et al., 2014b). On the other hand, the EVOLVE prioritisation method through means of machine learning approaches along with the stakeholders' priority preferences of requirements overcomes the scalability issue when prioritising numerous requirements but does not perform well when dealing with the dynamically changing priorities of the same set of requirements over time (Greer & Ruhe, 2004).

Furthermore, more recent efforts from researchers have focussed on merging different methods together (Abou-Elseoud et al., 2016; Ahmad et al., 2011). For instance, Ahmad et al. (2011) have combined CV and BST to prioritise a small set of requirements belonging to a mobile application whose end-users were geographically distributed. In another study, Abou-Elseoud et al. (2016) merged AHP, CV and QFD to prioritise a few illustrated requirements belonging to a software development company. This method dynamically computed the hierarchical levels (Alternatives, Criteria and Goal) and created two separate levels to prioritise the requirements. The CV method was used to rank the requirements at the lower level while the QFD operated at the upper level to generate the list of prioritised requirements. Similarly, Berander and Jonssen (2006) have addressed the weakness of CV and AHP, and combined their strengths to develop HCV that partitions complex requirements into smaller low-level requirements to create a hierarchical structure of all the requirements and later generates the priorities of these requirements based on the priority preferences of the product's stakeholders. However, the application of HCV was found to be restricted to only a small set of requirements. From the above-mentioned studies it was concluded that, to achieve the objective of prioritisation in the most optimal way researchers or software teams often aim to use a method that requires less time to reliably prioritise the given set of requirements with the assurance of optimal utilisation of allocated resources (e.g., manpower, time, computing power and so on) (Achimugu et al., 2014b).

The examples of the requirements prioritisation methods mentioned above assist with requirements prioritisation and there exist a plethora of promising requirements prioritisation methods across several disciplines that may influence the development of a prioritisation method to prioritise numerous useful reviews (Achimuguet al., 2014b). For instance, for many years the product manufacturing discipline is facing the requirements prioritisation challenge and the researchers from that discipline have explored the utility of several prioritisation methods for addressing the challenge (Chen & Yu, 2014; Nepal et al., 2010). Similarly, the researchers from software engineering discipline have explored prioritisation methods confined to the same discipline (Pergher & Rossi, 2013). Thus, *there exists an opportunity to comprehensively and systematically survey the literature available on requirements prioritisation from different disciplines that may influence the development of new prioritisation method(s). In addition, the findings of such survey may inspire a discipline (e.g. software engineering) to consider the strengths or address the weakness of requirements prioritisation method(s) from other disciplines or vice-versa*. Moreover, challenges appear for prioritisation when there are numerous requirements to deal with, especially in the case of crowdsourced requirements logged by large number of end-users (Iacob & Harrison, 2013; Maalej et al., 2016b) (e.g., as evident for app reviews (Iacob et al., 2014; Licorish et al., 2017)). These challenges occur mainly due to the conventional prioritisation methods constantly demanding the involvement of humans to ensure the reliability of the prioritisation outcome which in turn increases the overall time required for prioritisation and simultaneously compromise the scalability of the prioritisation method (Achimugu, et al., 2014b; Licorish et al., 2017). For instance, Licorish et al. (2017) were unable to discover uniform markers that could potentially remove the need for human involvement when utilising multiple regression for prioritising app features.

Hence, the first phase of this study deals with the detailed investigation of the requirements prioritisation domain by performing a comprehensive systematic mapping study of the same. The objective of this mapping study is to understand what has been done in the requirements prioritisation domain across all disciplines, and later conduct a critical evaluation of empirical studies that have provided knowledge towards the understanding of requirements prioritisation methods that are spread across different disciplines. This will assist in understanding and evaluating the various prioritisation methods that might provide the inspiration and essential guidelines towards solving the problem of prioritising numerous useful reviews.

In the next chapter, we provide the details regarding the conducted systematic mapping study on requirements prioritisation.

3 Systematic Mapping Study on Requirements Prioritisation

This chapter provides the details of the systematic mapping study on requirements prioritisation that was carried out to uncover what exists across the requirements prioritisation domain to seek inspiration from the prominent studies existing in this domain towards solving the problem of prioritising numerous useful reviews. That said, the four phases reported in this chapter follow an empirical research methodology since such a methodology is extensively followed in academia as well as industrial software engineering research. This is mainly due to the methodology being based on data and observations rather than general theories (Abran et al., 2004). As noted from the previous chapters, in phase 1 we conducted a comprehensive systematic mapping study of the requirements prioritisation domain and performed a critical evaluation of the empirical studies that provided implementations of the requirements prioritisation methods across multiple disciplines. The overarching RQ driving this study is:

RQ1. What is the state-of-the art of requirements prioritisation?

To conduct the necessary investigations, RQ1 was decomposed into six RQs aimed towards understanding the interest in requirements prioritisation across multiple disciplines (RQ1.1), approaches followed by researchers to conduct research on requirements prioritisation (RQ1.2), the form of requirements prioritisation contributions provided by the researchers (RQ1.3), the various requirements prioritisation methods (RQ1.4), the dimensions that were evaluated for empirical requirements prioritisation methods (RQ1.5) and the performance outcomes of the evaluations along with the investigation of any existing evidence regarding the relationship between attributes of the empirically evaluated requirements prioritisation methods and their performance outcomes (RQ1.6).

The detailed elaboration of the above-mentioned systematic mapping study is provided in the relevant sections (i.e., sections 3.1 to 3.4) below.

3.1 Introduction

By means of this systematic mapping study, we identify requirements prioritisation methods proposed by researchers and later consider those empirical methods that are noteworthy for further evaluation. This allows us to generate a knowledge base for software engineering researchers to understand the different problem specific solutions generated via requirements prioritisation methods that are available for use from disciplines other than software engineering. In this way, researchers from a discipline can become aware of the ways in which other disciplines have addressed the requirements prioritisation problem and may seek inspiration from them towards addressing the encountered requirements prioritisation problem. The critical evaluations present the previously unexplored knowledge and potential limitations in a discipline that could be central to steering research on requirements

prioritisation. To our best knowledge, the findings from the systematic mapping study have not been discovered by previous review studies on requirements prioritisation.

3.2 Background and Related Studies

The ultimate success of a software product is determined by the proper execution of its requirements engineering phase which deals with the elicitation and addressing of the stakeholders' requirements pertaining to the product (Nuseibeh & Easterbrook, 2000). Most often written or graphical methods, and in certain cases, a combination of both is used to elicit requirements from stakeholders (Sommervile, 2009). User stories is the most popular written method whereas UML (Unified Modelling Language) is the most commonly followed graphical method (Sommervile, 2009). Furthermore, under the traditional software development models like Waterfall, the stakeholders usually approve their requirements identified by the software team and simultaneously resolve any conflicts related to the requirements by means of negotiations. After the requirements engineering phase is completed, the software team models the software product design and develops the software product. Sometimes the software product design and development is done incrementally where agile models such as SCRUM are followed. Thus, the requirements engineering phase lays the important foundations towards software product design and development (Zowghi & Coulin, 2005). That said, sometimes challenges are encountered during the requirements engineering phase that may compromise the development and release of a software product. For instance, Wnuk et al. (2009) while investigating the requirements engineering phase of a software product at Sony Ericsson enterprise uncovered that the software team faced challenges that were severe and needed immediate attention. One of the challenges was to address the complex requirements and dependencies that existed among those requirements that made the software team to question the feasibility of addressing those requirements. The second challenge was associated with the ineffective communication that took place between the stakeholders and software team. Another critical and notable challenge was to constantly address the numerous requirements (and software product bugs) logged by the stakeholders of the software product after the software product was released in the market. This challenge in particular has been the centre of attention as the software engineering community still lacks a reliable and efficient solution to address the challenge (Licorish et al., 2017).

Requirements prioritisation is an effective solution that is initiated at regular intervals as it guides the decisions regarding the order in which the requirements should be addressed. However, challenge remains for the software engineering community in terms of prioritising numerous requirements, especially those that exist in crowdsourced information (Groen et al., 2015; Khalid et al., 2015; Licorish et al., 2017; Maalej et al., 2016b). This is because researchers are still unable to attain satisfactory performance of requirements prioritisation methods when the number of requirements to prioritise increase significantly (Achimugu et al., 2014b; Babar et al., 2011; Licorish et al., 2017). Some survey

studies have reviewed requirements prioritisation methods from the software engineering discipline (Achimugu et al., 2014b; Pergher & Rossi, 2013; Sher et al., 2014). We examine such studies towards identifying gaps to justify our research agenda of undertaking the systematic mapping study on requirements prioritisation.

Achimugu et al. (2014b) have published a literature review on requirements prioritisation that highlights various methods which have been used in the software engineering discipline until 2014. The key finding of this study was that the existing requirements prioritisation methods suffered from performance issues which were found to be associated with the scalability of the methods, and pointed towards the need for requirements prioritisation methods that could prioritise numerous requirements reliably and efficiently. In a similar study conducted in 2013, Pergher and Rossi (2013) have reviewed only certain requirements prioritisation methods used in academic software engineering research and these methods were from studies extracted from four knowledge databases which were IEEE Xplore, ACM Digital Library, Science Direct and Springer. The authors reported that the majority of the methods focussed only on the prioritisation of functional requirements and side-tracked the non-functional ones. Moreover, accuracy was often used as a dimension to evaluate the performance of the requirements prioritisation methods. Accuracy indicated the percentage of correctly prioritised requirements based on specific ground truth data. Finally, it was suggested by the authors that researchers should emphasise on the prioritisation of non-functional requirements as they reflect important business values. In another study, Sher et al. (2014) carried out a systematic mapping study of requirements prioritisation in 2014 that was restricted to the requirements prioritisation studies of the software engineering discipline. The authors found out that most of the requirements prioritisation methods did not support business or stakeholders' goals and lacked empirical validation.

3.3 Research Questions

The previously mentioned studies have focused on obtaining the knowledge on how the research on requirements prioritisation is carried out in the software engineering discipline. However, there is scope to understand the research on requirements prioritisation across multiple disciplines with the intent of developing a possible interdisciplinary prioritisation method towards addressing the requirements prioritisation challenge. Novel and reliable requirements prioritisation methods are especially required to prioritise large scale requirements and feedback such as those existing in useful reviews (Groen et al., 2015; Licorish et al., 2017; Maalej et al., 2016b; Pagano & Maalej, 2013). Hence, we aim towards conducting a comprehensive systematic mapping study and critical evaluation for understanding the research on requirements prioritisation that is available across multiple disciplines. To achieve this, we formulate six RQs listed as follows:

RQ1.1 What has been the interest in requirements prioritisation over time, what are the different publication venues, and what are the various disciplines in which the application of requirements prioritisation exist?

RQ1.2 What approaches have been used to study requirements prioritisation?

RQ1.3 What form did the contributions of the requirements prioritisation studies take?

RQ1.4 What prioritisation methods have been studied or developed?

RQ1.5 What are the dimensions that were evaluated for requirements prioritisation methods?

RQ1.6 What are the performance outcomes of the evaluations, and is there evidence of relationship between attributes of requirements prioritisation methods and their performance outcomes?

The objective of RQ1.1 is to uncover the interest in requirements prioritisation over time and the venues (e.g., journals or conferences) where the studies on requirements prioritisation from multiple disciplines (e.g., software engineering or product manufacturing) have been published. RQ1.2 helps to understand the nature (e.g., surveys, proposed or empirically evaluated prioritisation method) of the studies conducted on requirements prioritisation. RQ1.3 aims towards analysing the type of contributions (e.g., taxonomy or tool) that are provided by researchers to solve the requirements prioritisation problem. This will assist in distinguishing theoretical postulations from the empirical requirements prioritisation methods used for solving real world requirements prioritisation problems. RQ1.4 assists in identifying the various requirements prioritisation methods (e.g., AHP or CV) existing across multiple disciplines, while RQ1.5 examines the dimensions (i.e., focus of evaluation - e.g., accuracy or time) that were evaluated while conducting research on requirements prioritisation. RQ1.6 analyses the performance outcomes (i.e., the requirements prioritisation performance that is benchmarked when a method is evaluated - e.g., 84 % accurate in prioritising requirements) of the suitable empirically evaluated requirements prioritisation methods, and aims to uncover any relationship between the performance outcomes and the attributes (i.e., criteria within a study that affects the performance outcomes - e.g., number of requirements) used by researchers for meaningful insights.

3.4 Methodology

To answer the RQs mentioned in the previous sub-section we conduct a systematic mapping study which is suitable for exploring research published on a particular topic of interest using different facets. For instance, one facet is visualising how many studies published in a particular year were empirical (Petersen et al., 2008). Figure 5 provides the visualisation of the systematic mapping study process that was followed. Figure 5 portrays that initially we scoped the requirements prioritisation problem which lead to the formulation of the RQs. Next, we developed the appropriate keywords that were used to search and find studies on requirements prioritisation from various knowledge databases. The identified

studies were subjected to an exclusion and inclusion filtering criteria which removed irrelevant studies and retained the pertinent ones. The filtered studies were subjected to reliability checks before being classified according to the developed classification schemes. Additional reliability checks were performed to assure the reliability of the results. Finally, the RQs were answered based on the obtained results of the study and the results were meaningfully visualised.

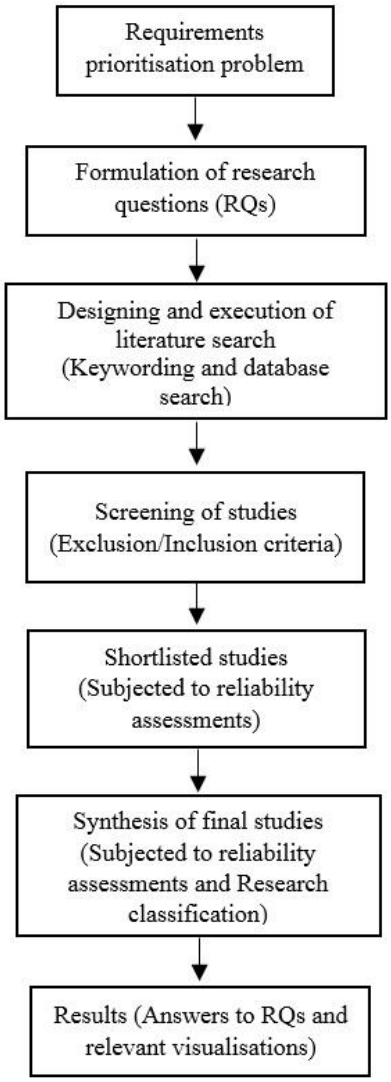


Figure 5. Systematic mapping study process related to requirements prioritisation

To find keywords that were used to search and extract studies on requirements prioritisation on a global scale we used Google's search engine (Mccallum & Bury, 2013). This was undertaken to explore the keywords related to requirements prioritisation. We followed the navigation and information search strategies that are commonly used by researchers to narrow down the keywords related to a topic of interest (Lorigo et al., 2008). Initially, *requirements prioritization* was used as a search keyword on the Google search engine and the search results from the first three pages were analysed using the guideline towards the relevance of the search results (Broder, 2002). The guideline suggests the use of an information retrieval model that is driven by humans to distinguish relevant search results pertaining to

a particular topic of interest from the non-relevant ones retrieved via the search engine. This outcome is achieved when the retrieved results are manually analysed based on human judgements and understanding of each result. That is, verifying if the contents of the retrieved search results are highly relevant to the topic of interest by means of thorough examination of the contents. Next, using the navigation and information search strategies, we uncovered the three additional most frequent keywords (along with *requirements prioritization*) related to requirements prioritisation and these were; *requirements prioritization techniques*, *requirements prioritization methods* and *requirements prioritization strategies*. The navigational strategy assists humans in visiting the web links that point towards webpages that hold the relevant contents related to the topic of interest. For instance, the retrieved web link via search engine 'https://ieeexplore.ieee.org/document/6615215' contains the study on 'requirements prioritisation'. In addition, the information search strategy initiated via navigation strategy assists in identifying the markers such as additional keywords to search for supplementary contents of the topic of interest. Finally, we adapted Kitchenham's approach for utilising the shortlisted search keywords on knowledge databases (Kitchenham, 2007). Following the mentioned approach, we expressed the keywords in the form of Boolean searches and used the right combination of 'AND' / 'OR' operators to inform accurate searches for the topic of interest. This process also helps to simplify the search, and reduce time. In executing our targeted searches, the following Boolean search strings were developed from the set of keywords that were finalized through the wider Google pilot search mentioned above. Thus, we developed the following Boolean search keywords from the set of shortlisted keywords:

1. ('requirements') AND ('prioritization')

2 ('requirements') AND ('prioritization') AND ('methods' OR 'techniques' OR 'strategies')

The two Boolean search keywords were developed so that the studies on requirements prioritisation could be extensively searched.

We used the developed Boolean search keywords to search for studies on requirements prioritisation from eight recommended knowledge databases: ScienceDirect, IEEE Xplore, Springer, ACM Digital Library, Inspec, EI Compendex, Web of Science and Scopus (Kitchenham, 2007; Rowley & Slack, 2004). These knowledge databases cover most of the disciplines given our objective to uncover studies from multiple disciplines. We had initially included Google Scholar[5] as part of the knowledge database. However, after going through the first ten pages of the search results on Google scholar we noticed that even though few relevant studies that were already captured by the performed searches on the knowledge databases mentioned earlier were evident, there were many irrelevant studies that were

[5] https://scholar.google.com/

captured (e.g., studies in which words requirements and prioritisation appeared separately but the studies were not about requirements prioritisation). Thus, we removed Google Scholar from our knowledge database list. The searches were conducted in December 2017 and the summary of the search results from the knowledge databases is provided in Table 3.1. Table 3.1 shows that Scopus had the highest number of studies on requirements prioritisation (3,325) followed by IEEE Xplore (795), ACM (499), ScienceDirect (407) and Inspec (7). It was also observed that all the search results of the second Boolean search keywords were a subset of the first Boolean search keywords. All the results of the conducted search were exported to a Microsoft Excel file for further analysis.

Table 3.1 Search results

Knowledge Database	**Keywords**				**Total**
	Requirements Prioritisation	**Requirements Prioritisation Methods**	**Requirements Prioritisation Techniques**	**Requirements Prioritisation Strategies**	
ScienceDirect	236	82	41	48	407
IEEE Xplore	478	135	139	43	795
Springer	35	1	0	1	37
ACM	325	67	78	29	499
Inspec	2	2	1	2	7
EI Compendex	6	5	2	3	16
Web of Science	118	4	16	1	139
Scopus	1964	644	457	260	3325
∑	3164	940	734	387	**5225**

After conducting the necessary searches, we followed the guidelines provided by Petersen et al. (2008) to develop an exclusion and inclusion criteria to filter studies. The criteria used is as follows:

Exclusion:

1. Study that is not available in English language.

2. Duplicate instances of the same study.

3. Studies which just mention the summaries.

4. Studies that highlight only extended abstracts or proposals.

5. Studies which are not peer-reviewed.

Inclusion:

1. Study in which the abstract clearly mentions requirements prioritisation and the study goes beyond the extended abstract.

2. Study that investigates the methods related to requirements prioritisation.

3. Studies which propose and develop methods related to requirements prioritisation.

While performing the screening of the studies, we first applied the exclusion criteria to remove the irrelevant and unwanted studies, and then applied the inclusion criteria to shortlist the pertinent ones. As all the search entries of the second Boolean search keywords were subsets of the primary Boolean search keywords, 2,061 studies were discarded. The remaining 3,164 were checked for duplicates and a total of 844 duplicate studies were detected which were then discarded. Out of the remaining entries, 72 studies were found to be just summaries and 7 studies were not documented in English language along with 6 studies that had unwanted characters. With the assistance of the exclusion criteria 2,990 studies were eliminated and the remaining 2,235 studies were subjected to inclusion criteria which removed 2024 unwanted studies with the final set of 211 studies left for further analysis. The distribution of the shortlisted studies is highlighted in Table 3.2.

Table 3.2 Final entries from knowledge databases

Knowledge Database	**Total**
ScienceDirect	18
IEEE Xplore	66
Springer	24
ACM	12
Web of Science	24
Scopus	67
∑	**211**

To ensure that we conducted proper filtering of the studies based on the exclusion and inclusion criteria, we conducted reliability assessments using Fleiss' Kappa which is the extension of Cohen's Kappa to support the independent evaluations of three or more human evaluators (Fleiss & Cohen, 1973). To ensure that no study was discarded or included by mistake, the mentioned reliability assessment was performed. Three of us (i.e., two supervisors and the PhD candidate) independently performed a screening of the total number of studies towards exclusion and inclusion. The convergence or divergence between the three was mapped to the relevant yes or no flag (i.e., study to be shortlisted or not) for each analysed study. The Fleiss coefficient was found to be 0.85 which indicated a near perfect agreement (Landis & Koch, 1977). Further discussions were held accordingly to resolve any disagreements and establish consensus (i.e., 100% agreement).

3.4.1 Classification Schemes (RQ1.2 and RQ1.3)

To answer RQ1.2 and RQ1.3 we first had to develop the relevant classification schemes related to the respective RQs. To develop a classification scheme to answer RQ1.2 we adapted Wieringa et al.'s

(2005) classification guidelines which were developed to classify studies based on the approaches[6] presented in the studies on requirements engineering. The authors state that the studies reflect approaches such as Proposal of Solution, Validation Research, Evaluation Research, Philosophical, Opinion or Experience, and hence the studies can be classified into the relevant research approach. The authors also mention that studies may cover multiple approaches. For requirements prioritisation studies we merged Opinion and Philosophical approaches to form a new approach Opinion/Philosophical as some of the shortlisted studies provided views or opinions of the authors, reports or surveys pertaining to requirements prioritisation. Secondary Evaluation/Categorisation was created as some of the studies revealed that there were a number of secondary studies. The three approaches Proposal of Solution, Validation Research and Evaluated Research were redefined into Proposed Solution, Simulated Solution and Evaluated Solution. Proposed Solution highlights studies that present requirements prioritisation solutions proposed by authors and these solutions are yet to be evaluated. Simulated Solution highlights studies that provide solutions to the requirements prioritisation problems that were evaluated only at experimental level, whereas Evaluated Solution highlights the studies that provide solutions which were empirically evaluated. The retained Experience approach presents the studies which describe the authors experience regarding utilised requirements prioritisation methods. The classification scheme for approaches followed in requirements prioritisation studies is presented in Table 3.3. The final list of approaches that are developed for classifying approaches is listed in the first column, the description of the approaches is provided in the second column followed by the suitable example in the third column.

Similarly, to answer RQ1.3 we first developed a classification scheme for classifying contributions provided by the researchers working on requirements prioritisation. Petersen et al. (2008) have provided a set of guidelines to classify studies according to the contributions provided by those studies. Following these guidelines and reviewing each shortlisted study we came up with six types of contributions; Taxonomy, Single Method, Multiple Methods, Hybrid Method, Tool and Process. The majority of the shortlisted studies covered most of the contributions, however, few studies did not provide any type of contribution. Hence, we assessed the six types of contributions against those provided by other studies. Lehtola (2017) have used a manually derived framework to classify the type of contributions into Activities, Techniques, Methods and Process. After cross checking our six types of contributions with those developed by these authors, we noticed that Methods and Process were covered with Technique being an abstraction of Method. Activities was found to be granular for adaption (i.e., activities was found to be a subset of Taxonomy or Process), and hence was discarded. Next, we cross checked the six types of contributions against those developed by Pergher and Rossi (2013) and noted that the authors have Framework as a type of contribution, however, this type of contribution was already

[6] Type of approach followed by authors for conducting research such as evaluated solution or proposed solution.

captured under our Taxonomy type of contribution, and hence, we did not include this as a separate type of contribution. Studies that did not provide any type of contribution were classified as Others.

Table 3.3 Classification scheme for evaluating research approaches

Approach	Description	Example
Proposed Solution	For solving a requirements prioritisation problem, a particular solution is proposed in the study which could be existing or new, however, the solution is not practically implemented and evaluated.	Cleland-Huang and Mobasher (2008) proposed a new solution that is built on data mining and recommender systems concepts.
Simulated Solution	Similar to proposed solution mentioned above, however, simulated solution is validated only at the experimental level mostly in the form of a solved example or simulation.	Shao (2008) have presented the simulation results of a proposed requirements prioritisation method.
Evaluated Solution	Study presents a developed empirical solution to solve a requirements prioritisation problem which may be already existing or new.	Carod and Cechich (2010) provide an empirical requirements prioritisation solution that is thoroughly evaluated beyond the experimental level.
Opinion/Philosophy	A study that presents concepts, opinions, ideas or views expressed by the surveyed participants or authors.	Babar et al. (2011) provide their opinions on the different challenges and future trends in requirements prioritisation.
Secondary Evaluation/ Categorisation	Studies present a literature review or systematic mapping study on requirements prioritisation.	Achimugu et al. (2014b) provide a systematic literature review of studies on requirements prioritisation from the software engineering discipline.
Experience	Studies that highlight authors' experience regarding the application of requirements prioritisation method(s) on a requirements prioritisation problem.	Berander and Svahnberg (2009) have experimented with HCV and have worked on different ways towards generating the priorities of requirements.

Table 3.4 highlights the type of contributions, provides the description of these contributions and mentions an appropriate example of the same. The final list of contributions that are developed for classifying contributions is listed in the first column, the description of the contributions is provided in the second column followed by the suitable example in the third column.

Table 3.4 Classification scheme for evaluating research contributions

Contribution	Description	Example
Taxonomy	Studies that provide a taxonomy describing methods, challenges, future trends and so on.	Babar et al. (2011) described the limitations of the requirements prioritisation methods and pointed towards the need for an automated requirements prioritisation that could process numerous requirements.
Single Method	Study that presents a single method to perform prioritisation of requirements.	Sadiq et al. (2009) used AHP to prioritise a small set of requirements.
Multiple Methods	Two or more requirements prioritisation methods are presented in a study	Felfernig and Ninaus (2012) have evaluated multiple heuristic methods (least distance, standard deviation, random selection, average value, median based, majority voting and ensemble) to prioritise requirements.
Hybrid Method	A study presents a hybrid method that combines and synthesises aspects (e.g., prioritisation mechanism) from two or more methods.	Abou-Elseoud et al. (2016) developed a hybrid requirements prioritisation method that combines decision matrix method with CV method.
Tool	Tool based contributions represent software artefacts or prototypes that prioritise a given set of requirements. The tool can be in the form of an app, website and so on.	Ryan and Karlsson (1997) have implemented a prototype requirements prioritisation tool for the Ericson Radio Systems.
Process	Studies provide an elaborate description of the various steps involved in the prioritisation of requirements. These steps could be planning and executing the activities related to requirements prioritisation, participation of stakeholders and product teams in those activities and the evaluation of the outcomes associated with those activities.	Lehtola and Kauppinen (2006) have documented all the practices and their associated challenges while carrying out the prioritisation of requirements in a software company. The authors initially gather information on requirements from the participants of different companies and later evaluated the outcome of the conducted requirements prioritisation process.
Others	Studies did not provide any type of the above-mentioned contributions.	Forouzani et al. (2012) developed a tool that provides teaching regarding requirements prioritisation and not an actual tool to prioritise requirements.

It is to be noted that there is no specific methodology associated with RQ 1.1, RQ1.4, RQ1.5 and RQ1.6, as answering these would require reviewing the studies individually and noting the findings, and later performing reliability assessments to validate those findings. For instance, RQ1.1 deals with identifying the number of studies on requirements prioritisation published each year, the different publication venues where the studies were published and the different disciplines the studies belonged to. Answering such research question would require a manual review of each study to identify its year of publication, the study's publication venue and discipline of the study. For answering RQ 1.4, each study needs to be reviewed to identify the requirements prioritisation methods examined in the study.

Similarly, for answering RQ 1.5, each empirical study out of those which were shortlisted as an outcome of the systematic mapping study on requirements prioritisation needs to be reviewed to identify the dimensions that were evaluated for requirements prioritisation method. The analysis of the outcome achieved through means of RQ1.5 would finally assist us to answer RQ1.6.

While developing the classification schemes to answer RQ1.2 and RQ1.3, the two supervisors and PhD candidate performed reliability assessments where each one of us independently assessed the shortlisted 211 studies to manually classify each study into the particular approach or contribution of the above-mentioned classification schemes. That said, Fleiss coefficients 0.78 and 0.82 were returned respectively indicating substantial agreement and a near perfect agreement (Landis & Koch, 1977). Follow up discussions were held among us to resolve any disagreements and establish consensus (100% agreement) to answer RQ1.2 and RQ1.3.

We repeated the same reliability assessment procedure after answering the remaining RQs (i.e., to validate the authenticity of the generated results) where the respective outcomes answering RQ1.1, RQ1.4, RQ1.5 and RQ1.6 from the 211 studies were subjected to reliability assessments. We noted a Fleiss coefficient of 0.83 indicating a near perfect agreement (Landis & Koch, 1977). Follow up discussions were held accordingly to resolve any disagreements and establish consensus to achieve 100% agreement after performing all the reliability assessments.

That said, the full list of the shortlisted studies from which the necessary information was extracted for answering RQ1.1 to RQ1.6 is made available in the Appendices (refer to section A). The results from answering RQ1.1 to RQ1.6 are presented in section below.

3.5 Results

In this section, we report the results of the systematic mapping study on requirements prioritisation. These findings provide insights on the research interest, publication venues and disciplines pertaining to requirements prioritisation (RQ1.1), the approaches followed by researchers to conduct research on requirements prioritisation (RQ1.2), the different types of contributions provided by the researchers towards requirements prioritisation (RQ1.3), the various proposed requirements prioritisation methods (RQ1.4), the distinct dimensions evaluated in empirical requirements prioritisation studies by researchers (RQ1.5) and the requirements prioritisation performance outcomes reported in empirical studies and the relationship between the identified attributes and those outcomes (RQ1.6). In addition, these results also provide triangulations for RQ1.

3.5.1 Interest, Publication Venues and Disciplines (RQ1.1)

Figure 6 provides a summary of the requirements prioritisation studies that were published over the past years. Out of the shortlisted 211 studies, the first study appeared in 1993 and in the subsequent years

the publication of at least one paper up to 1998 is observed. As no studies were published between 1999 and 2003, a reduced interest in requirements prioritisation research can be concluded. On the contrary, there has been an increase in publications on requirements prioritisation since 2004, with 2017 showing the highest number of studies published.

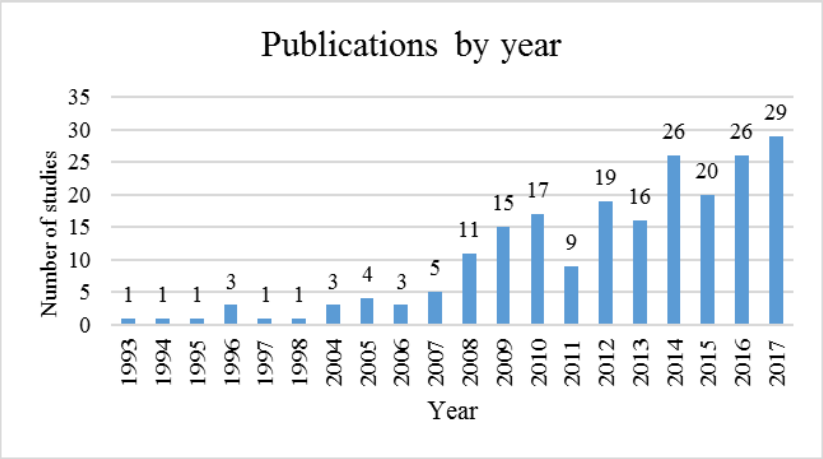


Figure 6. Requirements prioritisation publications summary over the past years

Next, we report the venues targeted by researchers for publishing the studies on requirements prioritisation. From Figure 7 it is evident that majority of the studies were published in conferences (48.8% or 103 studies), followed by journals (35.5% or 75 studies). Of the total studies, 16 studies were published in workshops (7.6%), 9 were published as book chapters (4.3%), 7 were published in symposiums (3.3%) and 1 was published in a world forum.

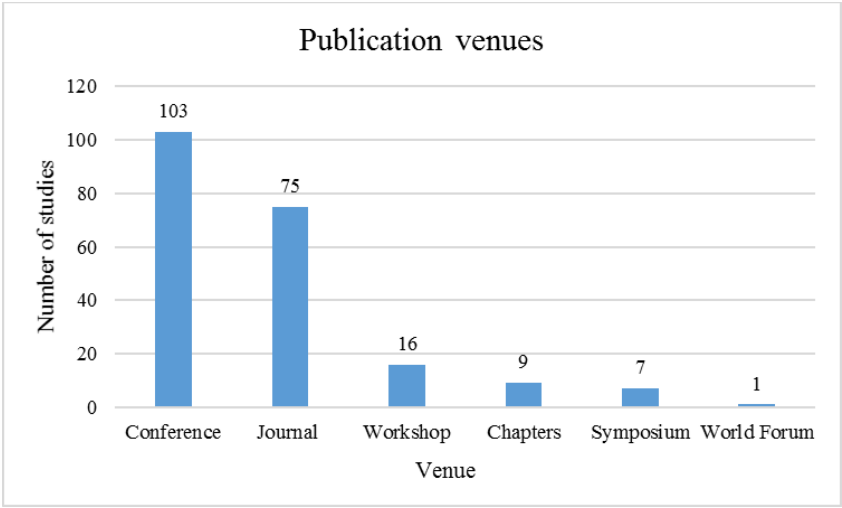


Figure 7. Requirements prioritisation publication venues

Next, we examine the disciplines for the publications on requirements prioritisation in Figure 8. Figure 8 shows that the majority of the studies were from the software engineering discipline (82.9% or 175 studies), and 22 studies (10.4%) were from the product manufacturing discipline. The remaining studies

were from education (5 studies), finance (4 studies), real estate (3 studies), law (1 study) and transport (1 study) disciplines respectively.

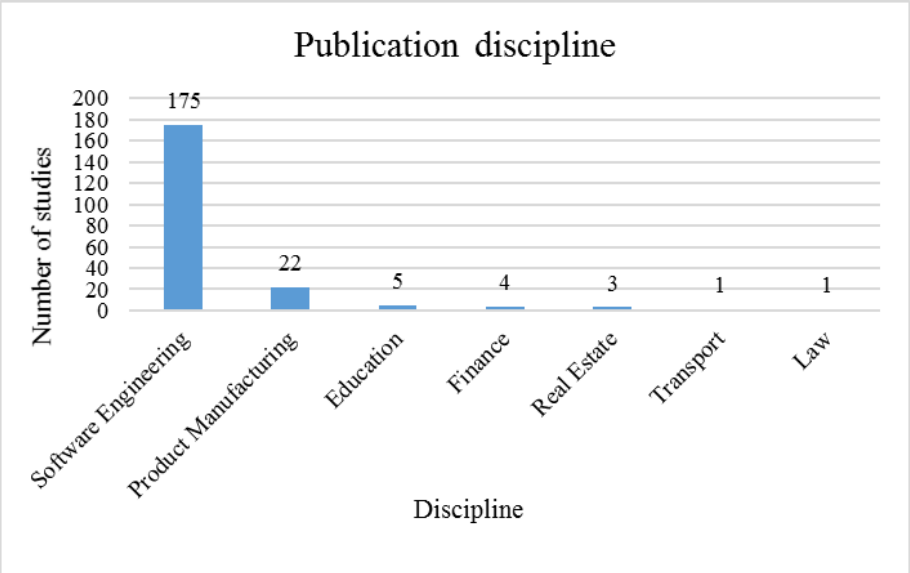


Figure 8. Requirements prioritisation publication disciplines

To gain further insights into the disciplines and publication venues, we plot a bubble chart as shown in Figure 9. Such type of visualisation is common in systematic mapping studies as it helps to analyse the findings from multiple facets (Petersen et al., 2008). We have utilised such visualisations in the remaining sections of the systematic mapping study on requirements prioritisation. Figure 9 shows that software engineering discipline had the highest number of studies published in conferences, journals, workshops, book chapters and symposiums in comparison to product manufacturing (175 versus 36 studies) and other disciplines. Although few studies were published across the other disciplines, we can observe a similar pattern for education, finance, and real estate disciplines in Figure 9.

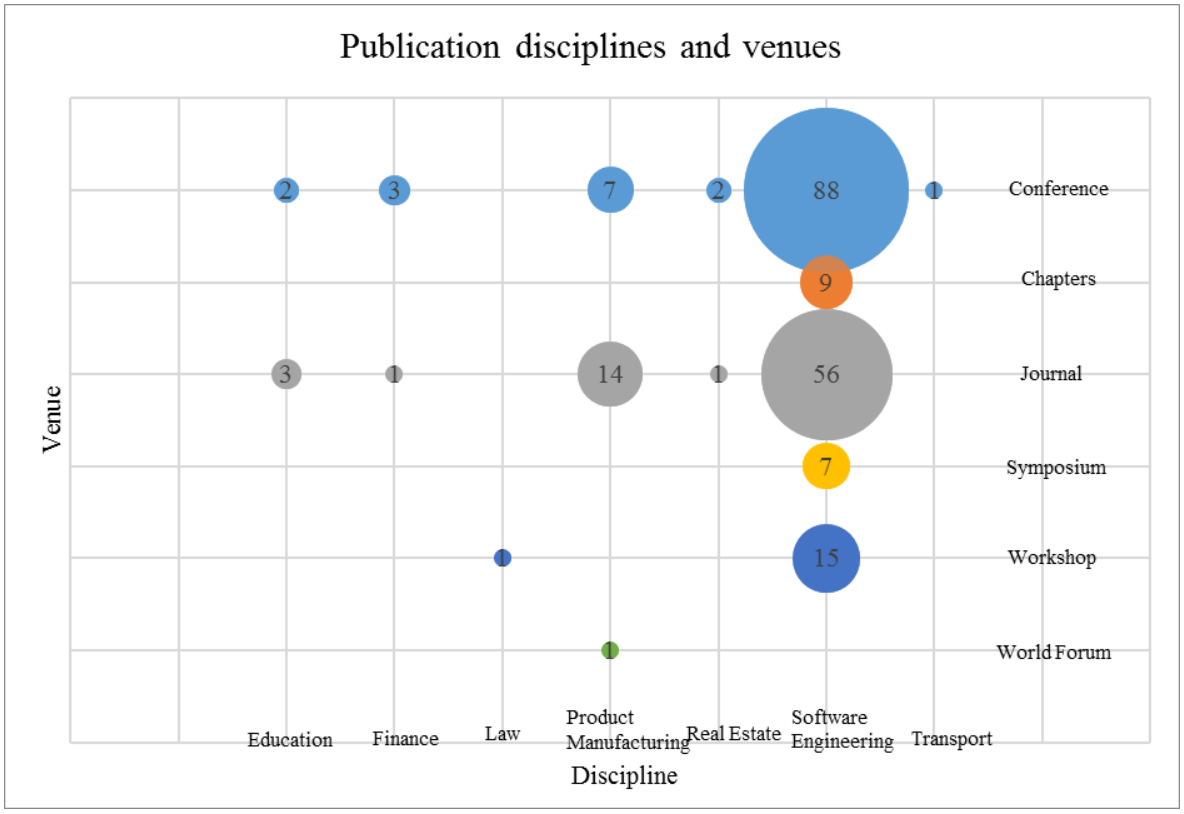


Figure 9. Requirements prioritisation publication disciplines and venues

3.5.2 Requirements Prioritisation Approaches (RQ1.2)

We report the approaches provided in the 211 studies on requirements prioritisation and plot a summary of it in Figure 10. Figure 10 shows that 91 studies (43.1%) proposed and empirically evaluated a requirements prioritisation solution, 32 studies (15.2%) proposed a requirements prioritisation solution but the solution was not evaluated, 31 (14.7%) studies highlighted authors' experience with requirements prioritisation, 28 (13.3%) studies presented a simulated solution, 17 studies (8.1%) were found to be secondary evaluation or categorisation, and 13 studies (6.2%) stated authors' opinion/philosophy. Overall, the number of studies reported in Figure 10 add up to 212 as one study provided experiences as well as a proposed solution.

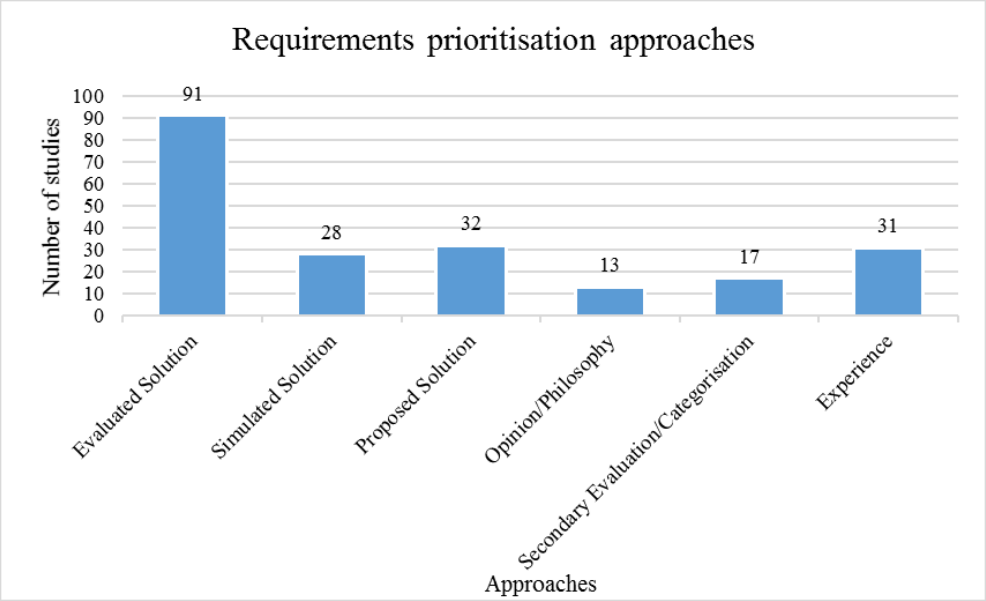


Figure 10. Requirements prioritisation approaches

Next, we plotted approaches across the various disciplines in Figure 11, where it can be observed that of all the approaches followed in the software engineering discipline, evaluated solutions were provided by a large number of studies (37.1% or 65 studies). The higher number of studies reflecting evaluated solution were also observed in cases of product manufacturing, education, law and real estate disciplines. Interestingly, it is to be noted that only software engineering discipline has reviewed the secondary evaluation/categorisation studies. However, such studies were not undertaken in other disciplines. In Figure 11, studies classified into various approaches under the software engineering discipline add up to 176 (instead of 175), since one study provided experiences as well as a proposed solution.

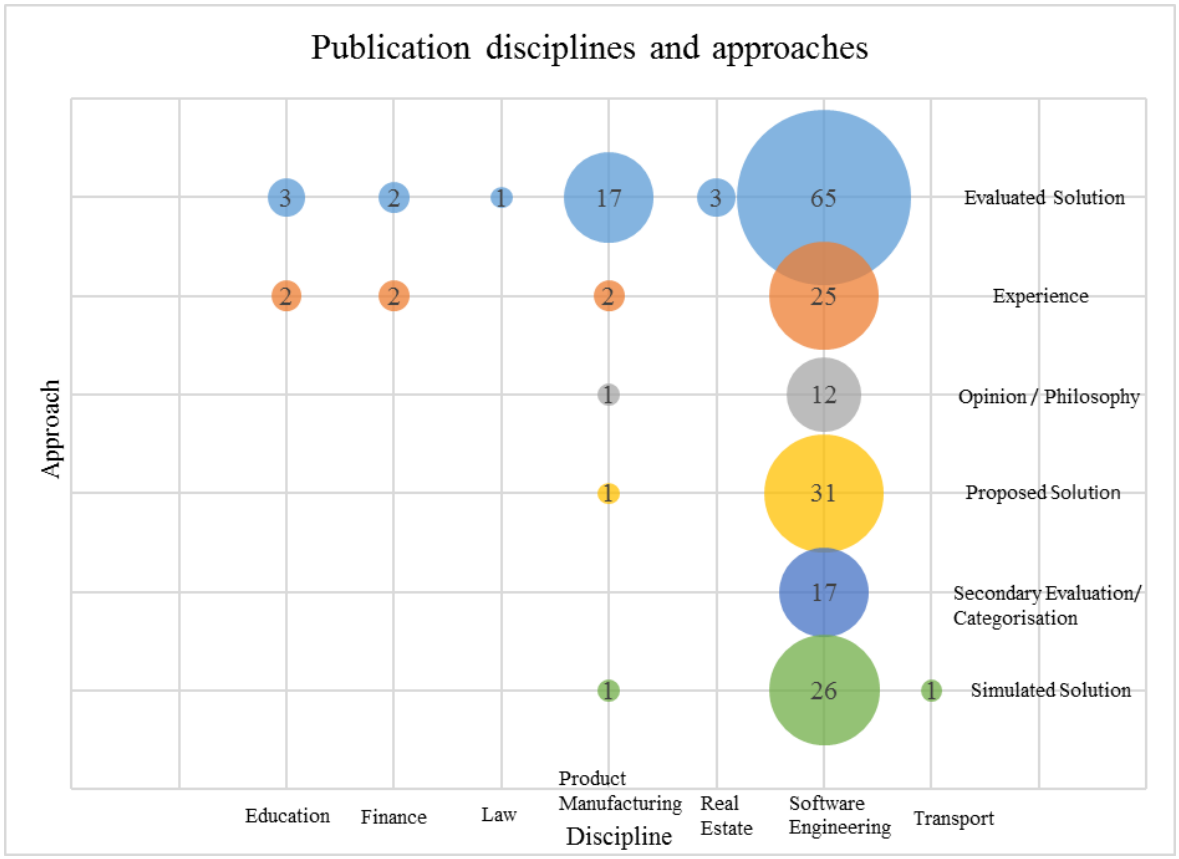


Figure 11. Requirements prioritisation publication disciplines and approaches

3.5.3 Requirements Prioritisation Contributions (RQ1.3)

We visualise the requirements prioritisation contributions provided by researchers in Figure 12. From Figure 12 it can be observed that most studies contributed a single method (32.7% or 69 studies), followed by 58 studies (27.5%) contributing hybrid methods. Researchers also experimented with multiple methods (17.5% or 37 studies) and developed taxonomies (12.8% or 27 studies). Other contributions were provided in the form of processes (7.1% or 15 studies), tools (4.7% or 10 studies), and 5 studies (2.4%) were classified under the 'Other' category. Some studies were classified under multiple contributions, and hence, a total of 221 studies (instead of 211) is observed in Figure 12. For instance, while 52 studies were classified under hybrid method, 6 were classified under hybrid method as well as multiple methods, resulting in 58 studies being classified under hybrid method. A similar pattern was observed for the other types of contributions.

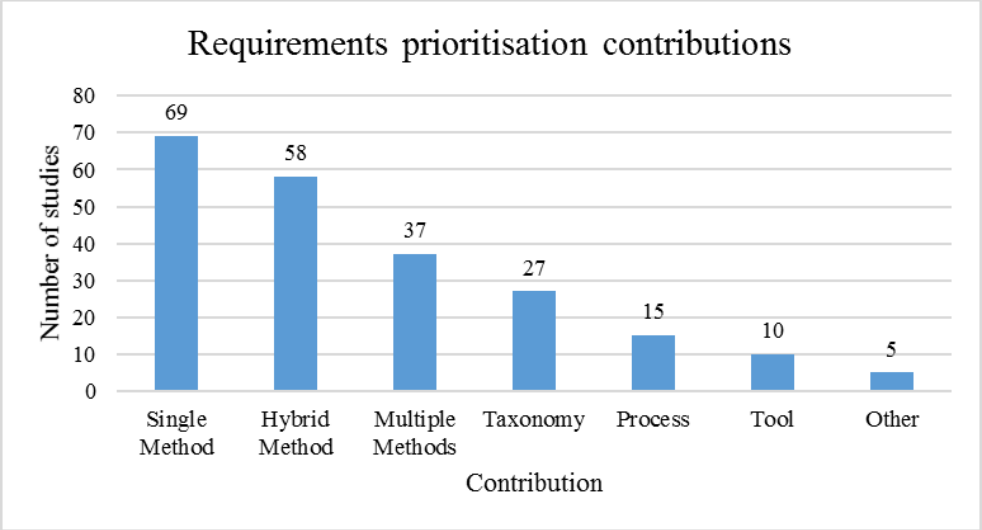


Figure 12. Requirements prioritisation contributions

Next, we plot the contributions against the approaches in Figure 13. From Figure 13 it can be observed that the single and hybrid method contributions are dominant, with frequent classification being evaluated and proposed solution. Additionally, a convergence between taxonomy contribution and secondary evaluation/categorisation, opinion/philosophy is observed. This is because, most papers have conducted secondary evaluation, have proposed a taxonomy (14 out of 17). In addition, papers that belong to opinion/philosophy, proposed a taxonomy (9 out of 13). The tools on requirements prioritisation were largely evaluated by researchers. The number of studies in Figure 13 add up to 223 (instead of 211) due to multiple classifications. For instance, one study was classified as proposed solution and experience when being reviewed to answer RQ1.2 that emphasised on requirements prioritisation approaches and the same study was also classified under taxonomy and single method when reviewed in terms of requirements prioritisation contributions to answer RQ1.3.

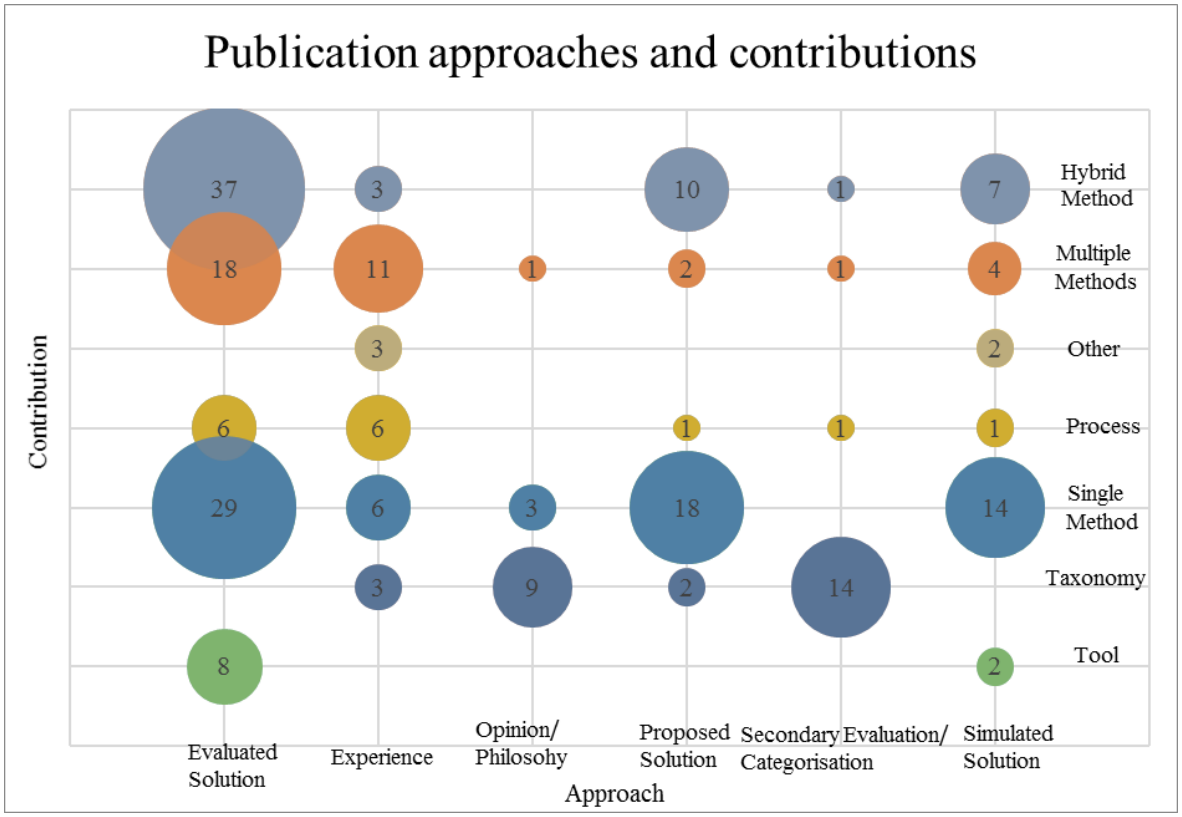


Figure 13. Requirements prioritisation publication approaches and contributions

3.5.4 Requirements Prioritisation Methods (RQ1.4)

We identified 157 different requirements prioritisation methods from the 211 shortlisted studies, with 90 of these methods researched only once and 31 methods were researched two times. The remaining 37 methods were researched three or more times. We show the top 10 frequently utilised requirements prioritisation methods in Figure 14 where it is observed that Analytical Hierarchical Process (42 studies), Cumulative Voting (13 studies) and Quality Function Deployment (12 studies) were most frequently researched. Specifically, AHP was researched across all the disciplines with contributions ranging from hybrid method to tools as observed from Figure 15. From Figure 15 it is evident that many methods were presented in different taxonomy studies and researchers frequently researched multiple methods but often did not merge multiple methods into a hybrid method.

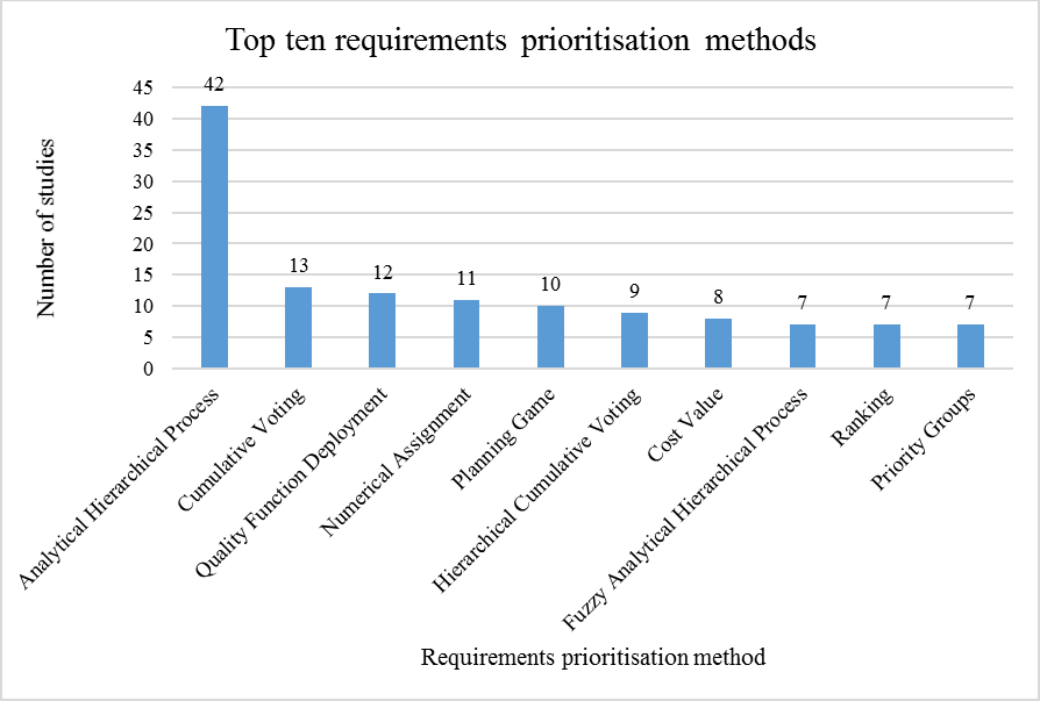


Figure 14. Top 10 requirements prioritisation methods

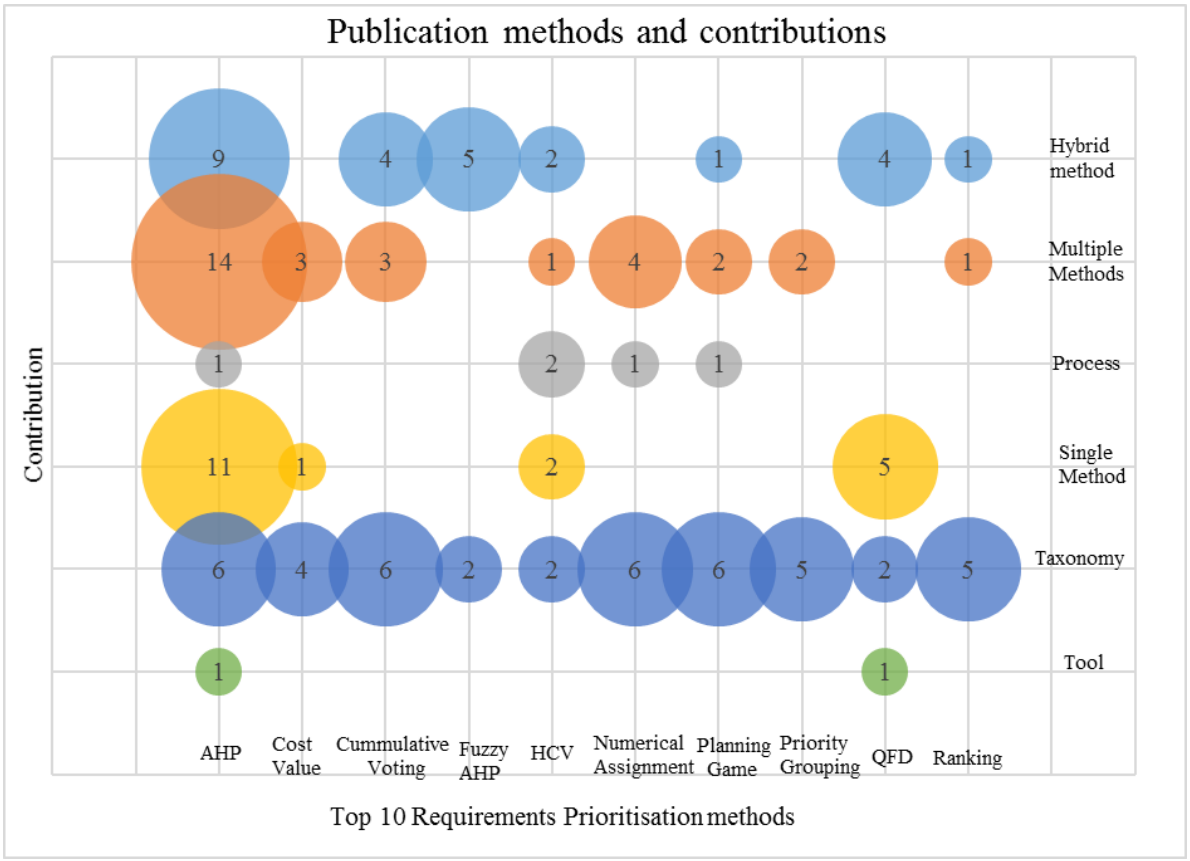


Figure 15. Requirements prioritisation methods and contributions

The full list of the requirements prioritisation methods has been made available in the Appendices (refer to section E).

3.5.5 Dimensions of evaluated requirement prioritisation solutions (RQ1.5)

To answer RQ1.5 we reviewed the 91 studies that presented an empirically evaluated solution. Out of the 91 studies, we noticed that 15 studies were the extended versions of their previous studies. For instance, Carod and Cechich (2010) have published the extended version of a previous study with additional contents such as literature, methodology, experimental results and discussions. Such predecessor studies were excluded from further review. The remaining 76 studies were thoroughly reviewed and it was observed that the solutions provided by the empirical studies were evaluated along eight dimensions; operational demonstration, accuracy, stakeholders preferences, scalability, time, requirements dependencies, requirements updates and computational complexity. Table 3.5 explains the identified dimensions with the support of a relevant example.

Table 3.5 Requirements prioritisation evaluated dimensions

Dimension	Description	Example
Operational demonstration	Requirements prioritisation method is applied to a set of requirements and only a list of prioritised requirements is returned as an outcome with no specific measure reported.	Popli et al. (2014) have proposed a requirements prioritisation method that prioritises a small set of user stores of an online Quiz system.
Accuracy	Requirements prioritisation method is applied to a set of requirements and its accuracy is reported (i.e., correct vs incorrect priorities of the requirements)	Bebensee et al. (2010) have reported that Binary priority list method was 70.0% accurate in prioritising 114 requirements while Wieger's method exhibited 45.0% accuracy while prioritising the same set of requirements.
Time	Requirements prioritisation method is utilised and the time required by the method to perform prioritisation is reported.	Nidhra et al. (2012) have reported the time required by NAcAHP and AHP to prioritise 40 requirements.
Stakeholders' preferences	Requirements prioritisation method is able to accommodate stakeholders' preferences (i.e., requirements' priorities assigned by each stakeholder) when utilised for prioritisation.	Zhaoling et al. (2009) proposed a requirements prioritisation method that is able to accommodate stakeholders' preferences and resolve any conflicts pertaining to those preferences when prioritising 4 requirements from 7 stakeholders.
Requirements dependencies	Requirements prioritisation method is able to discover dependencies among the requirements and utilise the knowledge of dependencies for prioritisation.	Yutao Ma et al. (2012) have identified dependencies among requirements and represented them in the form of a network graph to filter insignificant requirements and optimise the prioritisation process.
Requirements updates	The requirements prioritisation method is capable of dynamically updating the priorities of the same set of requirements over time based on certain changing conditions (e.g., business value of the requirements).	Peng et al. (2012) have demonstrated that in a cohort of 1878 requirements, the updated priorities of the same set of requirement groups were captured which informed the undertaken requirements prioritisation process.

Dimension	Description	Example
Scalability	Requirements prioritisation method is capable of handling and prioritising an increasing number of requirements.	Elsood et al. (2014) have determined the scalability of two requirements prioritisation methods while prioritising 7 requirements using the particular method's operational cycle.
Computational Complexity	Requirements prioritisation method aims to investigate or attempts to optimise the utilisation of system resources (e.g., memory)	Voola and Babu (2017) have utilised the Big - O notation to investigate the computational complexity of three requirements prioritisation methods while prioritising 15 requirements.

Next, we analyse the number of studies in which requirements prioritisation methods were empirically evaluated using the above-mentioned dimensions in Figure 16. It is observed that the majority of the studies provided operational demonstrations (67.1% or 51 studies). Accuracy was found to be another popular dimension used for evaluation (26.7% or 19 studies) followed by stakeholders' preferences and time (17 and 13 studies respectively). The remaining studies (31.5% or 24 studies) evaluated requirements updates, scalability, requirements dependencies and computational complexity. The number of studies reported in Figure 16 add up to 124 (instead of 76) as certain studies utilised multiple dimensions.

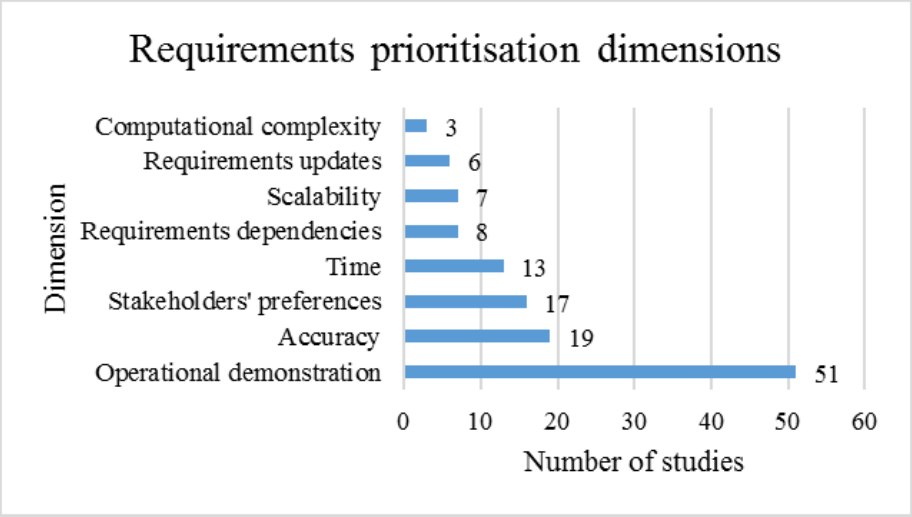


Figure 16. Requirements prioritisation dimensions

Next, we examine how these identified dimensions evaluated by the researchers conducting research on requirements prioritisation are distributed across the empirically evaluated studies and how they are linked. Figure 17 shows an undirected network graph where each node in the graph represents a dimension. The node CC indicates computational complexity, OD indicates operational demonstration, SI indicates stakeholders' preferences, ACC indicates accuracy, T indicates time, RD indicates requirements dependencies, RU indicates requirements updates and SCA indicates scalability. These dimensions are connected by a set of links and the weights on the links represent the number of studies

that contain the two connecting dimensions. The weights on self-looping links indicate the number of studies solely focusing on one particular dimension. In Figure 17 we can observe a wide spread of dimensions, where the spread of certain connected dimensions is dense while for others it is sparse. As observed from Figure 17, 33 studies solely focused on operational demonstration of a particular requirements prioritisation method, five studies worked towards the handling of stakeholders' preferences, four studies focused solely on accuracy and one study exclusively investigated the scalability dimension. Other remaining studies have evaluated multiple dimensions.

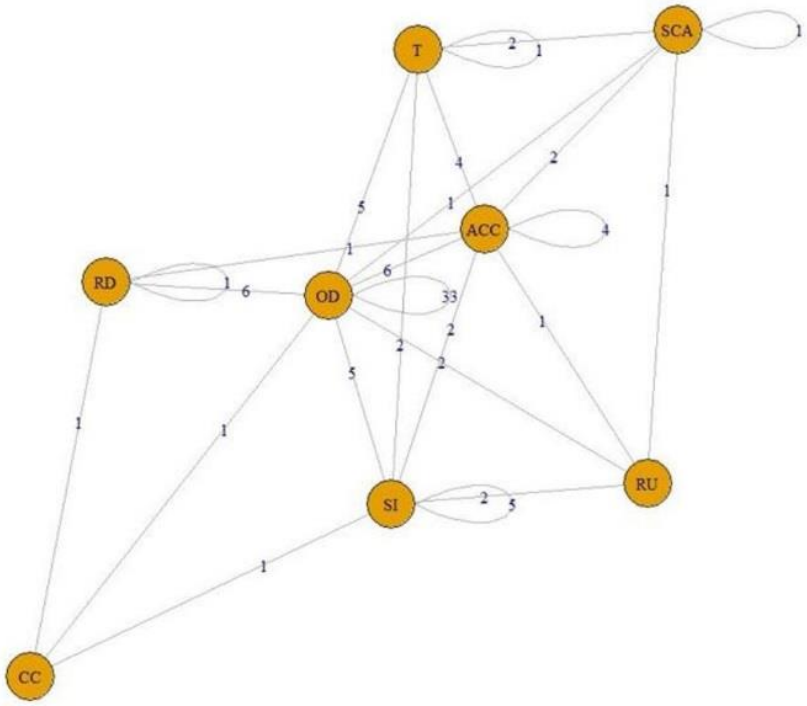


Figure 17. Representation of dimensions based on their occurrence in empirical studies

3.5.6 Performance Outcomes and Relationship between Attributes and Outcomes (RQ1.6)

Out of the eight dimensions reported in sub-section 3.5.5, accuracy was utilised to evaluate the correctness of a requirements prioritisation method to benchmark its performance. We report the studies that reported outcomes based on *accuracy dimension* in Table 3.6 wherein we report the number of requirements prioritised by the particular requirements prioritisation method along with the criteria (i.e., ground truth) used to validate the accuracy of the requirements prioritisation method and the assessment procedure used to evaluate the accuracy. It can be observed that accuracy was found to be in the range of 16% to 99% with varying number of requirements and the majority of the requirements prioritisation methods were evaluated for accuracy using the stakeholders' preferences as the ground truth.

Table 3.6 Requirements prioritisation accuracy dimension outcomes

Study	Number of requirements	Criteria	Assessment	Method	Result (%)
Asghar et al. (2013)	36	Stakeholders' preferences	Predicted priority vs actual priority	Maintainability Based Approach	16.7
Chopra et al. (2016)	103	Dataset consisting of prioritised requirements	Predicted priority vs actual priority	AHP - A1 AHP - A2 AHP - A3	35.0 68.0 90.0
Bebensee et al. (2010)	114	Stakeholders' preferences	Predicted priority vs actual priority	BPL Weiger	70.0 45.0
Achimugu et al. (2016)	1820	Stakeholders' preferences	Predicted priority vs actual priority	ReproTizer	98.9
Gärtner and Schneider (2012)	8	Stakeholders' preferences	Predicted priority vs actual priority	ConTexter	87.5
McZara et al. (2015)	100	Stakeholders' preferences	Predicted priority vs actual priority	WSM SNIPR	74.9 82.0
Kukreja et al. (2012)	31	Stakeholders' preferences	Predicted priority vs actual priority	TOPSIS	85.0
Laurent et al. (2007)	202	List of prioritised requirements	List of prioritised requirements	Automated Requirements Triage	82.2

Next, we report our findings related to the *stakeholders' preferences dimension* that was evaluated by nine studies as shown in Table 3.7, where four studies resolved uncertain stakeholders' preferences along with the conflicting ones, three studies resolved only uncertain stakeholders' preferences and two provided a solution to resolve conflicting stakeholders' preferences.

Table 3.7 Requirements prioritisation stakeholders' preferences dimension outcomes

Study	Number of requirements	Number of stakeholders	Method	Resolve uncertain preferences	Resolve conflicting preferences
Bajaj and Arora (2013)	3	3	Fuzzy alpha cut	0	1
Zhaoling et al. (2009)	4	7	Grey and weighted average method	1	1
Achimugu et al. (2014)	262	76	Metric distance	0	1
Achimugu, et al. 2014a)	4	9	Fuzzy logic	1	0

Study	Number of requirements	Number of stakeholders	Method	Resolve uncertain preferences	Resolve conflicting preferences
McZara et al. (2015)	100	45	AHP	1	1
Philip Achimugu (2014)	4	4	Fuzzy logic	1	1
Chen and Yu (2014)	4	5	Fuzzy logic and game theory	1	0
Xuemei et al. (2008)	5	5	Fuzzy weighted	1	1
Voola and Babu (2013)	20	8	Uncertainty modelling	1	0

(Legend: 0 - absent, 1 - present)

Requirements dependency dimension was evaluated by studies mentioned in Table 3.8 where it is observed that majority of the requirements prioritisation methods utilise a graph-based approach to uncover the dependencies that exist among the requirements. However, these studies have not provided any specific evaluation outcomes.

Table 3.8 Requirements prioritisation requirements dependency dimension

Study	Number of requirements	Method	Prioritisation	Dependency type
Peng et al. (2012)	1878	Ontology modelling	X	1
Delia Ilie et al. (2009)	52	Cross linking degree	Y	1
Atukorala et al. (2016)	18	Situation transition framework	Y	1
Sharma (2007)	10	Integration of requirements weights with correlation triangle values (QFD method specific only)	Y	0
Thakurta (2013)	7	Hierarchical structure	Y	1
Yutao Ma et al. (2012)	34	Network analysis	Y	1
Sureka (2014)	100	Value analysis	Y	1

(Legend: Y - Individual, X - Group-based; 1 - Graph, 0 - Matrix)

Next, we report the studies that investigated the *time dimension* in Table 3.9. Researchers have benchmarked the performance of the particular requirements prioritisation method by noting the time required by the method to prioritise a given set of requirements. Table 3.9 indicates the particular requirements prioritisation method that was evaluated in the study along with the measure of time utilised by the method to prioritise a given set of requirements. For simplicity of understanding, we

provide an additional column indicating the number of requirements prioritised per minute by the particular method. ReproTizer, SMT and IGA were found to be the top three best performing requirements prioritisation methods with regards to the time dimension.

Table 3.9 Requirements prioritisation time dimension outcomes

Study	Number of requirements	Measurement	Method	Metric (minutes)	Requirements prioritised per minute
Voola and Babu (2013)	20	Average time (minutes)	NA AHP ENA	11.0 36.0 7.0	1.8 0.6 2.9
Bebensee et al. (2010)	114	Average time (minutes)	BPL Weiger	25.0 87.5	4.6 1.3
Achimugu et al. (2016)	1820	Average time (milliseconds)	ReproTizer	0.3	7280.0
Misaghian and Motameni (2016)	11	Average time (seconds)	Tensor AHP	5.7 311.0	2.0 <0.1
McZara et al. (2015)	100	Average time (minutes)	SNIPR WSM	41.4 51.3	2.4 2.0
Yutao Ma et al. (2012)	34	Total time (minutes)	Hybrid AHP Bubblesort	82.0 2083.0 1074.0	0.4 <0.1 <0.1
Nidhra et al. (2012)	40	Average time (minutes)	AHP NAcAHP	326.5 272.3	0.1 0.2
Palma et al. (2011)	109	Average time (minutes)	SMT IGA IAHP	0.8 0.8 14.2	145.3 132.9 7.7

It is to be noted that the information presented in Tables 3.6 - 3.9 is a summary of the relevant data presented in the studies mentioned in the respective tables. As the studies mentioned in the specific table (e.g., Table 3.9) follow different experimental settings (e.g., research methodology, data for experimentation or validation criteria and procedures) we are not performing any comparison analysis.

Furthermore, two noteworthy studies focused on the *requirements updates dimension*. Asghar et al. (2013) observed that their proposed requirements prioritisation method was capable of generating updated priorities of the same set of requirements according to the evolving software architecture of the system. Achimugu et al. (2016) have developed 'ReproTizer' which computes new priorities of requirements when a particular requirement or a stakeholder's preference is excluded from or included in the system.

Next, we observed that three studies investigated the *scalability dimension* of a particular requirements prioritisation method. Nidhra et al. (2012) have compared the scalability of NAcAHP with AHP and found out that the former method was more scalable than the later one as it reduced the time complexity

involved in performing pairwise comparisons. Achimugu et al. (2016) proposed requirements prioritisation method is claimed to accommodate new requirements at runtime. Elsood et al. (2014) through the means of operational cycles (i.e., number of iterations required for prioritising requirements) found out that their proposed goal based requirements prioritisation method was more scalable than AHP.

With regards to the *computational complexity dimension*, Bajaj and Arora (2013) have utilised the Big - O notation to investigate the computational complexity of the different stages of their proposed requirements prioritisation method. Thakurta (2013) through means of Big-O notation found out that AHP suffered from scalability issues as the number of requirements to prioritise increased whereas their proposed requirements prioritisation method (i.e., quantitative framework) was found to be linear.

Finally, we report the findings related to the relationship between the attributes and performance outcomes. Due to the subjective nature of some of the evaluations performed by the researchers (e.g., scalability) and the few studies under certain dimensions (e.g., requirements updates), we were able to include only two dimensions (i.e., accuracy and time) in the statistical significance analysis. We performed the Spearman correlation test to examine the relationship between the number of requirements and the accuracy of the requirements prioritisation method (refer to Table 3.6) as we had the appropriate number of studies from accuracy and time dimensions to perform the test (Myers & Sirois, 2004). The correlation coefficient was found to be 0.1 (p-value < 0.05) indicating that the accuracy of the requirements prioritisation methods increased as the number of requirements to prioritise increased. It is to be noted that the correlation reported is weak but it is statistically significant. Next, we examined the correlation between the number of requirements and the time taken by the requirements prioritisation methods to prioritise requirements. We recorded a weak statistically significant correlation coefficient of -0.27 indicating an inverse relationship (i.e., time required for prioritisation decreased with the increase in number of requirements). In addition, on average, requirements prioritisation methods researched in education discipline required less time to prioritise requirements than the methods from the software engineering and product manufacturing disciplines (average time: education = 18 minutes, software engineering = 303 minutes and product manufacturing = 158 minutes).

We discuss the results of the undertaken systematic mapping study on requirements prioritisation and the considerations of their implications for theory and practice in the Discussion section (refer to section 3.7). In the next section, we present the remaining overarching RQs.

3.6 Remaining Overarching RQs

We present the remaining overarching RQs of the subsequent phases (i.e., 2-4) in this sub-section as these phases are inspired and influenced from the outcomes of the systematic mapping study. We provide a detailed elaboration of the motivation for the same in this sub-section.

As not all the reviews of an app logged by its end-users on app distribution platforms are useful reviews, we had to investigate a filtering approach that identified and extracted useful reviews to prevent the performance (i.e., accuracy and time) of the particular prioritisation method from being hampered by the presence of non-useful reviews (Achimugu et al., 2014b; Maalej et al., 2016a; Panichella et al., 2015). The filtering approach came into consideration and the idea towards a filtering approach was inspired by several requirements elicitation methods and the requirements prioritisation method proposed by Peng et al. (2012) that acted upon a set of elicited requirements made available by domain experts and such methods suggested the avoidance of information that did not reflect stakeholders requirements to generate reliable prioritisation results (Garg et al., 2017; Thew & Sutcliffe, 2017; Zowghi & Coulin, 2005). Moreover, app developers are always on the lookout for efficient and automated information retrieval approaches that are able to filter (or extract) useful reviews logged about their apps given the vast amount of reviews that are provided online (Maalej et al., 2016a). The knowledge obtained from the useful reviews significantly assists the app developers in their software quality evaluations, and software maintenance and evolution cycles (Ghose & Ipeirotis, 2011; Maalej et al., 2016a). However, as online apps distribution platforms hold numerous reviews which are open to public access, manually extracting these useful reviews from a vast pool of numerous reviews is potentially challenging as it would be an error-prone and arduous task for the app developers. Such situation demands a reliable approach to filter useful reviews. This leads towards the next RQ which is

RQ2. How can useful reviews be filtered?

The objective of RQ2 is to identify an approach that will allow us to filter useful reviews for classification or prioritisation purpose. After developing a filtering approach to distinguish useful reviews from non-useful ones and extract the useful reviews from a vast pool of reviews, we had to develop a method that could prioritise the numerous useful reviews for remedial actions to support the app's maintenance and evolution cycles. Among the empirical studies reviewed during the systematic mapping study of requirements prioritisation, the requirements prioritisation method proposed by Peng et al. (2012) targeted the highest number of requirements (i.e., total - 1878) for prioritisation, and thus assured its scalability. Scalability is of prime importance in this study as the prioritisation of numerous useful reviews is the aim of the work. Moreover, the authors' requirements prioritisation method was a hybrid method i.e., it combined several methods for requirements prioritisation purpose and through means of the systematic mapping study on requirements prioritisation we observed that researchers

often developed hybrid methods as these methods have shown more promise towards generating reliable and efficient requirements prioritisation solutions than the other types of methods (i.e., single methods or multiple methods) (Abou-Elseoud et al., 2016; Achimugu & Selamat, 2015; Santos et al., 2016; Yutao Ma et al., 2012). This proposed hybrid method initially classified a set of elicited requirements into manually predefined groups of interest using the domain knowledge made available by domain experts. Later, using a combination of methods such as SemanticVOC, Domain Semantic Model, SELRank algorithm and a query processing module along with the priority preferences of the stakeholders, the method generated the priorities of the predefined groups of interest. This method provided us the inspiration (i.e., following the steps of this method) to come up with the three steps in our proposed prioritisation approach i.e., filtering useful reviews (Phase 2), classification of the useful reviews into groups of interest (Phase 3) and prioritisation of useful reviews and the groups using a hybrid method (Phase 4). However, all the empirical studies identified and reviewed via the systematic mapping study presented requirements prioritisation methods whose designs and developments were based on the availability of domain knowledge or priority preferences of the stakeholders. For instance, Ninaus (2012) utilised a specific heuristic method which with the support of varied priority preferences of the stakeholders operated on the same set of requirements to generate compatible priorities of the requirements (i.e., converting the dissimilar priority preferences into universal requirements' priorities). Franceschini et al. (2015) have used domain knowledge made available from domain experts to develop and accordingly customise QFD method for prioritising the requirements of a pencil product, trekking products and office products based on the priority preferences of the stakeholders. Hence, for prioritising useful reviews we cannot directly adapt such methods or inherit guidelines from them to develop our hybrid prioritisation method because: 1) millions of apps hosted on app distribution platforms belong to a wide spectrum of domains (e.g., games, entertainment, education, tools, communication, music, shopping, travel and so on), hence it is not practically possible to contact the app developers (i.e., domain experts) of these apps to gather and store the boundless domain knowledge required for prioritisation (or classification). 2) Moreover, it is practically impossible to request the priority preferences on useful reviews from the countless and geographically scattered end-users (i.e., presiding stakeholders) of the apps (Pagano & Maalej, 2013; Sorbo et al., 2016). In addition, it would be a challenging and intricate task to handle any missing priority preferences or resolve any conflicts related to different priority preferences on the same set of useful reviews to achieve consensus. Furthermore, the application of requirements prioritisation methods is confined to requirements whereas useful reviews are an extension of requirements as they contain bugs or enhancements along with the requests for features logged by the end-users (Maalej et al., 2016a; Panichella et al., 2015).Therefore, these reasons point towards the development of an automated prioritisation method that is not dependent on the availability of domain knowledge and is independent of the priority preferences of the end-users (i.e., end-users who are not available to provide priority preferences).

Before suitable prioritisation methods can be developed, we had to figure out an approach to automatically classify useful reviews into specific groups of interest as we had taken inspiration from the requirements prioritisation method proposed by Peng et al. (2012) which initially classified requirements into predefined groups of interest based on domain knowledge made available by domain experts. This leads to the RQ which is

RQ3. How can the useful reviews be classified into groups of interest?

After achieving the outcome of classifying useful reviews into specific groups of interests, our final objective was to prioritise the useful reviews and their associated groups which leads towards the final RQ of this study, that is

RQ4. How can an automated prioritisation method be developed to prioritise numerous useful reviews?

It is to be noted that the requirements prioritisation method proposed by Peng et al. (2012) generates only the priorities of the predefined groups based on the priority preferences of the individual requirements assigned by the stakeholders whereas in our study we have proposed to automatically generate the priorities of the useful reviews as well as the groups in which the useful reviews are classified into (i.e., our work proposes two prioritisation methods – one for grouped useful reviews and the other for individual useful reviews).

That said, it is to be noted that considering the prime objective of this undertaken research is the prioritisation of numerous useful reviews, we followed the pilot study approach in the relevant phases of this research as a pilot study allows to perform preliminary investigations and experiments which aim to validate the feasibility of a proposed approach (e.g., approach to filter useful reviews, classify useful reviews based on an automatically generated taxonomy, and so on) and steers the research in the right direction through the outcomes of the pilot studies while optimising the utilisation of scarce resources (e.g., time, human evaluators, research funds, and so on) associated with the undertaken research (Allan et al., 1998; Thabane et al., 2010). Thus, the primary objective of these pilot studies was to examine the feasibility of the proposed approaches (e.g., filtering of useful reviews, automated taxonomy generation and so on) and quantifying evaluation of the outcomes generated by those approaches in the respective phases. This was due to the time and human resource constraints that were associated with the development of approaches and evaluations of the outcomes generated in the subsequent phases. Hence, based on the outcomes of the pilot studies from phase 2, 3 and 4 we were able to conduct a full-scale study on the prioritisation of numerous useful reviews in phase 4.

In the next section, we highlight the discussions related to the systematic mapping study on requirements prioritisation.

3.7 Discussion

By following the systematic mapping study process proposed by Petersen et al. (2008) we were able to derive relevant RQs (i.e., RQ1.1 to RQ1.6) piloting the systematic mapping study on requirements prioritisation. We developed several classification schemes to appropriately organise the studies on requirements prioritisation. The analysis of the findings primarily focused on answering the RQs. From a holistic viewpoint, the conducted systematic mapping on requirements prioritisation in phase 1 provided a structure (i.e., roadmap to investigate a field of interest) that assisted us in identifying the type of research studies (i.e., proposed solution, evaluated solution, simulated solution, taxonomy, opinion/philosophy, hybrid method, secondary evaluation/categorisation, experience, single method, multiple methods, tool, and process) that have been published and classify those studies into suitable categories based on the relevant classification scheme. Because of this, we could generate visual summaries of the findings reported in the Results section (refer to section 3.5), thereby providing an overview of the comprehensive findings. Furthermore, as we were able to conduct the systematic mapping study on requirements prioritisation we were able to get an overview of the requirements prioritisation field, and along with this, filter empirical studies of this topic which were then reviewed in detail. The subsections below discuss the results and implications of RQ1.1 to RQ1.6.

3.7.1 RQ1.1 What has been the interest in requirements prioritisation over time, what are the different publication venues and what are the various disciplines in which the application of requirements prioritisation exist?

Results in the previous chapter reveal that there has been growing interest in requirements prioritisation over the years, with the highest interest observed for 2017. Most studies were found to be published in conference and journal venues. Such findings potentially point to the fact that requirements prioritisation is gaining the attention of the scientific community, with studies addressing the particular requirements prioritisation problem encountered by product developers. Beyond journal and conference venues, a breadth of requirements prioritisation studies across other venues is observed. That said, an interesting observation is that the proportion of journals to conferences in the product manufacturing discipline is higher (0.67) than that of software engineering discipline (0.39). This may be because of discipline specific publication norms (e.g., a larger number of publications in the software engineering discipline appear in conferences when compared to other disciplines). In addition, we performed an analysis of the publication locations of the 211 studies which shows that the studies have been contributed by researchers from several countries across the world. Figure 18 shows a heat map where the intensity of the colour corresponds to the frequency of publications presented on the colour scale. Looking at the top 10 countries, the majority of the publications were from India (43 studies), followed by USA (32 studies), Malaysia (24 studies), Italy (18 studies), Sweden (17 studies), Pakistan (11 studies), Netherlands (11 studies), Germany (10 studies), China (9 studies), and Brazil (7 studies).

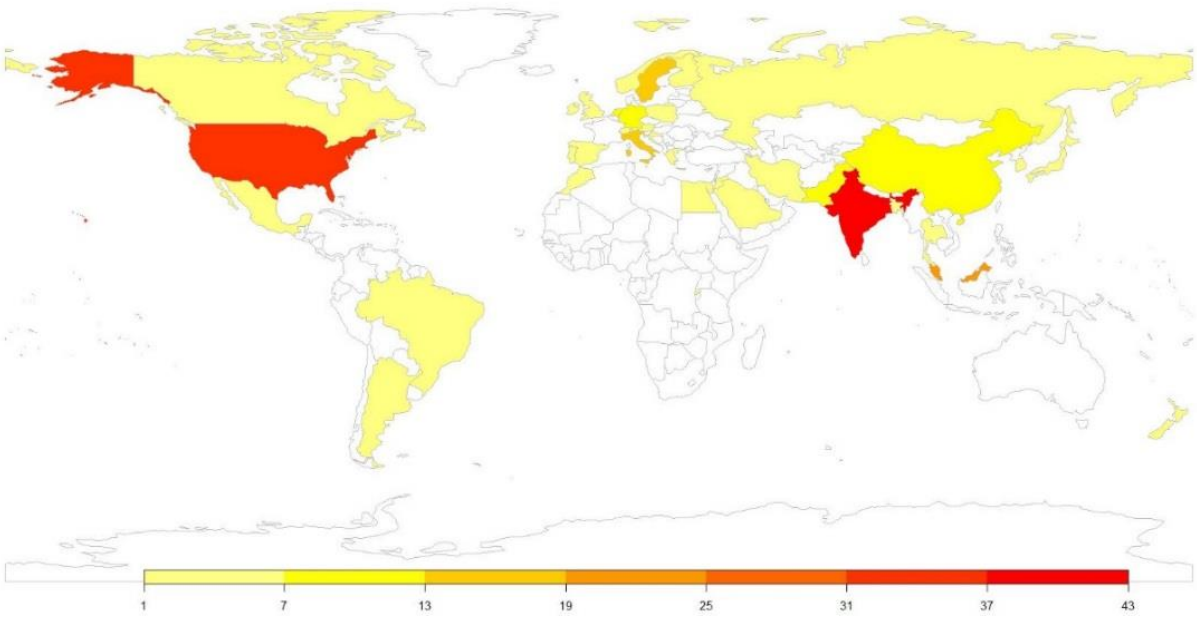


Figure 18. Representation of requirements prioritisation publications on the world map

While the increase in studies on requirements prioritisation is evident in the software engineering discipline over the years, there was no initiative observed towards understanding how the requirements prioritisation problem is addressed in other disciplines or vice-versa. This may be due to studies from particular disciplines are not being considered or missing out on the repository of knowledge evident in other disciplines. For instance, from the product manufacturing discipline, Zhaoling et al. (2009) have utilised QFD integrated with grey relational analysis to examine the relationships between stakeholders' preferences and product engineering characteristics. In this study, the authors have investigated the application of the weighted average method to resolve any conflicts related to the priority preferences of stakeholders. In another study, Nepal et al. (2010) have evaluated the fuzzy analytical hierarchical process to prioritise the requirements pertaining to an automobile. The proposed prioritisation method considers customer satisfaction attributes along with the priority preferences of stakeholders while prioritising the requirements which in turn assists the automotive company to refine their vehicle design and its performance. The method developed by these authors initially identifies the aspects that influence the decision making process of the automobile's requirements engineering phase, in promoting an organised view of requirements' priorities. Fuzzy logic is used to compute the priorities of the requirements which are expressed in a hierarchical representation. Overall, requirements prioritisation methods developed in other disciplines have the potential to resolve stakeholders' conflicts on priority preferences of requirements, address dependencies among requirements, handle updated priorities of requirements and enhance the outcome of requirements prioritisation (Li et al., 2012; Nepal et al., 2010; Zhaoling et al., 2009). Therefore, such requirements prioritisation methods may be of utility to researchers working on the requirements prioritisation problem in the software engineering discipline.

3.7.2 RQ1.2. What approaches have been used to study requirements prioritisation?

The findings in this work reveal that the majority of the requirements prioritisation studies targeted empirically evaluated solutions (43%). These findings are appropriate for the requirements prioritisation field, as the particular requirements prioritisation problem needs to be addressed with continuous experimentation and different types of case studies. That said, we observed the presence of proposed solutions that were not evaluated, or at times such solutions existed only in some form of simulation. Moreover, fewer studies have highlighted opinions and gathered evidence around various requirements prioritisation solutions that are provided through secondary evaluation/categorisation. While secondary evaluation/categorisation is performed, such evaluations have focused on analysing requirements prioritisation approaches in the software engineering discipline and especially targeted secondary studies. For instance, Aasem et al. (2010) have published a secondary study which emphasises on the significance of requirements prioritisation towards launching essential software updates, and the merits and limitations of AHP, B-Tree, CV, Ranking, Top 10, NA, CV, and PG requirements prioritisation methods. Garg et al. (2017) states that the identification of requirements through means of a reliable requirements elicitation method is crucial as the outcome of prioritisation is dependent of the requirements elicitation process. This study also examines advancements related to requirements elicitation and prioritisation. In another study, Fadhl Hujainah et al. (2016) has mentioned that complexity, time, value, accuracy, risk, importance and benefit are crucial factors that drive the requirements prioritisation process. The author has evaluated several requirements prioritisation methods such as AHP, NA, Top Ten, Ranking, Priority Groups, CV, Hierarchical AHP, Planning Game, B-Tree, Minimal Spanning Tree, Benefit and Cost Prediction, PHandler, Case Based Ranking, Requirements Uncertainty Prioritisation Approach, Evolve, SERUM, Cost Benefit and Pairwise Comparison. That said, Fadzir et al. (2016) have provided a systematic literature review on requirements prioritisation practices evident in the software engineering discipline.

Hence, our results suggest there is a need for studies that perform a more comprehensive investigation of the evidence and proposed solutions on the requirements prioritisation problem existing across different disciplines. Such evidence would probably inform the efforts directed towards developing the relevant requirements prioritisation solutions for software developers and particularly those addressing numerous requirements. For instance, Laurent et al. (2007) from the software engineering discipline have proposed a requirements prioritisation method that uses a classification approach to categorise requirements having similar characteristics into classes such as business goals, non-functional requirements, functional requirements and so on. Later, the requirements are prioritised with the assistance of priority preferences provided by the stakeholders and these priority preferences are utilised as weights to perform prioritisation. Such requirements prioritisation methods have been informally claimed to be scalable. That said, these authors can take inspiration from studies from other disciplines that provide effective mechanisms to generate reliable stakeholders' preferences for prioritisation

purpose (Nepal et al., 2010; Zhaoling et al., 2009). On the contrary, the studies from other disciplines can seek inspiration from studies proposed by Laurent et al. (2007) for addressing scalability issues associated with prioritisation.

Interestingly, beyond the software engineering discipline, other disciplines have not performed secondary evaluation/categorisation towards developing a repository of methods and evidences around requirements prioritisation. That said, some studies reflected authors' opinions and experiences. Secondary evaluation/categorisation studies are essential for providing the key concepts of the field of interest, identifying different research trends, uncovering challenges and exploring the solutions proposed to address challenges in a discipline. In certain cases, secondary evaluation/categorisation studies classify details of the primary studies into categories of interest for meaningful interpretation, providing a plethora of evidence around a field. While the lack of studies reflecting secondary evaluation/categorisation in other disciplines besides software engineering demands attention of researchers, several contributions provided by these disciplines are noteworthy. This aspect is discussed further in the next sub-section.

3.7.3 RQ1.3 What form did the contributions of the requirements prioritisation studies take?

Overall, a wide spread of requirements prioritisation contributions is observed, ranging from taxonomies to tools. While some studies investigated multiple methods, these methods were evaluated for their individual merits and demerits. That said, hybrid methods potentially harness the strengths of multiple methods and attempt to avoid their weaknesses. While hybrid methods are an amalgamation of several requirements prioritisation methods, there has only been a small-scale effort (16%) observed towards a systematic evaluation of the single methods in view of developing reliable hybrid methods. In fact, around 63% of the hybrid methods have undergone empirical evaluations. This was one of the inspirations that lead to the development of our proposed prioritisation methods (group-based and individual) that are hybrid. That is, multiple prioritisation methods are incorporated as variables of a multi-criteria heuristic function. Moreover, other researchers could also pursue such undertakings related to hybrid methods in developing well-founded requirements prioritisation solutions.

For instance, the advantages and disadvantages of AHP have been briefly examined by Nidhra et al. (2012) to develop a requirements prioritisation method which includes all the strengths of AHP but overcomes its weaknesses. To achieve this, the authors have combined NA with AHP and termed it as 'NAcAHP'. The NA method first classifies each requirement into groups and later AHP prioritises the requirements present in those groups. The performance of 'NAcAHP' was compared with AHP in terms of time. A set of forty requirements were prioritised by both methods with results showing 'NAcAHP' to be faster than AHP. In another study, Abou-Elseoud et al. (2016) have merged AHP, CV and QFD

to develop a new requirements prioritisation method named ‘EHRP’. Initially, the method generates three levels of hierarchical nodes (i.e., Goal, Criteria and Requirements). This generation approach is similar to AHP, however, the method utilises two non-identical prioritisation pathways instead of a matrix-based approach to prioritise requirements. CV is used to rank requirements residing at the lower levels of the hierarchy while QFD operates at the last level to generate the final priorities of the requirements. This hybrid method was empirically evaluated in an enterprise resource planning organisation to prioritise a set of small requirements where the method was validated to be efficient for prioritisation purpose.

Similarly, Garg and Singhal (2017) have come up with an approach that establishes a relation between functional requirements and non-functional requirements. These functional requirements are prioritised on the basis of their degree of relationship with the non-functional requirements. The authors were able to achieve this by merging three requirements prioritisation methods together, which were the cost-value approach, NA and matrix multiplication. The cost-value approach enables the pairwise comparison of stakeholders' preferences on non-functional requirements. This output is then used as input to the NA method, where values are assigned to functional requirements based on the operational outcome of the pairwise comparisons. The final priorities of the requirements are then calculated by matrix multiplication. This approach was used to prioritise the requirements of an article publishing software where it was assessed to be effective. In another study, Kamvysi et al. (2014) have improved the performance of the QFD method in prioritising the requirements of students belonging to an educational institute. They were able to achieve this outcome by customising the internal structure of the QFD method. Fuzzy logic and linear programming concepts were integrated with the QFD method. It was claimed that the utilisation of fuzzy logic resolved the issue of incomplete, vague or conflicting priority preferences that were provided by the students. The utilised fuzzy linear programming approach operated on the priority preferences to generate the essential weights (i.e., compatible students’ priority preferences) required by the QFD method to generate the prioritised list of requirements. This hybrid method successfully prioritised the requirements of students which assisted the educational institute in updating teaching objectives and techniques according to the requirements of students. Sensitivity analysis was used to determine the accuracy of the method which showed that this hybrid method performed better than the traditional requirements prioritisation methods.

Furthermore, the outcomes of the undertaken systematic mapping study are assessed in relation to the discipline of enquiry, and specifically in terms of hybrid methods in other disciplines apart from software engineering. In this regard, we found that other disciplines like product manufacturing have also developed hybrid requirements prioritisation methods that are novel. For instance, Fung et al. (1996) have combined AHP and QFD to prioritise requirements to improve the design of a product. Armacost et al. (1994) have also merged QFD and AHP to capture and prioritise the requirements of

the stakeholders. Such hybrid methods are particularly suitable for supporting stakeholders' multi-criteria decision making process related to requirements prioritisation. In addition, we observed that specific methods tend to gain attention across the identified disciplines. This aspect is examined further in the following sub-section.

3.7.4 RQ1.4 What prioritisation methods have been studied or developed?

We were able to identify 157 requirements prioritisation methods that were researched by those investigating a particular requirements prioritisation problem. We noticed an interest among the researchers to propose new requirements prioritisation solutions or perform replication studies. AHP, CV, QFD, NA and PG were the top five prominent requirements prioritisation methods that were given the most attention by researchers. These methods were involved in different types of research, ranging from taxonomies, processes to single methods. The merging of these methods to form hybrid requirements prioritisation methods is noteworthy as evidence points to the fact that many single methods do not perform satisfactorily on their own (Abou-Elseoud et al., 2016; Achimugu & Selamat, 2015; Sadiq et al., 2017; Yutao Ma et al., 2012). Moreover, we observed that 26 hybrid requirements prioritisation methods were developed as variations of the top ten requirements prioritisation methods (refer to Results sub-section 3.5.4). That said, many of these methods were found to be evaluated on a small number of requirements, and these evaluations involved real world requirements prioritisation problems. However, researchers of these methods often encountered scalability issues or computational complexity challenges (Berander & Jonssen, 2006; Thakurta, 2013). From a discipline perspective, we observed that out of the twelve evaluated hybrid methods, seven were from software engineering, four were from product manufacturing and one was from the real estate discipline. Among these, hybrid variants of AHP dominated the entries. However, the hybrid variants of AHP prioritised only a few requirements and were not capable of handling dependencies among the requirements or requirements updates. Thus, such approaches were found to be non-scalable. Nonetheless, this method (i.e., AHP) was researched often as researchers aim to generate accurate prioritisation solutions for a small number of requirements. That said, there exists an opportunity to investigate and experiment with the combination of other single methods to validate their utility as hybrid methods.

Interestingly, after reviewing the studies across all the disciplines that highlighted tools and taxonomies, we found out that only one study from the software engineering discipline indicated that the QFD method was operationalised in the form of a tool. Similarly, only one study from the product manufacturing discipline indicated that AHP was contributed as a tool. However, AHP, CV, NA, PG, HCV, ranking, and priority groups were commonly examined as a part of taxonomies. This shows that researchers often tend to conduct reviews or empirical studies of the same methods rather than developing new hybrid methods, which may be derived from other methods.

3.7.5 RQ1.5 What are the dimensions that were evaluated for requirements prioritisation methods?

Our analysis of the dimensions that were evaluated for requirements prioritisation methods revealed that the majority of the studies have investigated the methods from a single dimension (e.g., time). Some dimensions (e.g., requirements update) are often neglected when methods are developed and evaluated in favour of the operational demonstration dimension. Accuracy is often utilised by researchers for validating correctness of the outcome (i.e., list of prioritised requirements) generated by a requirements prioritisation method (Achimugu et al., 2016; Asghar et al., 2013; Bebensee et al., 2010). Such studies ascertain stakeholders' acceptability of the priorities generated by a requirements prioritisation method. Accordingly, this dimension is important in determining the effectiveness of a requirements prioritisation method. In addition, as a product is developed to satisfy the requirements of its stakeholders, their preferences (i.e., priorities of requirements) act as the ground truth for evaluating the outcomes of a particular requirements prioritisation method. Hence, this aspect is considered in our work wherein we utilise stakeholders' priority preferences to validate the accuracy of the group-based and individual prioritisation methods.

Next, we noticed that the scalability dimension was utilised by researchers to check for the accommodation of new requirements during the requirements prioritisation process. In general, scalability in requirements prioritisation points towards the ability of a requirements prioritisation method to accommodate a large number of requirements before performing the prioritisation (Achimugu et al., 2016). Furthermore, computational complexity considered by researchers quantifies the performance of the requirements prioritisation methods in terms of their space and time complexity. Studies covering this dimension show that researchers investigated the computational complexity of the particular requirements prioritisation method to record the requirements prioritisation method's execution time and memory utilisation (Thakurta, 2013). The computed computational complexity may assist researchers to optimise the particular requirements prioritisation method for delivering efficient performance. However, only three out of the shortlisted seventy-six empirical studies covered the computational complexity dimension. This suggests that computational complexity is not often considered to address a requirements prioritisation challenge. This aspect needs the attention of researchers given that computational complexity influences other dimensions such as time or accuracy (Voola & Babu, 2017).

As a product evolves after it is launched in the market, its requirements are often subjected to change (Oliveira & Almeida, 2015). This is particularly evident when certain stakeholders specify changes that may lead to a change in priorities around requirements or requirements themselves (e.g., the marketplace to sell or buy products on Facebook app was not initially part of the app's features). Therefore, it is beneficial to consider prioritisation mechanisms that accommodate the evolving

requirements of a product. That said, only six studies covered the requirements update dimension while thirteen studies recorded the time required for prioritising a set of requirements. The inclusion of the time dimension in a requirements prioritisation study is important when the study is related to enterprises targeting the release of new versions of their products within limited time intervals. This is because as the amount of time the particular prioritisation method takes for prioritisation should be less than that of the release date of the new version since there should be enough time for the developers to address the top concerns that are recommended by the prioritisation methods (Oliveira & Almeida, 2015). Hence, we have included this dimension in our study.

Finally, we reviewed studies that have covered the requirements dependencies dimension with the intent of considering the discovered knowledge of dependencies among requirements or addressing the dependencies among the requirements in the process of prioritising those requirements. Such studies provide an understanding around the way requirements are nested and how such nesting influences product engineering processes such as impact analysis, planning, design, development, testing, and so on (Li et al., 2012). The studies covering the requirements dependencies dimension uncovered and visualised dependencies using graph-based mechanisms. The outcomes of the requirements dependency analysis are challenging to evaluate given that stakeholders often agree to an outcome, whereas in some scenarios the requirements dependencies are discovered by a mechanism and the stakeholders may have limited prior knowledge of the dependencies to evaluate its correctness. Thus, there exists an opportunity to develop new mechanisms to fulfil this objective where the effectiveness of dependency mechanisms could be evaluated.

3.7.6 RQ1.6 What are the performance outcome of the evaluations, and is there evidence of relationships between attributes of requirements prioritisation methods and their performance outcomes?

We observed that the majority of the studies that included the accuracy dimension had used stakeholders' preferences to validate the accuracy of the requirements prioritisation methods. This assisted researchers to compare the priorities of the requirements generated by the particular requirements prioritisation method against those provided by the stakeholders to evaluate the method's accuracy. Such practice of evaluating accuracy highlights the significance of stakeholders' preferences in the requirements prioritisation process, and indicates that requirements prioritisation methods are developed towards the prioritisation driven by the preferences of stakeholders. This highlights the importance of addressing the requirements in the order preferred by the stakeholders. Accuracy measures in the results ranged from 16% to 99%, with ReproTizer tool reported to be most accurate while accommodating the stakeholders' preferences (Achimugu et al., 2016). The overhead involved in getting priority preferences from stakeholders and resolving any conflicts among them is challenging

and thus it is conclusive that studies measuring the accuracy of a requirements prioritisation method tend to operate on limited set of requirements (Asghar et al., 2013; Bebensee et al., 2010; Sadiq, 2017).

With regards to the studies that dealt with the handling of stakeholders' preferences, we noticed that researchers had encountered two major challenges. The first challenge was the stakeholders' preferences were often incomplete or vague, thus making them unsuitable for prioritisation (Inoki et al., 2014; Voola & Babu, 2013). The second challenge being that the stakeholders had different priority preferences for the same set of requirements (Bajaj & Arora, 2013; Zhaoling et al., 2009). Initially, the conflicting preferences had to be transformed into a compatible priority preferences to satisfy the stakeholders when the requirements prioritisation method was initialised. Fuzzy logic was found to be an effective solution to achieve this, providing outcomes that were suitable in terms of generating a prioritised list of requirements that encompassed conflicts, vagueness or uncertainty (Achimugu et al., 2014; Achimugu et al., 2014b; Bajaj & Arora, 2013). That said, when compared to the accuracy dimension, it is difficult to evaluate the utility of the outcomes of the stakeholders' preferences dimension as there was no objective measure reported that could prove the correctness of processing the stakeholders' preferences. Similar conclusions were drawn after reviewing the empirical studies that focussed on the scalability dimension, where it was observed that the researchers have developed their own practices to assess the scalability of the requirements prioritisation methods (Achimugu et al., 2016; Nidhra et al., 2012). For instance, Achimugu et al. (2016) have measured scalability in terms of the number of requirements their requirements prioritisation tool (ReproTizer) could accommodate at runtime. Whereas, Nidhra et al. (2012) have utilised the time dimension to informally determine the scalability of the particular requirements prioritisation methods. Such practices are dissimilar in terms of how scalability was assessed, thus pointing towards the need for the establishment of a common practice to assess the scalability of requirements prioritisation method. That said, in our work we term the empirical requirements prioritisation method handling the highest number of requirements for prioritisation purpose as most scalable Peng et al. (2012). This is based on the ability of the requirements prioritisation method to handle the highest number of requirements at runtime (refer to results section, Table 3.8).

Furthermore, the Big-O notation was most favoured by researchers for computing the computational complexity of the requirements prioritisation methods. This shows that the performance of any module (component) of a requirements prioritisation method or the method itself can be inspected and the results of such an inspection could prove beneficial towards analysing or fixing flaws or to further optimise the method for better performance. This is clearly evident in the work of Thakurta (2013), where it was proved that AHP suffered scalability challenges as the number of requirements to prioritise increased linearly.

Only two studies out of the six covering the requirements updates dimension proposed mechanisms that allowed the particular requirements prioritisation method to handle dynamically changing priorities or incorporate new requirements during the requirements prioritisation process. These two studies may inspire the implementation of a requirements prioritisation method that adapts to evolving requirements. Another dimension that was covered in studies was requirements dependency where the proposed solutions varied. For instance, Peng et al. (2012) have performed ontology analysis to uncover dependencies among the classified textual requirements. Other studies that covered the requirements dependency dimension used matrix or graph-based mechanisms (refer to results section, Table 3.8). Such mechanism holds promise for discovering dependencies among large-scale requirements, especially those that are crowdsourced.

With regards to the studies covering the time dimension, we observed that the time required for a particular requirements prioritisation method to prioritise different sets of requirements was nonlinear. For instance, several studies utilised AHP, and the time required for AHP to prioritise a given set of requirements differed (Misaghian & Motameni, 2016; Nidhra et al., 2012; Voola & Babu, 2013; Yutao Ma et al., 2012). This was because the dimensions accuracy, scalability and stakeholders' preferences influenced the total time required for AHP to prioritise the requirements. This suggests that the requirements prioritisation process tends to influence the outcome measures (e.g., a customisation of AHP that makes it accurate may make the solution slower, or vice-versa). This in turn could affect the perception of the dimensions' utilisation, effectiveness, and outcomes. We further consider this issue in sub-section 3.7.7.

Finally, we examined the attributes of requirements prioritisation methods and their performance outcomes. As noted in the Results sub-section 3.5.6, due to the limited number of studies under certain dimensions such as scalability, requirements updates, requirements dependency and stakeholders' preferences, and the subjective nature of some of the dimensions (e.g., stakeholders' preferences), we were able to include only accuracy and time dimensions for statistical analysis. The statistical analysis results show that the number of requirements affect the accuracy that was reported marginally. A more detailed review of the studies covering the accuracy dimension shows that the accuracy of the requirements prioritisation method was dependent on the complexity of the requirements that were prioritised, along with the structure and operating mechanism of the particular requirements prioritisation method (Misaghian & Motameni, 2016; Nidhra et al., 2012; Palma et al., 2011; Voola & Babu, 2013). In addition, over the years, it was found that requirements prioritisation methods require less time to prioritise increasing number of requirements (Achimugu et al., 2016; Misaghian & Motameni, 2016; Voola & Babu, 2013). Such findings are central to the process of scoping a specific requirements prioritisation problem and developing a problem specific requirements prioritisation

solution. Moreover, these findings may inform researchers in terms of optimising the requirements prioritisation process to produce efficient methods and obtain beneficial results.

3.7.7 Summary of the way evaluated requirements prioritisation dimensions influence each other

In this sub-section, we summarise the dimensions that were studied by those that have examined requirements prioritisation to understand how they influence each other. These connections were revealed after an investigation of the empirical studies in our sample. For instance, Yutao Ma et al. (2012) study shows that as requirements are subjected to updates, dependencies among old and new requirements are constantly affected. In another study, McZara et al. (2015) convey that resolving vague preferences of stakeholders affect the accuracy of requirements prioritisation methods and this ultimately affected the time that is required for completing the requirements prioritisation process. Although we are not able to precisely measure the degree of influence the various evaluation dimensions have on each other, we review certain observed relationships below.

Firstly, multiple studies show that the time dimension is influenced by all the other dimensions (Asghar et al., 2017; Kukreja et al., 2012; Voola & Babu, 2013; Yutao Ma et al., 2012). Therefore, it can be inferred that the overall time required for the requirements prioritisation process changes as researchers add other dimensions (e.g., requirements dependency) for evaluation in their study, or as the nature of the requirements prioritisation problem and the application of the requirements prioritisation method varies (Nidhra et al., 2012; Voola & Babu, 2013; Yutao Ma et al., 2012). While only one study investigated the accuracy of requirements prioritisation method along with its computational complexity, this investigation allowed the researchers to understand how accuracy and computational complexity were interrelated (Voola & Babu, 2017). Similarly, stakeholders' preferences are known to play a major role in influencing the accuracy of a requirements prioritisation method, as in many scenarios it is used as the baseline for validating requirements prioritisation outcomes. In fact, given that stakeholders ultimately assess the outcomes of requirements prioritisation methods, this dimension also affect requirements updates, scalability and computational complexity dimensions (Achimugu et al., 2014a; Bajaj & Arora, 2013; Berander & Svahnberg, 2009; Ninaus, 2012).

Furthermore, managing scalability is challenging, and this dimension has an impact on the computational complexity of a requirements prioritisation method (Bajaj & Arora, 2013; Kukreja et al., 2012). Interestingly, it was also discovered that requirements updates affected accuracy, scalability, requirements dependencies, and computational complexity (Achimugu et al., 2016; Asghar et al., 2013; Perini et al., 2013; Yutao Ma et al., 2012). This is primarily due to the overhead associated with an increasing number of requirements or their associated updates. Only one study revealed that requirements updates are dependent on stakeholders' preferences (Santos et al., 2016). That said,

requirements dependencies also impact scalability and computational complexity (Delia Ilie et al., 2009; Kukreja et al., 2012; Santos et al., 2016; Sharma, 2007). In addition, prior to evaluation, any empirical requirements prioritisation method is established to be operational. Such considerations are insightful for balancing the trade-offs in performance outcomes of requirements prioritisation methods.

3.8 Threats to Validity

In this section, we present the threats to validity that can potentially affect the outcomes reported in this systematic mapping study on requirements prioritisation.

The selection of studies in the systematic mapping study can be seen as a threat as we have considered studies which were published in English language, and thus, we might have missed pertinent studies documented in other languages. Subsequently, we have not targeted studies that were not peer-reviewed (e.g., technical reports, proposals) or reports (e.g., thesis) which may contain relevant details for answering the research questions RQ1.1 to RQ1.6. The approach used to develop the search keywords could potentially pose a threat. However, we have conducted substantial searches to understand the keywords that are used for identifying studies on requirements prioritisation (Broder, 2002; Lorigo et al., 2008). In this regard, we have followed the guidelines provided by Kitchenham (2007) for piloting search keywords that are likely to uncover a substantial number studies rather than miss out on the pertinent studies. Hence, although posing a threat with regards to the large number of studies that were returned for shortlisting, broader search keywords were utilised to reduce or avoid the threat of missing studies, as specific keywords covering narrow search could have resulted in missing certain studies. To address this, formal reliability checks were performed to ensure agreement on the excluded and included studies. Furthermore, beyond using the broad search keywords to address the threat related to missing of relevant papers for answering RQ1.1 to RQ1.6, systematic searches were conducted in eight prominent digital knowledge databases as recommended by Kitchenham (2007).

It is to be noted that, for the subsequent chapters, we have summarised the threats to validity associated with these Chapters into three relevant sub-sections: internal validity, external validity and construct validity. Internal validity reflects the confidence in results, and the factors that are attributed towards the results. In other words, internal validity rules out alternative explanations for a result. Due diligence towards minimising this threat involves the consideration of all possible factors associated with the suitable research activities to perform robust research. Such activities include developing a suitable research methodology, formulating appropriate research questions, extensive searches for literature, utilising the right protocol for performing literature searches, experiments, relevant rigorous assessments, standard reliability assessment procedures and so on that generate reliable research outcomes (Grafton et al., 2011). External validity highlights aspects related to the generalisability of

the outcomes and construct validity points towards the validity of the conclusions that are drawn from the results generated from the conducted experiments (Grafton et al., 2011).

That said, we provide the concluding remarks of this phase, its research contributions and summary of implications in the Conclusion chapter (refer to Chapter 7). In the next chapter, we present the details of Phase 2 (i.e., filtering of useful reviews)

4 Filtering of Useful Reviews

In this chapter, we present phase 2 of the undertaken research through a pilot study where we worked on an approach to filter useful reviews from a vast pool of reviews (RQ2) by answering RQ2.1 that dealt with the investigation of the performance of six Naïve Bayes variants by exploring their utility towards the automated filtering of useful reviews based on a set of rules that distinguished useful reviews from the non-useful ones.

4.1 Introduction

Manually identifying and extracting useful reviews from a vast pool of reviews is a challenging task as it requires high levels of cognitive load, time and effort from the app developers and this task may be compounded due to the presence of large numbers of non-useful reviews (Pagano & Maalej, 2013; Panichella et al., 2015). Moreover, as the group of app developers usually tend to be small, error proneness and lack of scalability may compromise the manual filtering task. Thus, in this phase we conducted a pilot study to review information retrieval studies in which the limitations of the filtering approaches proposed by these studies were observed (Fu et al., 2013; Keertipati et al., 2016). The most significant limitation was that the approaches often failed to filter most of the useful reviews. Furthermore, in studies from the software engineering disciplines it was found out that Multinomial Naïve Bayes method was proven to be most appropriate and reliable for automating the filtering approaches based on a set of predefined rules (i.e., classifying new information using previously classified information) (Caruana & Niculescu-Mizil, 2006; Wang et al., 2018). Therefore, we identified and empirically evaluated six variants of the Multinomial Naïve Bayes method and benchmarked their performances. We present the essential details associated with this phase for piloting and evaluating the proposed filtering approach to identify useful reviews in the next sub-sections.

4.2 Related Studies

Traditional filtering approaches cannot reliably filter useful reviews as they are unable to perform filtering based on the disambiguation of the information conveyed by the reviews (Pagano & Maalej, 2013). For instance, Licorish et al. (2017) have filtered reviews whose ratings were less than 3 and thus may have missed out on crucial reviews that had higher ratings that may reflect useful end-user feedback about improving an app. In another study, Fu et al. (2013) have used sentiment analysis to filter reviews having negative end-users' sentiments associated with them with the assumption that such reviews indicate app issues (bugs). Similarly, Shah et al. (2018) have evaluated the performance of BoW (Bag of Words) and CNN (Convolutional Neural Networks) towards the extraction of app features from reviews. It was reported by the authors that BoW performed better than CNN but suffered from overfitting of the learning data (Luo et al., 2014). These filtering approaches usually tend to miss some of the useful reviews or return non-useful reviews (Hoon et al., 2013). For instance, consider a review

that is filtered based on negative sentiment and lower rating filtering approach, '(a) Useless app, uninstalling it as it left me very disappointed ☹ !!', and another review discarded because of its higher rating '(b) Fantastic app and works well but has a small problem with screen resolution and sometimes lags'. App developers may find review (a) to be of no use, and on the contrary, addressing review (b) allows the app developers to fix bugs related to screen resolution and optimisation of the app.

Apart from the filtering approaches mentioned above, linguistic approach governed by a set of application specific filtering rules are seen promising by researchers. For instance, Iacob and Harrison (2013) have defined a set of linguistic rules to extract only feature requests (i.e., unigrams of interest) from reviews. Such an approach is often combined with an appropriate machine learning method for scalability purpose to identify useful features that require attention. For instance, Cleland-Huang et al. (2007) have developed a machine learning method (i.e., probabilistic classifier) to classify non-functional requirements by predicting their appropriate labels (i.e., performance, availability, security, usability and so on). However, the machine learning method (like many others) used by the authors require a large amount of learning data (i.e., requirements with their associated labels) to attain the required level of accuracy needed for performing predictions (Michie et al., 1994). That said, Multinomial Naïve Bayes is the most popular and commonly utilised supervised machine learning method that has been empirically evaluated to be a reliable option for text related software engineering applications (i.e., information conveyed through English language and expressed in text) and was found to outperform other machine learning methods (Caruana & Niculescu-Mizil, 2006). For instance, Wang et al. (2018) have benchmarked the performance of Decision Trees, KNN (K Nearest Neighbours), Bagging and Multinomial Naïve Bayes towards the classification of functional and non-functional requirements, and discovered that Multinomial Naïve Bayes generated most reliable results. Moreover, Multinomial Naïve Bayes often prevents overfitting of the data made available for learning purpose due to its mechanism of generalisation towards predictions, further leading towards the requirement of less data for learning purpose (McCallum & Nigam, 2001). In addition, the semi-supervised variant of Multinomial Naïve Bayes method i.e., Expectation Maximisation for Multinomial Naïve Bayes further reduces the amount of data required for learning purpose (Collins, 2012; Nigam et al., 2000). Thus, Multinomial Naïve Bayes method is widely used in software engineering applications such as software bug predictions, predicting the labels of non-functional requirements, spam content filtering and so on (Bacchelli et al., 2012; Calders & Verwer, 2010). One study has developed a filtering approach to predict useful reviews using the Multinomial Naïve Bayes method, however the algorithmic and implementation details of the approach were not provided. In addition, even though the filtering approach was used to predict numerous useful reviews belonging to different apps, the approach's filtering performance (F-Measure = 0.86) was reported for reviews of only one app, further questioning its generalisability (Chen et al., 2014). This also raises the question '*Under what circumstances and settings does the Multinomial Naïve Bayes method deliver the reported performance?*'. Hence, based

on the recommendations provided by the above-mentioned pertinent studies that show the Multinomial Naïve Bayes method to be superior in terms of performance than other algorithms for software engineering based application, we shortlisted and reviewed the method and its associated concepts for further investigation. Subsequently, this method is specialised in text based prediction applications, and further assisted us to identify and evaluate six variants of Multinomial Naïve Bayes methods towards their utility for information retrieval (i.e., filtering useful reviews) via text classification (Collins, 2012; Nigam et al., 2000; Yuan et al., 2012). The prime objective of investigating these variants is to assist app developers in filtering useful reviews to support the maintenance and evolution cycles of the apps. This leads to the RQ

> RQ2.1 What are the performances of the Multinomial Naïve Bayes variants when extracting useful reviews, and are there differences in the outcomes of these variants?

It is to be noted that while Multinomial Naïve Bayes stands out as one of the most suitable for filtering of useful reviews, we have not observed published efforts aimed at designing its possible variants and evaluating the performances of those variants.

4.3 Methods and Concepts

We introduce the Multinomial Naïve Bayes method and concepts that assisted us in developing the six variants. The prime objective of these variants is to filter useful reviews by classifying useful and non-useful reviews present in the vast pool of reviews through means of learning and predictions. The required set of useful and non-useful reviews for learning purpose can be manually labelled using a set of filtering rules proposed by Chen et al. (2014). The rules related to useful reviews indicate feature requests (e.g., "please add the feature to search for multiple routes"), bugs (e.g., "the map freezes after few minutes of loading") or enhancements (e.g., "I suggest you also add the black theme for the layout to make it look better"). Subsequently, non-useful reviews indicate unwanted and irrelevant information (e.g., "stupid app is useless, uninstalling now!"). Thus, the objective of the respective variant is to assign each review to one of the two categories (C) (i.e., useful or non-useful) wherein each category would contain reviews with properties reflecting the relevant filtering rules. In the learning (training) stage the particular variant generates a classifier trained from a set of substantial manually labelled reviews that predicts the categories of unlabelled reviews in the classification stage (prediction or testing) and so the useful reviews can be distinguished from the non-useful ones for filtering purpose. In the following sub-sections, we document the transformation of reviews into a suitable dictionary that is used as an input for the six variants. Then we provide the overview of the Multinomial Naïve Bayes method followed by the concepts of Complement, Laplace Smoothing and Expectation Maximisation.

4.3.1 Reviews Pre-Processing

Text pre-processing allows the conversion of reviews into subsequent word vectors through a series of pre-processing operations (Aggarwal & Zhai, 2012). We pre-process the reviews by removing numbers, whitespaces, special characters (e.g., $, #) and punctuations (e.g., !, ?) before transforming them into lower case (Maalej et al., 2016a). Later, any stop words (e.g., is, and) present in the pre-processed reviews are removed and lemmatisation is performed to generate the original dictionary form of the words present in the pre-processed reviews (Maalej et al., 2016a). These mentioned steps are standard text pre-processing operations that are performed by researchers to have reliable features (words) for the specific research purpose (e.g., learning and prediction), and simultaneously prevent the generation of unreliable and noisy results (Maalej et al., 2016a). The final set of pre-processed reviews are used to form the dictionary (D) that provides the required word frequency information for the variants (McCallum & Nigam, 2001).

4.3.2 Multinomial Naïve Bayes

Multinomial Naïve Bayes is an extended version of the basic Naïve Bayes method and is specialised for text based machine learning classification applications (McCallum & Nigam, 2001). The foundation of this method is based on the principle of maximum likelihood estimates as the method uses word frequency information extracted from the reviews. Initially, Multinomial Naïve Bayes computes the probability of a review belonging to a particular category (C) which is given as

$$P(C) = Nrs(r=C) / Nrs \quad (1)$$

Where, Nrs indicates the total number of reviews and Nrs(r=C) indicates the number of reviews belonging to a category C, and C = {useful, non-useful}. Subsequently, the maximum likelihood estimate is computed as

$$P(w_n|C) = freq(w_n, C) / \sum_{w \in D} freq(w, C) \quad (2)$$

Where, $P(w_n|C)$ indicates the conditional probability of a word w_n given that it belongs to category C which is given as the ratio of the total number of occurrences of the word w_n in category C to the total number of words w present in the reviews of category C. This is the fraction of the total number of times word w_n appears among all words (D) in the reviews that belong to category C. The Multinomial Naïve Bayes method generates a word space for a category C by creating a dictionary of words belonging to the reviews of category C. This is achieved by identifying the frequency of occurrence of each word w. Using equations (1) and (2), the category of a review R can be predicted using

$$C_{MAP}(R) = \text{argmax}_c \, (P(C) * \Pi_n P(w_n|C)) \quad (3)$$

C_{MAP} indicates the most probable category defined as maximum a posteriori (MAP) which indicates the most likely category C for a review R given as the arguments of maxima over all the categories of the priori times the likelihood. The learning and prediction stage of Multinomial Naïve Bayes method is given in algorithm 1 (McCallum & Nigam, 2001)

Algorithm 1: Learning and prediction stage of Multinomial Naïve Bayes

Input: A set of reviews

Processing:
Begin

1. From the manually labelled reviews, extract Dictionary (D)
2. Calculate all the P(C) terms
   2.1 For each C do:
      2.1.1 $reviews_C$ ← all reviews in category C
      2.1.2 P(C) ← $|reviews_C|$ / |Total reviews|
3. For every word w_n, given every category C
   3.1 Calculate $P(w_n|C)$ (maximum likelihood estimates)
      3.1.1 $WordSpace_C$ ← words belonging to $reviews_C$
      3.1.2 For each word w_n in the Dictionary (D)
         3.1.2.1 n_n ← Total occurrences of w_n in $WordSpace_C$ consisting of a total of n words
         3.1.2.2 $P(w_n|C)$ ← n_n / n
4. For every unlabelled review (R):
   4.1 Compute $C_{MAP}(R)$

End

Output: Each review categorised into one of two categories (useful and non-useful).

4.3.3 Complement Naïve Bayes

The Complement Naïve Bayes is the complement concept of Multinomial Naïve Bayes that computes the likelihood of a category C using the training data of all the other categories $\bar{C}$ other than C. The Complement Naïve Bayes was developed to address a potential drawback of the Multinomial Naïve Bayes method which was its inability to generate accurate predictions if the method was trained with data (reviews) having imbalanced labels (categories), i.e., the reviews in the learning stage did not belong to approximately equal number of different types of categories (Rennie et al., 2003). Using equation (1), the Complement Naïve Bayes computes the prior probability. However, unlike the Multinomial Naïve Bayes method, the Complement Naïve Bayes computes the likelihood of a word w_n by considering its occurrences in category(ies) $\bar{C}$ other than C. Thus, the maximum likelihood is calculated as

$$P(w_n|\bar{C}) = freq(w_n, \bar{C}) / \sum_{w \in D} freq(w, \bar{C}) \quad (4)$$

Where $P(w_n|\bar{C})$ indicates the conditional probability of a word w_n given that it belongs to category(ies) $\bar{C}$ which is given as the ratio of the total number of occurrence of the w_n in category(ies) $\bar{C}$ to the total number of words w present in the reviews of category(ies) $\bar{C}$. The Complement Naïve Bayes creates a word space for a category C by creating a dictionary of words belonging to the reviews of category(ies) $\bar{C}$ by identifying the occurrences of w. Using equations (1) and (4), the category of a review R is predicted using

$$C_{MAP}(R) = \text{argmin}_C (P(C) * \Pi_n (1/ (P(w_n|\bar{C})))) \quad (5)$$

Where C_{MAP} (R) indicates the most probable category given as the argument of the minimum of likelihood estimates of the category(ies) computed as priori times the inverse likelihood. The learning and prediction stage of Complement Naïve Bayes is given in algorithm 2 (Rennie et al., 2003).

Algorithm 2: Learning and prediction stage of Complement Naïve Bayes

Input: A set of reviews

Processing:
Begin

1. From the manually labelled reviews, extract Dictionary (D)
2. Calculate all the P(C) terms
   2.1 For each C do:
      2.1.1 $reviews_C$ ← all reviews in category C
      2.1.2 P(C) ← $|reviews_C|$ / |Total reviews|
3. For every word w_n, given every category C
   3.1 Calculate $P(w_n|\bar{C})$ (maximum likelihood estimates)
      3.1.1 $WordSpace_C$ ← words belonging to reviews of category(ies) $\bar{C}$
      3.1.2 For each word w_n in the Dictionary (D)
         3.1.2.1 n_n ← Total occurrences of w_n in $WordSpace_C$
            consisting of a total of n words
         3.1.2.2 $P(w_n|\bar{C})$ ← n_n / n
4. For every unlabelled review (R):
   4.1 Compute $C_{MAP}(R)$

End

Output: Each review categorised into one of two categories (useful and non-useful).

4.3.4 Laplace Smoothing

The parameters of equations (2) and (4) that compute the maximum likelihood estimates are unable to deal with zero probabilities (Lowd & Domingos, 2005). Multinomial Naïve Bayes and Complement Naïve Bayes would return zero probability of a word if the particular word is not present in the learning stage which in turn affects the accuracy of prediction. This problem is resolved by subjecting the parameters to Laplace Smoothing (Jung et al., 2016; Yuan et al., 2012). Laplace smoothing enables the particular Multinomial Naïve Bayes variant (e.g., Complement Naïve Bayes) to keep track of the

frequency of words in predicting the relevant category by adding 1 to parameters to manage the zero occurrences of a particular word efficiently. Utilisation of the Laplace Smoothing concept is of prime importance when the particular Multinomial Naïve Bayes variant comes across a word in the prediction stage (classification) whose information is not present in the learning (training) stage. Hence, we update equations (2) and (4) to incorporate the Laplace Smoothing concept to manage the information related to a particular missing word w_n. For Multinomial Naïve Bayes using equation (2) the parameter that performs maximum likelihood estimation based on Laplace Smoothing is given as

$$P(w_n|C) = (freq(w_n, C) + 1) / (\sum_{w \in D} freq(w, C) + |D|) \quad (6)$$

Similarly, for Complement Naïve Bayes using equation (4) the parameter that performs maximum likelihood estimation based on Laplace Smoothing is given as

$$P(w_n|\bar{C}) = (freq(w_n, \bar{C}) + 1) / (\sum_{w \in D} freq(w, \bar{C}) + |D|) \quad (7)$$

In equations (6) and (7), as the addition of 1 is considered in the numerator, the size of the dictionary (|D|) is added in the denominator indicating the addition of one for every dictionary word in the denominator. Based on equations (6) and (7) the learning stages of Multinomial Naïve Bayes and Complement Naïve Bayes can be updated accordingly.

4.3.5 Expectation Maximisation

Multinomial Naïve Bayes and Complement Naïve Bayes are supervised machine learning algorithms that require a substantial number of manually labelled reviews to learn a classifier that is capable of accurately predicting the category of an unlabelled review (McCallum & Nigam, 2001; Rennie et al., 2003). Manually labelling the required number of reviews might become a time consuming task associated with potential errors. An appropriate semi-supervised learning concept addresses this drawback by reducing the labelling effort demanded from humans and Expectation Maximisation (EM) is a popular and commonly utilised semi-supervised concept (Collins, 2012; Nigam et al., 2000). EM comprises of two steps, Expectation (E) and Maximisation (M). The E step predicts and generates the unknown information based on the current maximum likelihood estimation parameters initiated by Multinomial Naïve Bayes method and the M step iteratively re-estimates the parameters which leads to the maximisation of the overall likelihood (Collins, 2012). EM allows the Multinomial Naïve Bayes method to run repeatedly until the maximum likelihood estimates become constant (Nigam et al., 2000). The goal of EM concept for this study is to create the respective semi-supervised variants of the Multinomial method and these variants were developed according to the algorithm mentioned in (Collins, 2012; Nigam et al., 2000). The initial stages of EM would consist of training the Multinomial Naïve Bayes method on the manually labelled categories of reviews and then later, using the learned information on categories related to the reviews to make predictions on the unlabelled reviews. Thus,

the predictions can be transformed into categories and can be utilised for subsequent iterative training of the Multinomial Naïve Bayes method using the unlabelled reviews with the previously predicted categories. The entire procedure needs to be repeated until the value generated by the maximum likelihood parameters becomes constant (likelihood is computed using the entire corpus of pre-processed reviews). The above-mentioned details of the EM of Multinomial Naïve Bayes (Collins, 2012; Nigam et al., 2000) have been elaborated stepwise in algorithm 3. The detailed explanation of the mentioned algorithm is as follows; consider a reviews set RS containing reviews wherein each review is manually labelled with a category C (useful or non-useful). The prime objective of EM is to predict the categories of the unlabelled reviews based on the prediction mechanism of Multinomial Naïve Bayes. In every iteration, EM calculates the appropriate probabilistic category and assigns it to the particular unlabelled review, that is $P(C_u|R_i)$ which is estimated to be 0 or 1. Here C_u denotes the particular category and R_i indicates the particular review. The labelled reviews having a specific category (a) is known prior, hence $P(C_a|R_i) = 1$ and $P(C_b|R_i) = 0$ for $a \neq b$. Using the information of labelled reviews and $P(C_u|R_i)$ an updated version of the Multinomial Naïve Bayes classifier is generated which works in a recursive manner until $P(w_n|C)$ and $P(C)$ become constant.

Algorithm 3: Expectation Maximisation of Multinomial Naïve Bayes

Input: A set of reviews

Processing:
Begin

1. Train the Multinomial Naïve Bayes mNB from the manually labelled and pre-processed set of reviews R.
2. Expectation (E):
   2.1 For each review Ri in the review set RS
      2.1.1 Using the method mNB, calculate $P(C_u|R_i)$
3. Maximization (M):
   3.1 Train an updated version of mNB from $R \cup RS$ by calculating $P(C)$ and $P(w_n|C)$
4. Repeat steps 2 and 3 until mNB's parameters (maximum likelihood estimators) become constant.
5. Return mNB after completion of step 4.

End

Output: Each review categorised into one of two categories (useful and non-useful)

That said, while EM integrates well with Multinomial Naïve Bayes, the Complement Naïve Bayes does not support any generative interpretations, and hence the creation of its EM variant is not possible (Rennie et al., 2003).

4.4 Multinomial Naïve Bayes Variants

The six Multinomial Naïve Bayes variants that were an outcome of the method and the concepts mentioned in section 4.3 are briefly elaborated in Table 4.1. Table 4.1 provides the name of the

particular variant along with its description. We first formulated the variants belonging to the Multinomial Naïve Bayes method. Based on the method mentioned in sub-section 4.3.2 and the concepts mentioned in sub-sections 4.3.4 and 4.3.5, there are four possible variants (I, II, III and IV) related to the Multinomial Naïve Bayes method. Similarly, based on the Complement concept mentioned in sub-section 4.3.3 and the concept mentioned in sub-sections 4.3.4 we formulated two possible variants (V and VI) of Complement Naïve Bayes.

Table 4.1 Six Multinomial Naive Bayes variants

Variant	Name	Description
I	Multinomial Naïve Bayes	This variant is the Multinomial Naïve Bayes method introduced in sub-section 3.2.2.2
II	Expectation Maximisation of Multinomial Naïve Bayes	The Expectation Maximisation concept described in sub-section 3.2.2.5 has been integrated with I. Therefore, this variant is the semi-supervised version of I
III	Multinomial Naïve Bayes with Laplace Smoothing	The Laplace Smoothing concept described in sub-section 3.2.2.4 has been incorporated in I and therefore this variant is the post version of I.
IV	Expectation Maximisation of Multinomial Naïve Bayes with Laplace Smoothing	The Multinomial Naïve Bayes method has been integrated with Expectation Maximisation concept and incorporated with Laplace Smoothing concept making this variant a semi-supervised version of III and a post version of II.
V	Complement Naïve Bayes	This variant is the Complement concept of the Multinomial Naïve Bayes method described in sub-section 3.2.2.3.
VI	Complement Naïve Bayes with Laplace Smoothing	Variant V has been incorporated with the concept of Laplace Smoothing making this variant a post version of V.

4.5 Experimental Settings

In this pilot study, the six variants described in Table 4.1 were implemented using Python[7] with the support of suitable libraries provided by Natural Language Tool Kit[8] (NLTK), numpy[9] and scikit-learn[10] packages. In the remaining experiments conducted in the subsequent phases, all the necessary implementations were done using the same programming language and the supporting libraries. Moreover, we utilised R[11] to perform the necessary statistical computing and analysis. That said, the performances of the six variants towards filtering of useful reviews were evaluated using the datasets provided by the app developers belonging to two different apps. In addition, the two supervisors of this PhD study had previously worked with some of the datasets while providing insights for the app developers, and hence, the supervisors have a thorough understanding and knowledge of the contents of these datasets that was later shared with the candidate of this PhD study (Keertipati et al., 2016;

[7] https://www.python.org/
[8] https://www.nltk.org/
[9] https://numpy.org/
[10] https://scikit-learn.org/
[11] https://www.r-project.org/

Licorish et al., 2017). The first dataset belonged to My Tracks app and the second dataset belonged to Flutter app (Keertipati et al., 2016; Licorish et al., 2017). My Tracks dataset consisted of 4003 reviews while Flutter dataset consisted of 3483 reviews. Using the set of filtering rules defined in (Chen et al., 2014) we independently labelled the reviews of both datasets before reliability assessments were performed. The task of manual labelling was undertaken to empirically evaluate the performances of six variants based on the domain knowledge made available by humans (i.e., cross-validating the results generated from human decisions against those generated by the respective variant). Such cross validation approaches are deemed reliable and the provided domain knowledge acts as the ground truth (Stumpf et al., 2007). That said, after performing the reliability assessments the Fleiss coefficients were found to be 0.74 (substantial agreement) and 0.77 (substantial agreement) for My Tracks and Flutter datasets (Landis & Koch, 1977). Follow up discussions were held to resolve any disagreements and establish consensus leading to 100% agreement. Based on the manual labelling task, My Tracks dataset consisted of 1638 (41%) useful reviews and 2365 (59%) non-useful reviews. Flutter dataset consisted of 2433 (70%) of useful reviews and 1063 (30%) of non-useful reviews making it imbalanced (Rennie et al., 2003).

The objective of classifying the reviews using the particular variant is to determine the category of each review by means of learning and prediction mechanism of the variant. The performance of each variant towards the binary classification of reviews present in the two datasets was evaluated using standard metrics such as accuracy, precision, recall, F-Measure and time (Michie et al., 1994; Sokolova & Lapalme, 2009). As app developers want to extract useful reviews in a timely manner due to the time constrained app maintenance and evolution cycles, we note the time (in seconds) required by each variant to perform learning and predictions (Michie et al., 1994). Accuracy determines the ability of a variant to correctly predict the category of the reviews given as the number of correctly classified reviews among the total number of classified reviews (Sokolova & Lapalme, 2009) and is given as

Accuracy = (true positives + true negatives) / (true positives + true negatives + false positives + false negatives) (8)

In (8), true positives term indicates the number of reviews correctly predicted as useful, true negatives indicates the number of reviews correctly predicted as non-useful, false positives indicates the number of reviews that were predicted as useful but were actually non-useful and false negatives indicates the number of reviews that were predicted as non-useful but were actually useful.

Precision indicates the number of correctly predicted useful reviews among the total number of reviews predicted as useful and recall indicates the number of correctly predicted useful reviews to the total number of actual useful reviews (Sokolova & Lapalme, 2009), and both are given as

Precision = true positives / (true positives + false positives) (9)

Recall = true positives / (true positives + false negatives) (10)

F-Measure is the harmonic mean of precision and recall which determines the robustness of the variants (Sokolova & Lapalme, 2009), and is given as

F-Measure = 2 * (Precision * Recall) / (Precision + Recall) (11)

The computer used for the conducting the experiments of this phase including those in the other phases had a CORE i5 CPU and 14GB RAM. For each experiment of this phase, we randomly split the respective dataset into a training set (90%) that is used to learn the relevant variant and a testing set (10%) which is used to evaluate the performance of the variant in predicting the categories of the nondisclosed reviews. Each experiment was run 100 times using ten-fold cross validation mechanism to obtain average results of the metrics mentioned above, and such evaluation approach is traditionally followed by researchers to ensure the stability of the machine learning methods (Arlot & Celisse, 2010; Kohavi, 1995).

We present the results of the experiments conducted on the two datasets in the Results section.

4.6 Results

In this section, we present the results of the pilot study conducted towards the filtering of useful reviews. We report the average results of 100 times ten-fold cross validation operations conducted on My Tracks and Flutter datasets in Table 4.2 and Table 4.3 respectively. These results provide context for answering RQ2.1 and provide triangulations for RQ2.

4.6.1 My Tracks Dataset

Initially, we evaluated the performance of the six Multinomial Naïve Bayes variants on the My Tracks dataset. Table 4.2 indicates average performances of the variants for the My Tracks dataset.

Table 4.2 Multinomial Naive Bayes variants average performance on My Tracks dataset

Variant	Accuracy (%)	Precision (0-1)	Recall (0-1)	F (0-1)	Time (seconds)
I	68.1	0.56	0.98	0.71	0.26
II	80.4	0.73	0.88	0.80	0.30
III	87.4	0.81	0.91	0.86	0.12
IV	89.2	0.84	0.94	0.89	0.19
V	84.6	0.76	0.90	0.82	0.15
VI	86.5	0.78	0.91	0.84	0.10

Initially, we conducted the Shapiro-Wilk test to evaluate the distribution of the results generated by each variant (i.e., 100 results of each variant) (Sheskin, 2003). From the conducted test, we found no evidence regarding normal distribution of the results (p-value < 0.01). Hence, we conducted the Kruskal-Wallis non-parametric test to check for statistically significant differences between the results of the Multinomial Naïve Bayes variants (Sheskin, 2003). Result of the conducted test showed that there were statistically significant differences (p-value < 0.01) among all the results of the Multinomial Naïve Bayes variants for all performance metrics (i.e., accuracy, precision, recall, F-Measure and time). Thus, we performed the pairwise Wilcox test to evaluate pairwise comparisons between the results of the Multinomial Naïve Bayes variants with corrections for multiple testing (Wilcox, 2011). We observed statistically significant differences for all comparisons (p-value < 0.01) pertaining to the accuracy, precision, recall, F-Measure and time metrics.

As observed from Table 4.2 variant I exhibited the lowest accuracy (68.1%) and F-Measure (0.71) compared to the other variants. On the contrary, variant IV exhibited the highest accuracy (89.2%) and F-Measure (0.89). In addition, variant VI required the least amount of time to perform learning and predictions (0.10 seconds) while variant II required the highest time (0.30 seconds). Moreover, the semi-supervised variants II and IV performed better than their supervised variants I and II in terms of accuracy (80.4% versus 68.1%, 89.2% versus 80.4%) and F-Measure (0.80 versus 0.71, 0.89 versus 0.80). However, these semi-supervised variants required more time than their predecessor variants. The supervised variants III, V and VI outperformed semi-supervised variant II in terms of accuracy (87.4%, 84.6%, 86.5% versus 80.4%), F-Measure (0.86, 0.82, 0.84 versus 0.80) and time (0.12 seconds, 0.15 seconds, 0.10 seconds versus 0.30 seconds). The variants V and VI derived from the Complement concept outperformed variants I and II in terms of accuracy (84.6%, 86.5% versus 68.1%, 80.4%), F-Measure (0.82, 0.84 versus 0.71, 0.80) and time (0.15 seconds, 0.10 seconds versus 0.26 seconds, 0.30 seconds). Similarly, variants III and IV outperformed variants I and II in terms of accuracy (87.4%, 89.2% versus 68.1%, 80.4%), F-Measure (0.86, 0.89 versus 0.71 and 0.80) and time (0.26 seconds, 0.30 seconds versus 0.12 seconds, 0.19 seconds). Furthermore, Laplace smoothing led towards an increase in accuracy of variants III, IV and VI when compared to their respective earlier versions I, II and V (87.4% versus 68.1%, 89.2% versus 80.4%, and 86.5% versus 84.6%) and F-Measure (0.86 versus 0.71, 0.89 versus 0.80, and 0.84 versus 0.82). In addition, Laplace smoothing assisted in reducing the time required for performing learning and predictions in cases of the same variant pairs (III-I, IV-II, and VI-V).

4.6.2 Flutter Dataset

Next, we evaluated the performances of the six variants on Flutter dataset. Table 4.3 indicates the average performances of the variants.

Table 4.3. Multinomial Naive Bayes average performance on Flutter dataset

Variant	Accuracy (%)	Precision (0-1)	Recall (0-1)	F (0-1)	Time (seconds)
I	76.2	0.75	0.97	0.85	0.19
II	80.3	0.82	0.91	0.86	0.23
III	80.5	0.81	0.94	0.87	0.12
IV	82.3	0.84	0.93	0.88	0.16
V	80.4	0.83	0.87	0.85	0.10
VI	84.4	0.87	0.91	0.89	0.08

Firstly, we repeated the Shapiro-Wilk test to investigate the distribution of the results generated by each variant and observed no normal distribution (p-value < 0.01). Hence, we conducted the Kruskal-Wallis test followed by the pairwise Wilcox test to check for any statistically significant differences among all the results of the six variants in Table 4.3. Both tests returned statistically significant differences (p-value < 0.01).

That said, as observed from Table 4.3, variant I exhibited the lowest accuracy (76.2%), whereas VI had the highest accuracy (84.4%), F-Measure (0.89) with least time (0.08 seconds) required for performing learning and predictions. Variant II had the highest time requirements (0.23 seconds) and variant IV ranked second in terms of accuracy (82.3%) and F-Measure (0.88). Variants II, III and V did not exhibit large differences in magnitude of accuracy and F-Measure results despite these differences being significant statistically (p-value < 0.01). In addition, Laplace smoothing assisted in reducing the time required by variants III, IV and VI to perform learning and predictions in comparison to their respective earlier versions I, II and V (0.12 seconds versus 0.19 seconds, 0.16 seconds versus 0.23 seconds, and 0.08 seconds versus 0.10 seconds). Laplace smoothing also assisted in increasing the accuracy (80.5% versus 76.2%, 82.3% versus 80.3%, and 84.4% versus 80.4%) and F-Measure (0.87 versus 0.85, 0.88 versus 0.86, and 0.89 versus 0.85) in cases of the same variant pairs (III-I, IV-II, and VI-V).

Furthermore, based on the evaluation results of My Tracks and Flutter datasets, in case of pure Multinomial Naïve Bayes variants (i.e., excluding the Complement concept) variant IV outperformed the other variants in terms of accuracy and F-Measure while variant III had the least time requirements to perform learning and predictions. In cases of variants belonging to the Complement concepts, variant VI outperformed its predecessor variant V in terms of accuracy, F-Measure and time.

We discuss the results of the undertaken pilot study on useful reviews filtering and the considerations of their implications in the Discussion section.

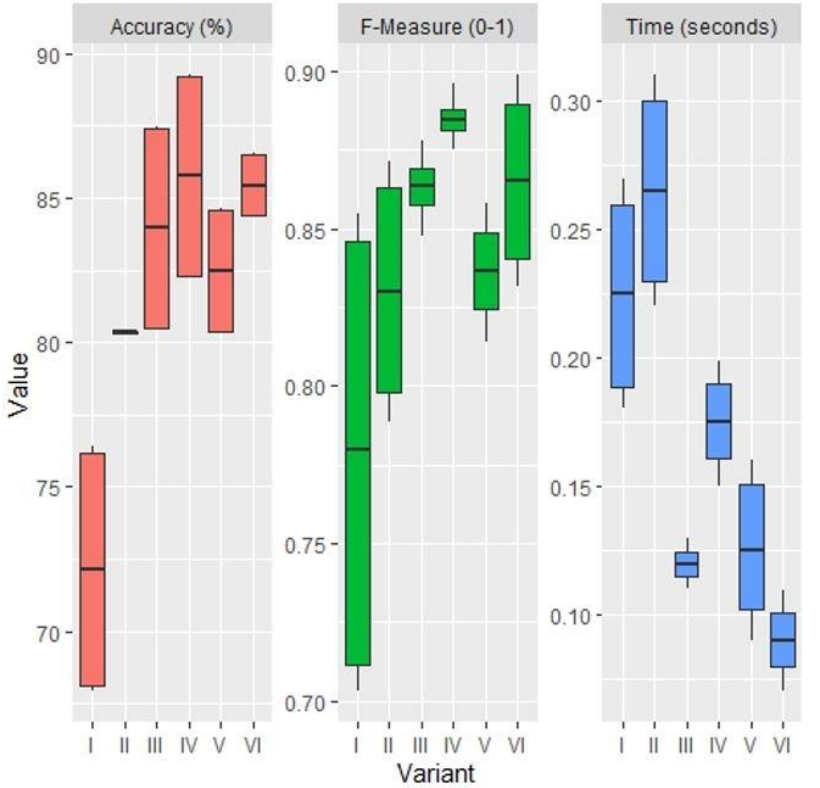
4.7 Discussion

As reviews pertaining to an app usually tend to be numerous, there was a necessity to develop an automated filtering approach to identify and extract useful reviews. Doing so would not only assist the app developers to analyse or visualise information within the filtered useful reviews which is beneficial but also generate accurate and timely classification or prioritisation results, as significant amount of noisy and unwanted information negatively affecting the accuracy and time metrics would be avoided. The studies on requirements elicitation identified from the systematic study on requirements prioritisation were the primary source of inspiration for such a filtering approach along with the need to process large numbers of reviews associated with apps which majorly consist of non-useful reviews (i.e., irrelevant information for app developers) (Chen et al., 2014; Garg et al., 2017; Sadiq et al., 2017). This guided us to review studies that specialised in automated filtering of information that was of interest to the researchers. In this regard, we identified several studies from the app reviews domain but these studies had several limitations in their proposed filtering approaches (refer to section 4.2). However, continuing our search further, we identified a study that showed the rule based Multinomial Naïve Bayes method is one of the most reliable and suitable approach to filter useful reviews (Chen et al., 2014). As the study did not provide the algorithmic and implementation details of the method, apart from the rules for filtering, we had to examine the Multinomial Naïve Bayes method which lead to the discovery of six variants pertaining to the same method whose utility towards filtering useful reviews was investigated in our pilot study. Such an empirical evaluation was never performed in any prior studies, which allowed us to provide a contribution to the software engineering discipline. The sub-section below discusses the results and implications of RQ2.1.

4.7.1 RQ2.1 What are the performances of Multinomial Naïve Bayes variants when extracting useful reviews, and are there differences in outcomes of the different implementations?

When the results for the two datasets (i.e., My Tracks and Flutter) are observed, we notice varied performances exhibited by the six variants of Multinomial Naïve Bayes (Schaffer, 1993). This is inferred based on the results conveyed through accuracy, F-Measure and time metrics (Schaffer, 1993). We believe the features affiliated with each label (i.e., useful or non-useful) play a significant role in predicting the relevant label of a review (Yuan et al., 2012; Zhu et al., 2006). This may be the reason behind the variations in performances exhibited for the six variants when predicting useful and non-useful reviews for the two datasets. Based on this observation, we are of the opinion that the variants may reliably predict the label of each review if the features associated with the label had substantial degree of distinctness (i.e., features related to a label are notably discrete in comparison to the features related to the other labels), an aspect that requires empirical examination (Yuan et al., 2012; Zhu et al., 2006). In the experiments conducted related to the two datasets and the results presented in Tables 4.2

and 4.3, the readings related to precision indicate a particular variant's ability to correctly identify useful reviews out of all the reviews that are actually useful. In such case, variant IV exhibits the highest precision for My Tracks dataset whereas, variant VI exhibits the highest precision for Flutter dataset. Subsequently, the readings related to recall indicate that for all the reviews that are useful, how many of such useful reviews did a particular variant correctly identified as useful. In such case, variant I exhibits the highest recall for both datasets. That said, app developers might utilise a particular variant towards filtering of useful reviews based on the application requirement. For instance, app developers might utilise variant I if the filtering of useful reviews is based only on recall metric. However, prominent app studies based on domain experts' suggestions have considered results based on F-Measure as a significant deciding factor towards determining the robustness of a machine learning method (i.e., based on combination of both - precision and recall) (Chen et al., 2014; Di Sorbo et al., 2016; Jiang et al. 2019). Hence, we formulate the further discussion of results based on F-Measure that considers the metrics precision and recall. Figure 19 provides a visualisation of the performance results based on accuracy, F-Measure and time metrics of the six variants pertaining to the two datasets.



Legend: I - Multinomial Naïve Bayes, II - Expectation Maximization of Multinomial Naïve Bayes, III - Multinomial Naïve Bayes with Laplace Smoothing, IV - Expectation Maximization of Multinomial Naïve Bayes with Laplace Smoothing, V - Complement Naïve Bayes, VI- Complement Naïve Bayes with Laplace Smoothing

Figure 19. Overall performance of Multinomial Naive Bayes variants based on accuracy, F-Measure and time. This is based on aggregate results for both datasets.

Figure 19 allows for meaningful interpretation of patterns observed in the generated results. From Figure 19, it can be observed that the Expectation Maximisation Multinomial Naïve Bayes variants (II and IV) significantly improved the performance (accuracy and F-Measure) of the primary Multinomial Naïve Bayes variants (I and III) respectively. The Expectation Maximisation customisations resulted in as much as 9.1% improvement in accuracy in retrieving useful reviews. On the contrary, the Expectation Maximisation variants (II and IV) required more time to perform learning and predictions (31.8% increase in time). The observed increase in accuracy and F-Measure as seen in Figure 19 is due to the EM mechanism of II and IV that allows these variants to gain maximum information about the words present in app reviews belonging to the same category (useful or non-useful) during the particular EM variant's learning phase.

This can be observed in sub-section 4.3.5 when unclassified and classified reviews are passed to the particular EM variant, which in turn allows the EM variant to gain insights about the different types of words pertaining to a specific category in the variant's learning phase. This crucial information gained during the learning phase leads towards the increase in accuracy and F-Measure. Moreover, the algorithmic structure of Multinomial Naïve Bayes (I) and Multinomial Naïve Bayes with Laplace Smoothing (III) is based on closed form formulas, which enable these variants to generate results quickly (Ren et al., 2009). However, the Expectation Maximisation of Multinomial Naïve Bayes and Expectation Maximisation of Multinomial Naïve Bayes with Laplace smoothing generate results based on an iterative approach (EM computation continues until likelihood parameters become constant), thus needing more time for learning and making predictions.

With regards to Laplace smoothing, results show that Laplace smoothing augmentation assisted significantly in increasing accuracy and F-Measure, and reduced the time required for learning and prediction purposes involving Multinomial Naïve Bayes, Expectation Maximisation of Multinomial Naïve Bayes and Complement Naïve Bayes. We note a 17.0% increase in accuracy, 0.1 improvement in F-Measure and 0.11 seconds reduction in time which were accounted for by Laplace smoothing. Laplace smoothing augmentation enhanced the retrieval of useful reviews. As observed from equations (6) and (7), Laplace smoothing prevents the zero counts of words whose information is not available during the learning phase, thus preserving the value of maximum likelihood estimates that are crucial towards the prediction of a category of a review. Hence, any maximum likelihood estimate being 0 compromises a variant's judgement towards determining the relevant category of a review. In addition, the Laplace smoothing variants (III, IV and VI) compute faster estimates of the parameters that generate the likelihood, hence improving Multinomial Naïve Bayes's overall performance.

From Figure 19 it is observed that, overall, Expectation Maximisation of Multinomial Naïve Bayes with Laplace smoothing (IV) performed well on both datasets in terms of accuracy and F-Measure. Thus, from an application perspective, Expectation Maximisation of Multinomial Naïve Bayes with Laplace

smoothing may be a suitable variant for the information retrieval task which involves limited number of reviews that are manually labelled (categorised) by app developers. Subsequently, Complement Naïve Bayes with Laplace Smoothing (VI) performed well on the Flutter dataset. This is because, the complement concept of Multinomial Naïve Bayes allows it to perform well when the dataset consists of reviews with imbalanced labels. That said, concerning both datasets, Complement Naïve Bayes with Laplace Smoothing had the least time requirements (average ~ 0.1 seconds). Therefore, the application of Complement Naïve Bayes with Laplace smoothing is potentially suited when app developers have a substantial number of labelled reviews whose categories (labels) are imbalanced and at the same instance are bounded by severe time constraints to extract useful reviews.

Furthermore, it is to be noted that all the Multinomial Naïve Bayes variants operated on the assumption of independence. This indicates that each variant disregards the meaning of the words it processes relative to other words. This is a questionable assumption as it may compromise a variant's ability to perform predictions when processing words belonging to real world learning and prediction applications (John & Langley, 1995). For instance, consider the review 'the map pixelates every time I run', the words 'map' and 'pixelates' are related as the word pair 'map - pixelates' indicates that the map becomes unclear to the app's end-user when the end-user starts running. This is not modelled by the Multinomial Naïve Bayes method and hence, the method and its variant exhibit the independence assumption. That said, other machine learning methods such as logistic regression attempt to fit a normal curve or discretise the words (Ng & Jordan, 2002). With regards to this, each variant assumes that the word space is normally distributed with zero variance between the words present in all the categories. Because of this, in some scenarios the particular variant may be unable to generate a reliable discretisation of interrelated (continuous) words (features) which may compromise the performance of the particular variant. A potential solution to solve this would be to test for the independence of the words to get a tentative estimate of prediction errors to determine the suitability of a particular variant or generate a zero normal distribution towards generating more efficient results in terms of accuracy and F-Measure (Boullé, 2006).

However, the results obtained from the conducted pilot study are promising (e.g., over 89% accuracy, 0.87 precision, 0.98 recall, 0.89 F-Measure, and 0.08 seconds time). Thus, the six Multinomial Naïve Bayes variants (especially variants: IV - Expectation Maximisation of Multinomial Naïve Bayes with Laplace Smoothing and VI - Complement Naïve Bayes with Laplace Smoothing) investigated in this pilot study on their own hold promise for aiding useful reviews filtering and software maintenance cycles.

4.8 Threats to Validity

In this section, we present the threats to validity that can potentially affect the outcomes reported in this Chapter.

4.8.1 Internal Validity

The prime objective of the pilot study conducted in this phase was to examine and compare the performance of the Multinomial Naïve Bayes variants against each other for their efficiency towards filtering of useful reviews. Hence, the performance of other machine learning approaches is not investigated in our study. However, potential future work aimed at conducting such an investigation could be planned. This investigation could involve the performance evaluation of popular machine learning algorithms such as BERT (Bidirectional Encoder Representations from Transformers), Decision Trees, Random Forests, Logistic Regression, SVM and so on, towards the filtering of useful reviews. In addition, addressing research aspects related to the distinct features (words) to be made available for learning and prediction purpose, and the independence assumptions made by the Multinomial Naïve Bayes variants were beyond the scope of this pilot study. That said, we have mitigated the threats related to the manual labelling of reviews for filtering purposes by: (a) using the feedback provided by app developers, (b) studying and becoming associated with the rules mentioned in (Chen et al., 2014) for labelling reviews, and (c) rigorously analysing the types of reviews that the app developers are concerned with. All the essential information including the rules were discussed among the three labellers for common understanding, before the reliability assessments were conducted which returned fair to substantial agreements. Follow up discussions were held to establish consensus before generating the appropriate results and finalising the particular outcomes.

4.8.2 External Validity

In this phase, we have used a computer with a particular hardware configuration (Core i5 CPU and 14GB RAM) which may limit the generalisability of certain results, especially those involving the measurement of time. However, the pattern of results is consistent across the two datasets used for evaluations, and hence, these results do not possess a threat to validity. We have utilised two datasets in the pilot study conducted in phase 2 and hence, the generalisability of the outcomes of this pilot study might be affected. However, the primary objective of this pilot study was to examine the feasibility of the proposed filtering of useful reviews approach and quantifying evaluation of the outcomes generated by this approach.

4.8.3 Construct Validity

To construct the ground truth data to filter useful reviews we followed the well-established rules from the prominent study to label the app reviews and the recommended practices from the software

engineering discipline (consensus formation). However, another alternative to construct this ground truth data would be to approach the app developers of the respective apps to obtain the labelled set of reviews to evaluate the performance of the filtering approach.

We provide the concluding remarks of this phase, its research contributions and summary of implications in the Conclusions chapter (refer to Chapter 7). In the next chapter, we present the details of Phase 3 (i.e., classification of useful reviews)

5 Classification of Useful Reviews

After figuring out an approach to filter useful reviews (refer to Chapter 4), our next objective was to convert the useful reviews into actionable knowledge by means of classification (RQ3). Thus, in phase 3 we developed and experimented with an automated taxonomy generation approach through means of a pilot study that answered one RQ. This RQ (RQ3.1) was aimed towards testing a preliminary approach to automatically generate a taxonomy for classifying useful reviews into groups of interest. In this chapter, we provide the details regarding phase 3 of the undertaken research work.

5.1 Introduction

In this phase, we conducted a pilot study which primarily deals with the classification of useful reviews into groups of interest. To achieve this, we first had to investigate the classification methods that classified reviews of apps where we identified a drawback. This drawback being, that all the classification methods were driven by manually derived taxonomy which is problematic when the domain knowledge is absent. Thus, we developed an approach that automatically creates a taxonomy from a corpus of useful reviews and later classifies them into specific groups of interests. The detailed elaboration towards the classification of useful reviews using the automatically generated taxonomy is presented in the following sections.

5.2 Related Studies

Beforehand, researchers have utilised classification as one of the approach to obtain actionable knowledge from reviews (Maalej et al., 2016a; Panichella et al., 2016). Such approach classifies reviews having common attributes into specific categories (groups) based on a taxonomy derived manually from domain knowledge, as a review of the literature shows that all the classification methods for classifying reviews are dependent on domain knowledge made available manually through means of extensive research or by domain experts. For instance, Panichella et al. (2015) have inherited a taxonomy from the taxonomy proposed by Pagano and Maalej (2013) and have evaluated the classification performance SVM (Support Vector Machines), Naïve Bayes, Decision Tress and Logistic Regression. Pagano and Maalej (2013) have manually assigned categories that constitute a taxonomy for classifying reviews. Similarly, Maalej et al. (2016a) manually developed four categories to classify reviews using methods such as keyword lookup classifying mechanism, Decision Tress, Naïve Bayes and Maximum Entropy. Such studies have provided inspiration for others. For instance, Panichella et al. (2016) developed a manual taxonomy that was inherited from the taxonomy created by Panichella et al. (2015) to automatically classify reviews using the J48 supervised machine learning method. In another study, Ciurumelea et al. (2018) have come up with five sets of categories by taking inspiration from (Panichella et al., 2015) and created a taxonomy to classify reviews using Gradient Boosting supervised machine learning method. Similarly, Dhinakaran et al. (2018) developed an automated classification approach

that classifies reviews into categories using a previously proposed taxonomy (Maalej et al., 2016a). In this study, a domain expert assigns categories to a set of random reviews and this information is used to automatically classify the remaining set of reviews using a machine learning method. The performance of supervised machine learning methods such as SVM, Logistic Regression and Naïve Bayes was evaluated towards the automated classification task. Sorbo et al. (2016) have developed a fine-grained taxonomy from the taxonomy proposed by Panichella et al. (2015) which consists of additional categories over the study it is based on. It is to be noted that, with such classification approaches the need to manually analyse reviews is unavoidable. For instance, Maalej et al. (2016a) have classified reviews into one of the four categories; user experience, bug reports, ratings and feature requests. Of note here is that the manually derived taxonomy does not provide the specific details (e.g., which feature is requested by the end-user or what type of bug is reported), thus requiring the app developers to analyse each of the classified review to obtain the necessary information. This limitation is observed for the above reviewed studies on classification of reviews. Table 5.1 provides a summary of the above-mentioned studies in which the first column indicates the study, followed by the type of taxonomy utilised, number of categories, the name of those categories and the automated classification methods evaluated.

Table 5.1 Summary of classification studies on reviews

Study	Taxonomy	Number of categories in taxonomy	Name of the categories	Classification methods
Pagano and Maalej (2013)	Manually derived	17	1. Recommendation 2. Helpfulness 3. Feature Information 4. How to 5. Praise 6. Content Request 7. Important Request 8. Other App 9. Feature Request 10. Noise 11. Other Feedback 12. Question 13. Promise 14. Shortcoming 15. Bug Report 16. Dispraise 17. Dissuasion	Manual classification
Panichella et al. (2015)	Manually derived	5	1. Information Seeking 2. Information Giving 3. Feature Request 4. Problem Discovery 5. Others	1. Naïve Bayes 2. SVM 3. Logistic Regression 4. Decision Tress

Study	Taxonomy	Number of categories in taxonomy	Name of the categories	Classification methods
Maalej et al. (2016a)	Manually derived	4	1. Bug Reports 2. Feature Requests 3. User Experience 4. Ratings	1. Keyword lookup grouping mechanism 2. Naïve Bayes 3. Decision Tress 4. Maximum Entropy
Panichella et al. (2016)	Manually derived	5	1. Information Giving 2. Information Seeking 3. Feature Request 4. Problem Discovery 5. Other	J48
Ciurumelea et al. (2018)	Manually derived	13	1. Device 2. Android Version 3. Hardware 4. App Usability 5. UI 6. Performance 7. Battery 8. Memory 9. Licensing 10. Price 11. Security 12. Privacy 13. Complaint	Gradient Boosted Tress
Dhinakaran et al. (2018)	Manually derived	4	1. Feature Request 2. Bug Report 3. User Experience 4. Rating	1. SVM 2. Naïve Bayes 3. Logistic Regression
Sorbo et al. (2016)	Manually derived	12	1. App 2. GUI 3. Contents 4. Pricing 5. Feature of Functionality 6. Improvement 7. Updates/Versions 8. Resources 9. Security 10. Download 11. Model 12. Company	Topic classification using WordNet and probabilistic classifier

From the information present in Table 5.1 it is evident that while studies inherit categories from the predecessor studies, there is no universal manually derived taxonomy to classify reviews, which raises the question, *are the taxonomies customised based on the types of reviews or the domain of the app(s)?,* another challenge that needs to be addressed. Another drawback of utilising a manually created taxonomy is the necessity to update the domain knowledge to create a new version of the taxonomy when the app evolves and new reviews are logged by the end-users (Pagano & Maalej, 2013; Peng et

al., 2012). To address such drawbacks, we have taken inspiration from well-known studies from several domains to develop an automated taxonomy generation approach to classify useful reviews which leads to the following research question

> RQ3.1 How can an approach be developed to automatically generate a taxonomy for classifying useful reviews, and how will such taxonomy compare to a manually developed one?

5.3 Classification Approach (RQ3.1)

The initial objective is to automatically generate a taxonomy to classify useful reviews and in this section we provide the details of the concepts and method that lead to the generation of such taxonomy from useful reviews. The performed investigations in this section answers RQ3.1.

5.3.1 Feature Engineering

Feature engineering helps to identify the relationship between a particular product's market characteristics and its features (Brunetti & Golob, 2000). However, performing feature engineering for app reviews is challenging because of the way in which the reviews are expressed by the end-users and thus, the presence of domain knowledge is required to identify the features (e.g., distance feature that indicates the possible distances between two routes that connect locations X and Y) and their associated market characteristics (e.g., requests, bugs or enhancements) (Ko et al., 2000; Licorish et al., 2017; Liu, 2000). To achieve this, researchers have used parts of speech (POS) tagging method which uses grammar concepts to identify the essential markers that represent the product features and their market characteristics (Cysneiros & do Prado Leite, 2004; Ko et al., 2000; Licorish et al., 2017; Zhang & Liu, 2011). For instance, Ko et al. (2000) have identified nouns as software product features and adjectives and verbs as the requests, bugs or enhancements in the domain knowledge provided by the domain experts to classify software product requirements expressed in natural language. Thus, we take inspiration and inherit guidelines from such studies to automatically generate a taxonomy to classify useful reviews and make an assumption that the nouns present in the useful reviews are app features, and adjectives and verbs are requests, bugs or enhancements related to the particular app feature (Ko et al., 2000; Licorish et al., 2017; Zhang & Liu, 2011). For example, consider the useful reviews, 'Distance (noun - app feature) is inaccurate (adjective - bug) and needs to be resolved (verb - request/enhancement)'. 'Map (noun - app feature) pixelated (verb - bug) continuously!'. In both examples, *Distance* and *Map* represent the app features, while words such as *inaccurate* and *pixelated* reflect bugs, and *resolved* reflects request or enhancement pertaining to the app feature *Distance*. Such patterns form the core of a potential taxonomy highlighted in Figure 20 generated via feature engineering (Htay & Lynn, 2013; Ko et al., 2000) and this taxonomy would assist towards classification of similar useful reviews sharing common characteristics (i.e., based on the presence of app features or their associated requests, bugs or enhancements) into the relevant groups of interest.

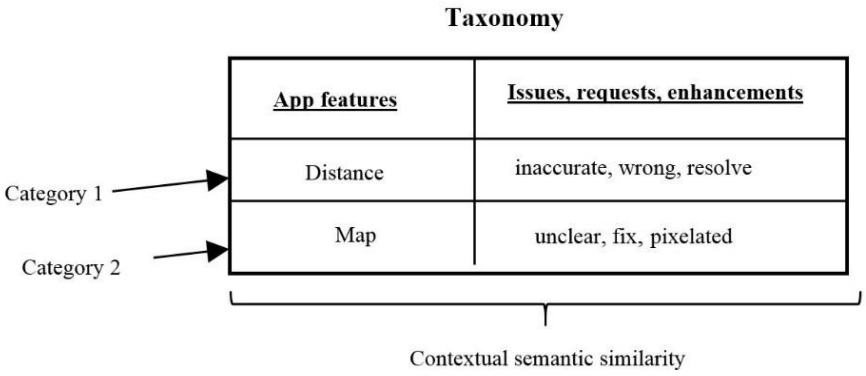


Figure 20. Example of a generated taxonomy

Given our objective to automatically generate a taxonomy independent of domain knowledge we utilise the POS tagging method from natural language processing to identify nouns, verbs and adjectives from the useful reviews (Hajič et al., 2009). Later, we identify adjectives and verbs which are semantically similar to the respective noun with regards to their contextual application to generate the taxonomy (Ko et al., 2000). Next, we report the investigation that lead towards the selection of a suitable semantic similarity method to assign similar useful reviews into groups.

5.3.2 Semantic Similarity Methods

There are several methods developed by researchers that determine semantic similarity between words which is quantified by a computed semantic score (Mihalcea et al., 2006). All these methods operate on the principle of word sense disambiguation that identifies the meaning of a word with reference to another based on its context of application measured via the computed semantic score (Karov & Edelman, 1998). Such methods may belong to multiple categories; (1) semantic similarity methods based on features, (2) semantic similarity methods based on graphical edges, (3) semantic similarity methods based on information theory and, (4) semantic similarity methods based on knowledge distribution (Sánchez et al., 2011). The methods from the first category determine the semantic similarity between words of interest by means of a dictionary (e.g., WordNet) (Petrakis et al., 2006) The methods falling in the second category, generate a graph based on the spread of word pairs according to a dictionary and later compute the semantic score of the word pairs based on the spread and distance values of the graph edges (Leacock & Chodorow, 1998). For the third category, methods utilise the knowledge of the distribution of words extracted from the information under scrutiny to determine the similarity between word pairs given an ontology derived from a dictionary (Jiang & Conrath, 1997). The methods from the fourth category use a common data source such as the world wide web (WWW), Wikipedia and so on as a dictionary to determine the semantic similarity between word pairs based on their co-occurrence (Bollegala et al., 2011). That said, as observed these methods use a dictionary which provides the formal descriptions of the words that can be compared for semantic similarity and hence, the suitable knowledge sources (e.g., dictionary) need to be obtained before the

methods compute semantic similarity scores. Thus, the prime disadvantage of such types of semantic similarity methods that are knowledge source based (e.g., dictionary, and online repository such as Wikipedia) is that they are entirely dependent on the existence of these knowledge sources (i.e., domain knowledge). In certain cases, it might not be possible to obtain the necessary domain knowledge or the domain knowledge from one domain may not be suitable for other domains. In fact, computing semantic scores between word pairs is far from perfect and is an ongoing challenge in the field of linguistics (Mihalcea et al., 2006). This is because of the variations in the way in which words are expressed by humans (Erk, 2010). This challenge lies at the heart of our proposed method to automatically generate a taxonomy from useful reviews. As mentioned in sub-section 3.6, we contend that the need to contact domain experts for gaining the necessary domain knowledge is problematic because of the differences in the way in which words are expressed by end-users of different types of apps that poses a challenge. For example, consider the word 'draining' whose meaning in a standard dictionary is 'liquid running out of a space' (Kozima & Furugori, 1993). On the contrary, in terms of useful reviews, 'draining' is related to the excessive consumption of a device's battery power. Moreover, the useful reviews contain words that are not covered by dictionaries. For example, urban words (e.g., hog, kill, drain - which are related to device battery), domain specific end-user generated words (e.g., spotting, capture, scan – which are related to camera use) and so on.

That said, given the nature of useful reviews, most often, the words in the useful reviews that are in close proximity of each other are contextually similar as the end-users who log the reviews often mention contextually semantically similar words in close vicinity of each other (Iacob & Harrison, 2013; Rohde et al., 2006). For instance, consider the useful review; 'not possible to accurately track route due to the wrong map'. This useful review indicates that the app is unable to accurately track the route because of the wrong map being loaded by the app. Of note here is that 'accurately', 'track', 'route', 'wrong', and 'map' are in close proximity to each other indicating their contextual semantic similarities. Such pattern is often repeated for many useful reviews indicating that in a vector space representation of words, semantically similar words are often close to each other because of their contextual application, while the irrelevant words are distant (Iacob & Harrison, 2013; Reisinger & Mooney, 2010). This forms the basis for the automatic generation of taxonomy which identifies the verbs and adjectives (i.e., requests, bugs or enhancements) that are semantically similar to the relevant nouns (i.e., app features) based on their context of usage. Hence, we reviewed semantic similarity methods that computed semantic scores of word pairs independent of domain knowledge.

To begin, the LSA (Latent Semantic Analysis) method initially constructs a word-document matrix, in which the words from a particular document *d* correspond to rows while the documents correspond to columns. Whenever a particular word *w* appears in a specific document, its frequency of occurrence is registered and updated in the respective word-document matrix's cell $C_{w,d}$ (Landauer & Dumais, 2008).

Thus, the matrix represents the dispersal of a word in the documents space. In the next stage, the rows of the matrix are normalised using an entropy-based normalisation method. Later, the semantic similarity between any two words existing in the word-document matrix is determined using the cosine distance measure. However, this method is often known to strongly depend on the learning data, thus causing substantial errors and further compromising its predictive judgments (overfitting) (Landauer & Dumais, 2008). More to this, in many cases the words, and documents represented by the word-document matrix data structure are interconnected by the Gaussian model, which makes LSA bias towards commonly occurring words (Evangelopoulos et al., 2012).

The probabilistic version of LSA tries to overcome the drawbacks of the LSA method (Hofmann, 1999). It achieves this by processing two conditional probabilities to determine the semantic similarity between two words. Firstly, it computes the probability of a word linked to a particular subject of interest, and secondly, it computes the probability of a document belonging to a given subject under a probabilistic model (e.g., Bayesian probability model). Finally, the occurrence of a word in a given document can be determined by the probability of the occurrence of a particular word related to the subject of interest, and the probability that the subject is related to the document under investigation. However, this model does not entirely solve the problem of overfitting (Leksin & Vorontsov, 2008). Furthermore, LSA operates with the assistance of the word-document matrix data structure that is only useful in determining the relevance of a word from the document's perspective, and not in terms of its contextual semantic similarity with other words, as necessary in case of useful reviews (Deerwester et al., 1990).

Another method named HAL (Hyperspace Analogue to Language) developed by Burgess (1998) measures the semantic similarities between two words based on their proximity in vector space. This method was developed based on the concept of representing each word in a vector space which assists in understanding the similarities between two words by calculating the pairwise distances between the points symbolised by the respective vectors. These vectors are created from the information on words' co-occurrences within a text corpus. The vector space[12] of words is a word-word matrix that indicates the semantic score of a particular word pair based on the vector distance of the two words from each other in a particular text corpus (Lund & Burgess, 1996). To achieve this, the authors first run a window of a size of ten words i.e., accommodating ten words in a single parsing operation, then moving from one word to another to repeat the same set of operations recursively. This mechanism is used to create a co-occurrence matrix of words present in the entire text corpus. For instance, for each word w1, the technique counts the number of times another word w2 occurs in close distance with w1. The counting is achieved using a weighted approach in which if w2 appears adjacent to w1, it assigns a weight of 10 (parsing window size), it assigns a weight of 9 if w2 is distant from w1 by one word, weight of 8 if w2

[12] Mathematical model for representing useful reviews as vectors wherein each dimension corresponds to a separate word. If a word occurs in the useful review, its value in the vector is non-zero. The dimensionality of the vector is the number of words in the useful reviews corpus.

is distant from w1 by two words, and so forth for a window of 10 word neighbours. When the entire corpus is traversed by the window, the word-word occurrence matrix's cell $C_{w1,w2}$ holds the weighted sum of all the occurrences of w2 in closeness to w1. Once the matrix is formed with all the necessary data, all the vectors being represented by the matrix are normalised to a fixed size, and finally, the similarity between two words' vector is determined using the Minkowski distance formula or Euclidean distance measure. However, it was found out that HAL was biased towards word pairs whose counts in the co-occurrence matrix were higher than the ones with rare or moderate counts (Rohde et al., 2006).

The strengths and weaknesses of LSA and HAL were studied by Rohde et al. (2006) to develop a new method COALS (Correlated Occurrence Analogue to Lexical Semantics) that inherited the strengths of HAL and LSA, and discarded their weaknesses. Unlike HAL, this approach uses four word window parser to create the words co-occurrence matrix. Once, the matrix is created, the counts are converted to correlations using the Pearson correlation function. Once the normalisation operation is complete, the negative values in the normalised words co-occurrence matrix are set to 0 (discarded), and the positive values are square rooted and retained in the matrix. This is done to prevent the method from being biased towards word pairs that are inversely related in terms of semantic similarity to prevent inaccurate results. Furthermore, in extensive empirical evaluations COALS outperformed the other methods as the semantic scores of several word pairs calculated by the method matched with those assigned by the domain experts (Rohde et al., 2006). Thus, we shortlisted COALS as the candidate method to evaluate the contextual semantic similarity between words of the useful reviews pertaining to an app and generate the required taxonomy. We describe this method below.

Initially, COALS creates a word-word co-occurrence matrix from the text corpus, using a window of size four. For each word w1, COALS counts the number of times every other word w2 occurs in proximity to w1, and stores the weighted count (i.e., total occurrences of a pair of words divided by total number of words) of the total occurrences of the relevant word pairs (w1 with w2) in the respective cell of the word-word (w1-w2) matrix. The ramped window of size four is responsible for generating the appropriate word counts. For instance, if w2 occurs adjacent to w1, the window assigns a count of four, if w2 is separated from w1 by one word, the window generates a count of three, and so forth, down to a count of one for a distance of three words. Finally, the word-word co-occurrence matrix portrays the weighted count of all occurrences of w2 in proximity to w1. In the next stage, the Pearson's correlation coefficient is calculated between the weighted vector counts of the occurrence of words w1 and w2, i.e., the word-word counts in the matrix are converted to correlations. This, in general, provides further insights into the vicinity of w2 with w1. Furthermore, with this context in the background, COALS converts all the negative correlation values in the matrix to zero and computes the square roots of the positive ones. The square root operation further normalises the matrix, thus making COALS unbiased towards larger positive values. The positive values of the matrix correspond to the word-word

pairs that convey a substantial amount of information. Finally, the semantic similarity score S of word pair (w1 and w2) is calculated using the data present in the normalised matrix as

$$S(w1,w2)= \sum_{i=1}^{n}(w1i - \overline{w1})(w2i - \overline{w2}) / \left(\left(\sum_{i=1}^{n}(w1i - \overline{w1})^2 \sum_{i=1}^{n}(w2i - \overline{w2})^2 \right) \right)^{1/2} \quad (12)$$

In equation (12), the value of i ranges from 1 to n and n indicates the maximum occurrence of the pair of words (i.e., w1i and w2i) together. Also, ($\overline{w1}$) and ($\overline{w2}$) indicate the average occurrence of words w1 and w2 in the calculated vector space. Since COALS operates only on positive values, the correlation distance measure is known to provide accurate results than the cosine measure, as correlations tend to be subtler than cosines (Rohde et al., 2006).

5.3.3 Pareto Principle

As COALS assisted in the automatic generation of the taxonomy based on feature engineering (refer to sub-section 5.3.1) we still faced the challenge of determining the number of categories for the taxonomy. The question encountered was, *do we consider all the app features (nouns) and their associated semantically similar requests, bugs or enhancements (adjectives and verbs) as categories for the taxonomy?* We address this challenge with the support of Pareto principle which gives the 80-20 rule that states that 80% of the contribution towards an outcome is given by 20% of its participating entries (Kiremire, 2011). The application of this principle is common in the software engineering discipline. For instance, Archak et al. (2007) have used the Pareto principle to identify 20% of the important software product features that influenced 80% of the software product sales. We take inspiration from these studies and utilise the Pareto principle to shortlist the necessary categories to generate the required taxonomy in which we identify the required number of categories that reflect 80% of the app features along with their semantically similar requests, bugs or enhancements. All the other app features are then classified in an 'Others' category. The detailed elaboration of the above-mentioned process is as follows, initially we sort the app features and their associated requests, bugs or enhancements in the descending order of the frequency of occurrences of the app features in useful reviews. Next, we compute the cumulative frequency based on the frequency of occurrences of the app features to compute the cumulative percentage to set the cut-off threshold (i.e., 80%) of the required number of categories representing the app features and their associated requests, bugs or enhancements. Thus, the Pareto Principal is used as an inspiration to identify the most frequently mentioned app features and these app features were identified based on cumulative frequency percentage where the cut-off threshold was set to 80%.

5.3.4 Keyword Lookup Classifying Mechanism

After determining the number of categories we utilise the keyword lookup classifying mechanism (i.e., basic string matching) to classify useful reviews into the relevant groups (Maalej et al., 2016a). A useful

review from a pool of useful reviews gets classified into a particular group if a word from the useful review matches with any word of a particular group (category) present in the taxonomy comprising of app features, bugs, requests, or enhancements. That said, if any useful review that is not classified in any group, it is classified in an 'Others' group. It is to be noted that the particular app feature in the taxonomy represents the name of the specific group and thus, it becomes easier for app developers to seek requests, bugs or enhancements pertaining to the particular app feature. Figure 21 provides a graphical illustration of the proposed classification approach mentioned above. After obtaining the useful reviews, the nouns, adjectives and verbs present in those are tagged (using NLP approaches), and are modelled as the basis for representing the categories of the taxonomy. Finally, the categories of the taxonomy are used for classifying useful reviews into different groups of interests (using the keyword lookup classifying mechanism).

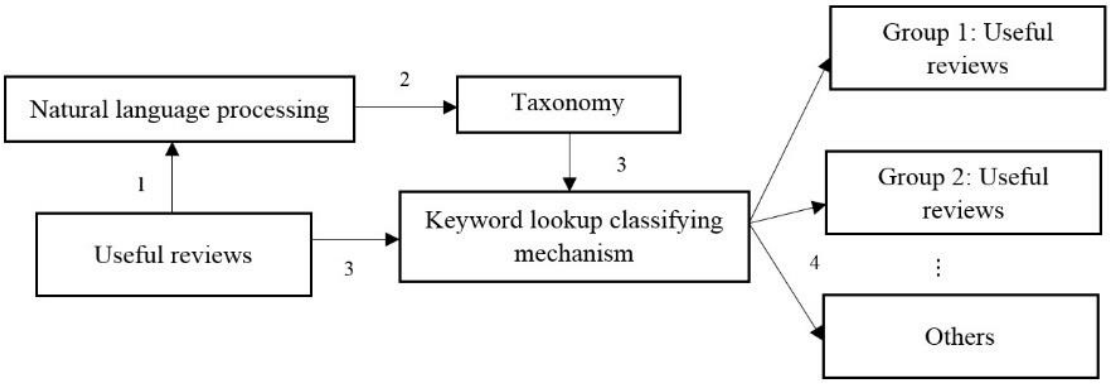


Figure 21. Proposed classification approach for useful reviews using an automated generated taxonomy

5.3.5 Generated Taxonomy Evaluation

In this sub-section we mention the steps that were used to validate the automatically generated taxonomy which were based on a qualitative content analysis approach (Mayring, 2004). Initially, the first noun entry from the automatically generated taxonomy is selected. We then check for the presence of the noun in the pool of useful reviews. The useful reviews containing the noun entry are selected for further analysis. Next, we manually analyse each useful review to determine the set of adjectives and verbs that are associated with the noun under analysis. After every useful review is analysed, we extend the list of adjectives and verbs (in the useful reviews) that are relevant to the noun under scrutiny. Finally, the manually finalised adjectives and verbs pertaining to the specific noun entry are compared against those present in the automatically generated taxonomy, where the accuracy is computed. In this scenario, accuracy indicates the percentage of adjectives and verbs that are common to both the automatically generated taxonomy and manual outcomes. The entire process is repeated for all the noun entries present in the automatically generated taxonomy until no noun entry is left for evaluation. After the manual evaluation process is completed, an overall average accuracy percentage is computed. In addition, the two supervisors and the PhD candidate independently performed the manual evaluation of

the automatically generated taxonomy using the useful reviews by following the above-mentioned evaluation process. Later, reliability assessments were conducted to resolve any disagreements and establish consensus to compute average accuracy. The average accuracy percentage reflects the average accuracy percentage of all the evaluated noun entries which ultimately indicates the overall accuracy of the automatically generated taxonomy.

5.4 Experimental Settings

In this section, we provide the details regarding the procedures that were enacted to drive our experiment and validate the primary outcome of the automated taxonomy generation based on the useful reviews classification phase. First, we provide a brief description of the dataset that was used for the pilot experimentation purpose. We then provide the details of the pre-processing and POS tagging operations that were performed. Thereafter, we provide details regarding the evaluation procedure followed to validate the taxonomy generated by COALS.

5.4.1 Dataset

To demonstrate and evaluate the approach to automatically generate a taxonomy for classifying useful reviews we utilise the My Tracks dataset. In addition, the My Tracks dataset was selected in the pilot studies of taxonomy generation (Phase 3) as well as prioritisation (Phase 4) as the two supervisors of this undertaken PhD work have previously provided software maintenance insights for the developers of this app, and thus this software provides a good baseline for comparing our outcomes in the pilot studies. That said, a set of 855 useful reviews were identified and extracted from this dataset for further experimentation (refer to sub-section 4.5 for more details).

5.4.2 Useful Reviews Pre-processing and POS Tagging

Initially, we performed the basic useful reviews pre-processing operations mentioned in sub-section 4.3.1. That said, the first task of this experiment was to identify nouns, adjectives, and verbs from the pre-processed useful reviews. To achieve this goal, we use the average perceptron POS tagger as it often outperforms the other types of POS taggers and is known to be scalable for domain specific text corpus (Hajič et al., 2009). After tagging the nouns, adjectives, and verbs in the pre-processed useful reviews, we provide the tagged useful reviews (e.g., GPS – NOUN, inaccurate – ADJECTIVE, drain – VERB) as input to COALS. Finally, the useful reviews were classified based on the generated taxonomy through COALS and evaluated the automatically generated taxonomy using the procedure mentioned in sub-section 5.3.5.

We provide results of this pilot study in the Results section.

5.5 Results

In this section, we report the results of the pilot study that reflects the evaluations related to the classification of useful reviews. We compare the automatically generated taxonomy against a manually developed taxonomy for the My Tracks dataset for evaluation. This outcome provides context for answering RQ3.1, and provides triangulations for RQ3.

5.6 Automatically Generated Taxonomy Validity

We evaluated the accuracy of the automatically generated taxonomy which consisted of 152 categories as mentioned in this Chapter (refer to sub-section 5.3.5). Prior to this, we had applied the Pareto principle on the result generated by COALS to identify the necessary categories constituting the automatically generated taxonomy required for classifying useful reviews. The Pareto principle returned 152 categories that depicted respective nouns along with their associated adjectives and verbs. This outcome indicating the followed steps (refer to sub-section 5.3.3) to shortlist the required number of the categories that reflect 80% of the app features along with their semantically similar requests, bugs or enhancements is made available online[13] where it can be observed that the top 152 categories are identified based on 'Cumulative Percentage' column (i.e., the application of Pareto Principle accounted for 152 app features out of 981 (15.49%)). A subset of the automatically generated taxonomy is visualised in Figure 22 where ten prominent app features sharing dependencies with each other via a common set of requests, bugs or enhancements are depicted. In the undirected graph, each node represents an app feature and the information on the links represent the requests, bugs or enhancements. For example, it seems that the travel or workout data provided by the 'stats' (statistics) feature of My Tracks app and the 'map' (app feature) selected for travel or workout are 'unreadable' to the app's end-users. Other conclusions of interest can be drawn from the visualisation. For instance, the relationship between GPS and signal (two nouns representing app features) was described using verbs such as fluctuate, drop, and lose.

[13] https://tinyurl.com/y5bq3vrh

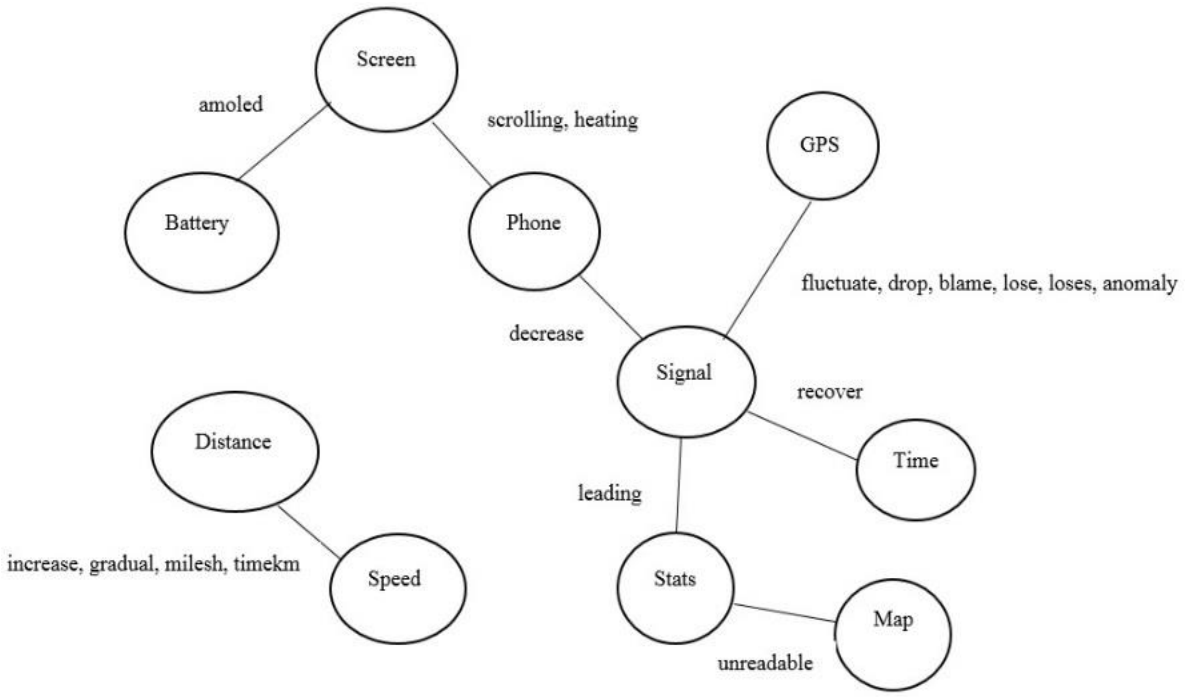


Figure 22. Visualisation of partial taxonomy consisting of ten prominent app features

The overall accuracy of the automatically generated taxonomy was found to be 72% which indicates a substantial match between the manual taxonomy created by us and the automatically generated taxonomy (Košmerlj et al., 2015). The two supervisors and PhD candidate followed the taxonomy evaluation process mentioned in sub-section 5.3.5 of this Chapter. Each evaluator independently analysed the 855 useful reviews to identify the requests, bugs, and enhancements (adjectives and verbs) related to the app features (noun) to finalise the contents of the manual taxonomy. With regards to the reliability assessment practise followed in this study, a substantial agreement of 0.62 was observed between the evaluators. The reported Fleiss coefficient indicates the agreements on the adjectives or verbs associated with the particular nouns. Follow up discussions were held among the supervisors and the PhD candidate to resolve any disagreements to establish consensus. After the consensus were established, the finalised manual taxonomy was compared with the automatically generated one to compute the overall accuracy. Table 5.2 shows the partial manual taxonomy generated for the app features presented in Figure 22.

Table 5.2 Partial view of manually derived taxonomy

App features	Requests, bugs or enhancements
GPS	drop, recognise, loses, loose, fluctuate, anomaly, blame, picked, regain
Map	destroyed, unreadable, lagging, nonresponsive, offline, preloading
Time	pausing, recover, reconnect, unsync
Battery	eat, amoled, drain, kill, consume, wasting, flatten, lowered
Phone	accessible, heating, decrease, scrolling
Distance	travelled, jagged, counted, incorrect, wrong, measured, overestimated, increase, timekm, gradual
Stats	aggregate, leading, unreadable, grouped, overview
Signal	recognise, fluctuate, decrease, drop, lose, leading, loses, anomaly, blame, regain, recover, dotted
Screen	lock, scrolling, unresponsive, amoled, heating, smaller, sliding, lag
Speed	increase, gradual, colorcode, jogging, calculated, traditional, kmh

In addition, the accuracy of the keyword lookup classifying mechanism was found to be 98.3%, the slight imperfection was due to the presence of misspelled words in the useful reviews.

In the next section, we provide the discussion related to the undertaken pilot study that deals with the automatic generation of a taxonomy for classifying useful reviews.

5.7 Discussion

The scalable requirements prioritisation method proposed by Peng et al. (2012) classified the requirements into groups of interest based on the domain knowledge (groups and their associated keywords of interest). The essential domain knowledge was provided by experts before the method prioritised the groups of interest using the stakeholders' priority preferences on individual requirements. As stated earlier (refer to Chapter 3, section 3.6), this method was one of the inspiration sources towards our proposed group-based prioritisation method. However, as the app domain is vast, it is not possible for us to gather the required enormous domain knowledge needed for classification or prioritisation (refer to Chapter 3, section 3.6). Thus, we reviewed studies from the app domain that provided context regarding classification of reviews pertaining to the apps. Our investigation of these studies revealed that all the proposed classification approaches from the app domain were dependent on the domain knowledge made available by experts, and there was no universal taxonomy encountered for classification purpose (Ciurumelea et al., 2018; Maalej et al., 2016a; Panichella et al., 2015). This was not suitable for our research and was identified as a critical research gap that lead us to propose an approach that automatically generates a taxonomy for classifying useful reviews with the intent of addressing the research gap. The sub-section below discusses the results and implications of RQ3.1.

5.7.1 RQ3.1 How can an approach be developed to automatically generate a taxonomy for classifying useful reviews, and how will such taxonomy compare to a manually developed one?

The outcomes reported in the pilot study show that it is possible to develop an approach that automatically generates a taxonomy to classify useful reviews into dynamically created groups of interest. This approach is directly able to extract app features and their associated requests, bugs or enhancements from a corpus of useful reviews without the necessity of human involvement and domain knowledge. This has a potential implication for supporting software maintenance and evolution cycles where a small group of app developers have to manually analyse numerous useful reviews. We believe the key aspect towards the development of the automatically generated taxonomy is the selection and utilisation of suitable concepts and methods from multiple domains (Ko et al., 2000; Maedche & Staab, 2000; Rohde et al., 2006; Turner et al., 1999). While natural language processing application involving POS is widely utilised, the level of human involvement in labelling large numbers of useful reviews in support of manually generating taxonomies for classification is a potential challenge (Maalej et al., 2016a). Feature engineering assisted us in developing a suitable taxonomy framework for constituting the automatically extracted domain knowledge (i.e., app features and their associated requests, bugs or enhancements) from the corpus of useful reviews, thereby solving a significant research problem that is evident for manually generated taxonomies which is the need to develop categories. With regards to this, our primary objective was to determine the requests, bugs or enhancements (adjectives and verbs) that were semantically similar (contextually similar) to app features (nouns) for which we evaluated COALS, where COALS directly operated on the distances of vector data belonging to the respective word pairs. It is to be noted that, in our research the relationship between the words are determined based on the primary principle of word sense disambiguation (Karov & Edelman, 1998). The application of a reliable contextual semantic similarity method such as COALS addresses a limitation that is observed for manually generated taxonomies which is the appropriate data for the categories of a taxonomy (Walid Maalej & Haader Nabil, 2015). Taxonomies generated by experts provide a limited number of categories, and hence, classification results often provide a holistic view of grouped reviews which is inappropriate if there are numerous useful reviews. In addition, the application of the Pareto distribution law seems useful in determining the prominent categories for the taxonomy and at the same time, prioritising the most significant categories (i.e., app features and their associated requests, bugs or enhancements) while still retaining an ‘Others’ category (Archak et al., 2007). That said, the keyword lookup classifying mechanism provides a near perfect classification of useful reviews in completing the automatically generated taxonomy which may be used as an inspiration for other software engineering research that focus on app reviews.

Furthermore, the automatically generated taxonomy compared substantially to the one that was developed manually. There was an overlap of 72% observed in the two taxonomies which suggested that the combination of concepts and methods provided an intuitive automated solution that closely aligned with human thinking. This is noteworthy as our proposed approach is in its preliminary stage and the utilised methods have not been refined or tuned for optimisation (e.g., tuning the threshold settings of COALS) which could lead to potential improvements (Konkol et al., 2015). For instance, in a recent study, Konkol et al. (2015) have integrated COALS with singular value decomposition (SVD) and subjected COALS to specific SVD parameters (careful tuning) to generate optimal data required for performing named entity recognition using latent semantics. That said, we believe that the fine-grained taxonomy that was generated automatically provides an explicit view of the prominent app features and their associated requests, bugs or enhancements for the app developers. Thus, app developers may directly utilise the generated taxonomy to identify app features that require immediate attention based on the requests, bugs or enhancements associated with these app features without the need to perform classification. Moreover, such a taxonomy indirectly represents the prioritised app features due to the application of the Pareto distribution law, as the app features (nouns) constituting the categories are arranged in descending order of prominence (based on frequency of nouns) (Licorish et al., 2017). In fact, the partial taxonomy presented in Figure 22 (refer to section 5.5) reveals that certain app features share common set of requests, bugs or enhancements. Such finding is crucial to app developers, as it would significantly assist them in uncovering dependencies among the app features. This in turn could assist in identifying the influence of one app feature on another based on the common characteristics (related requests, bugs or enhancements) that are shared among the app features (Li et al., 2012). Furthermore, based on the observed hierarchical dependencies among the app features, resolving certain requests, bugs or enhancements associated with specific app feature will reduce the burden of defects on the dependent app features. That said, the proposed approach of automatically generating a taxonomy to classify useful reviews requires limited human involvement and provides a wide spread of categories naturally. To conclude, the empirical evaluations conducted in the pilot study showed satisfactory result when the outcome (automatically generated taxonomy) of our proposed approach was compared against the one that is manually derived albeit we have used a single dataset. Therefore, our proposed automated taxonomy generation approach may be promising for the software engineering community.

5.8 Threats to Validity

In this section, we present the threats to validity that can potentially affect the outcomes reported in this Chapter.

5.8.1 Internal Validity

The pilot study conducted in this phase is limited to the grouping of useful reviews based on an automatically generated taxonomy. However, we have performed evaluation of the automatically generated taxonomy for triangulation. That said, coming out of the text pre-processing and POS tagging pipeline, it was not feasible to evaluate the nouns, adjectives and verbs that do not reflect app features, issues, suggestions, or requests, or those that were misclassified. This was largely due to the high levels of overhead involved with other rigorous manual evaluations that were performed. Furthermore, our proposed automatic taxonomy generation approach may potentially leave out some important app features that are less frequently requested. In addition, there could be presence of synonyms (e.g., track, tracker and so on), and misspelled words with associated bugs, requests or enhancements that could point to the same app feature. Thus, the size of the automatically generated taxonomy might increase, and such taxonomy may hold redundant information expressed in different forms. Concerning these, there is scope for future research to address the issues related to the presence of synonyms in the taxonomy or missed out less frequent but prominent app features. One potential solution towards resolving these issues would be to involve domain experts (i.e., app developers) to select the prominent app features of interest. Finally, investigations done using manual analysis are always criticised for subjectivity. We have worked to remove this threat by performing reliability assessments where substantial agreements were observed.

5.8.2 External Validity

We have used one dataset in this study, which may affect the generalisability of this study. However, the accuracy of the generated taxonomy reported for the app is substantial in terms of the validation of the automated taxonomy generation approach.

5.8.3 Construct Validity

The Pareto distribution law returned a significant number of categories for the generated taxonomies, which may seem excessive. That said, our manual evaluation confirmed that these categories were largely relevant. One way to limit the number of categories in the taxonomy is to implement a cut-off mechanism (e.g., top 10). Concerning this, there is scope to research '*How can the optimal categories for different apps be identified?*' Furthermore, an alternative to the validation of the automatically generated taxonomy would be to approach the app developers of the respective apps to evaluate the requests, bugs and enhancements associated with the features of the app.

We provide the concluding remarks of this phase, research contributions and summary of implications in the Conclusions chapter (refer to Chapter 7). In the next chapter, we present the details of Phase 4 (i.e., prioritisation of useful reviews).

6 Prioritisation of Useful Reviews

This chapter describes phase 4 of the undertaken research work in which we developed and experimented with a group-based prioritisation method that utilised the outcome of phase 3 (i.e., classified useful reviews into specific groups of interest) before generating the priorities of the classified useful reviews and their respective groups. In the same phase, we also developed an individual prioritisation method where the priority of each useful review was computed without performing classification. This phase answered RQ4 which is comprised of two research questions. The formulated RQ 4.1 and RQ 4.2 benchmarked the performance of the group-based prioritisation method as well as the individual prioritisation method to validate the application of the respective methods. Based on our findings of the pilot study conducted in this phase, we performed a full-scale study of the individual prioritisation method in phase 4 to demonstrate its general suitability across a range of apps (i.e., to show the comprehensive application of the method).

6.1 Automated Prioritisation Methods (RQ4)

After developing the required classification approach, our next step was to prioritise the classified useful reviews and their groups for which we utilised an automated hybrid prioritisation method. The automated hybrid prioritisation method reflects a multi-criteria heuristic function comprising of four prominent methods incorporated as variables in the function to prioritise the classified useful reviews and their groups. We provide all the details regarding this prioritisation method in sub-section 6.1.1. That said, as we have highlighted several limitations of the prioritisation methods in Chapter 2 (refer to section 2.1) and in Chapter 3 (refer to section 3.6) we had to seek inspiration from other domains such as feature engineering, information theory, information retrieval, marketing and artificial intelligence to develop the automated hybrid prioritisation method and benchmark its performance (Chea et al., 2009; Dasgupta et al., 2013; Fang & Zhan, 2015; Filcek et al., 2017; Htay & Lynn, 2013; Sundaram et al., 2005; Zhang & Tran, 2008). The RQ related to benchmarking the performance of the prioritisation method is

RQ4.1 What is the performance of the developed group-based prioritisation method?

6.1.1 Group-based Prioritisation Method

As mentioned in Chapter 3, section 3.6, the group-based prioritisation method inspired by the requirements prioritisation method proposed by Peng et al. (2012) was to be developed in such a way that it would be independent of domain knowledge and priority preferences of the stakeholders. Given the numerous useful reviews, our objective is to generate the required priorities so that the useful reviews can be addressed accordingly. In this sub-section, we mention the key concepts and methods that lead towards the development of our proposed automated hybrid prioritisation method.

6.1.1.1 Keywords of Interest

Referring to sub-section 5.3.1, the distinguishing nouns, adjectives and verbs identified from useful reviews are termed as keywords of interest (K) as these keywords of interest represent quantitative end-user feedback properties that have significant impact on a product's requirements engineering phase as such keywords of interest hold noteworthy meaning and eliminate the need for the availability of domain knowledge required for taxonomy generation or prioritisation (Chea et al., 2009; Htay & Lynn, 2013; Ko et al., 2000). In the next sub-section, we mention the methods that utilise the knowledge of such keywords of interest to generate the priorities of the useful reviews and their groups.

6.1.1.2 Methods

Studies have shown that the complex problem to prioritise requirements can be appropriately solved using multiple criteria as considering a single criterion does not guarantee exact or approximate exact prioritisation solution (Achimugu et al., 2014b; Garg et al., 2017). For instance, Asghar et al.'s (2013) prioritisation method on average was found to be 16% accurate when the method considered code metrics as a single criteria to prioritise requirements. On the contrary, AHP has proved to generate accurate prioritisation results because of its ability to incorporate multiple criteria along with the priority preferences of the stakeholders that significantly influence the priorities of the requirements. However, AHP is known to suffer from scalability and computational complexity issues due to its pairwise comparison mechanism (Achimugu et al., 2014b). Hence, taking inspiration from studies belonging to renowned domains such as information theory, information retrieval, marketing and artificial intelligence, we identify four prominent methods for prioritising useful reviews and represent them as criteria by encompassing them as variables in a multi-criteria heuristic function with the objective of generating the approximate exact solutions, i.e., priorities of the useful reviews. We provide a brief elaboration of each method in the next sub-sections.

6.1.1.2.1 Entropy

In information theory, entropy is a measure of information that is widely used to acquire knowledge about an entity of interest existing in vast information (Shannon, 1948). Knowledge gained through entropy indicates the product features of prime interest to its customers whose identification and weightage is essential to drive the development process of a product. For instance, Somprasertsri and Lalitrojwong (2008) have used entropy to automatically extract and prioritise product features from product reviews that required attention. Similarly, Zhang et al. (2008) have used the entropy to prioritise product reviews based on the helpful information conveyed by the reviews. In our study, the key objective of the entropy is to generate the priorities of the useful reviews based on the quantified measure of information conveyed by a useful review (R) in proportion to the information present in the entire corpus (C) of useful reviews. The priority E_R of R through means of entropy is given as

$E_R = - \sum_{i=1}^{n} P(K_i) * \log_2 P(K_i)$ (13)

Where K (K ∈ R) denotes the keyword of interest contained in R (R ∈ C), and the proportion P(K) is given as the total occurrences of K in R to the total occurrences of K in C. Furthermore, entropy enables the characterisation of the information through the probability distribution of the keywords of interest drawn from C. The probability distribution of the keywords of interest associated with the information quantity of every K forms a random variable whose average estimate is the average amount of information generated by the probability distribution (Rényi, 1961).

6.1.1.2.2 Frequency

Most often, the requirements elicitation phase captures the frequently stated stakeholders' requirements (Groen et al., 2015; Hosseini et al., 2015; Solemon et al., 2008). Similar is the case observed for useful reviews and researchers have exploited such knowledge for prioritisation purpose (Chen et al., 2014; Licorish et al., 2017). For instance, Licorish et al. (2017) have prioritised app features based on their frequency of occurrences in reviews with the assumption that end-users report buggy or most needed app features on a regular basis. Hence, taking inspiration from such studies, we utilise the keywords of interest frequency information to prioritise R belonging to C. We generate the priority of R as the summation of the frequency values associated with the respective keywords of interest contained in R (i.e., number of times each K in R appears in C). For example, consider keywords k_1, k_2, k_3, k_4,…, k_n present in C, and let fk_1, fk_2, fk_3, fk_4,…, fk_n represent their respective frequency of occurrence values, then priority F_R of R is generated as

$F_R = \sum_{i=1}^{n} f K_i$ (14)

Wherein K denotes the keyword of interest contained in R where K ∈ R and R ∈ C. Hence, in this study, the frequency method assists in prioritising R based on the frequently mentioned K that useful reviews captures from the end-users.

6.1.1.2.3 TF-IDF

TF-IDF (Term Frequency - Inverse Document Frequency) is a popular method that has a wide range of applications in the information retrieval field, especially to prioritise documents (e.g., sentences) present in a text corpus based on the importance of words that reside in those documents (Wu et al., 2008). Moreover, in the past TF-IDF method has been utilised for tracing and prioritising requirements present in a text corpus as well as to determine the significance of end-users' reviews by prioritising the requirements (Kim et al., 2006; Sundaram et al., 2005). Based on these studies and considering the functioning mechanism of TF-IDF, this method seems fit to generate priorities of useful reviews. Thus, given the useful reviews, we utilise the TF-IDF method to determine the TF-IDF weights of all the keywords of interest present in R residing in C. Wherein, C exhibits the role of the entire useful reviews

corpus and R exhibits the role of a document. The keywords of interest (terms) present in each R are subjected to TF-IDF method to determine their respective TF-IDF weights. Initially, we compute the term frequency count of each K in every R present in C. Term frequency count is the ratio of the number of times the K appears in R to the total number of keywords of interest present in R. Hence, every R will maintain its own term frequency count data structure, which is given as

$$Kf_{K,R} = n_{K,R} / \sum n_{K,R} \quad (15)$$

where $Kf_{K,R}$ indicates the number of occurrences of the K in R. Next, we compute the inverse document frequency term, which generates the weight of keywords of interest based on their spread across all the useful reviews in C. The inverse document frequency equation is given as

$$idf(K) = \log_{10} (NR/df_K) \quad (16)$$

where idf indicates the number of useful reviews containing K, NR indicates the number of useful reviews and df_K indicates the number of occurrences of K in those useful reviews. Next, we compute the TF-IDF score of K in R, which is given as

$$TF\text{-}IDF_{K,R} = Kf_{K,R} * idf(K) \quad (17)$$

Finally, the TF-IDF weights of all the keywords of interest present in R are summed up to determine the priority ($TF\text{-}IDF_R$) of R given as

$$TF\text{-}IDF_R = \sum_{K \in R} TF\text{-}IDF_{K,R} \quad (18)$$

Therefore, the objective of TF-IDF method in this study is to determine the importance of keywords of interest in R, given the collection of useful reviews in C. As observed from equations (15) to (18), the logarithmic values of the inverse reviews frequencies are considered as the keywords of interest frequencies and are distributed exponentially, thus generating a suitable weight concerning K's importance in R belonging to C (Wu et al., 2008).

6.1.1.2.4 Sentiment Analysis

In the marketing domain, the commercial value of a product is derived from the end users' sentiments affiliated with the product reviews and sentiment analysis is a method that has been widely utilised by researchers to investigate this aspect (Fang & Zhan, 2015). Sentiment analysis aids in measuring the content or discontent of end-users regarding their usage of the product, and significantly assists in flagging the requirements that raise concerns such as end-users' requests, suggestions or issues related to the product (Das et al., 2012; Galvis Carreño & Winbladh, 2013). For instance, Zha et al.(2014) have utilised sentiment analysis method to prioritise the concerns raised about a product by giving appropriate priority to the reviews that reflected a high level of negative end-users' sentiments as the

authors found that majority of such reviews reflected the raised concerns related to the product. Similarly, sentiment analysis has been performed on the reviews present in the app domain to substantially support the requirements engineering cycle associated with the development of apps to launch long term market sustainable apps in the app market (Goul et al., 2012). In this study, we utilise VADER (Valence Aware Dictionary and sEntiment Reasoner) to calculate the sentiment associated with R, as this tool has been empirically evaluated to perform significantly better than other tools in estimating the sentiments of reviews present in crowdsourced information (Hutto & Gilbert, 2015). The foundations of VADER are built on a human-centric approach for determining sentiments by combining qualitative analysis and empirical validation. Hence, VADER's sentiment analysis is sensitive towards polarity (positive or negative) and intensity of the particular emotion (i.e., anger, sad, happy, and so on) expressed in reviews.

To elaborate further, VADER's sentiment determining approach is based on lexicons[14] of sentiment-related words. To determine the polarity of these words, the developers of VADER utilised Amazon's Mechanical Turk[15] platform to get polarity (and optionally, to what degree) of the numerous words existing in crowdsourced information (such as reviews) from several human evaluators. Thus, VADER has a wide coverage of words and there is a substantial fit between the lexicon and the words mentioned in the reviews, and can return results of sentiment analysis faster than other sentiment analysis approaches (Hutto & Gilbert, 2015).

We illustrate the working of VADER with an example. Consider the review 'The product is good and it has nice features.' Initially, when VADER analyses this review it performs a check to determine if any words in the review are present in its lexicon. In case of the review, the review has two words in the lexicon (good and nice) with the positive polarity of 1.9 and 1.8 strength respectively. After analysis, VADER generates three sentiment scores from these words' polarities. For instance, assume that for the given review example, the review gets rated 45% positive, 55% neutral and 0% negative. Then, VADER computes a compound score[16] that is the sum of all the lexicon polarities (i.e., 1.9 and 1.8 in this case) and normalises the final score in the range [-1, 1]. In the mentioned example, the review gets a compound score of 0.69 that is termed to be substantially positive.

Based on this tool, we generate priority (SC_R) of R to measure the sentiment intensity of R present in C. The VADER sentiment analysis tool generates sentiment scores in the range [-1, 1]; -1 for the review with the most negative sentiments attached to it, and 1 for the review with the most positive end-user sentiments embedded in it. The useful reviews having a higher degree of the negative score are crucial than the ones having a positive score, and need to be addressed prior as such app reviews raise serious

[14] Vocabulary of language or a particular branch of knowledge
[15] https://tinyurl.com/8xzen9
[16] https://github.com/cjhutto/vaderSentiment

product concerns (Goul et al., 2012; Licorish et al., 2017). This consideration is modelled by -(SC_R) in this study.

Finally, to maintain the same range of the priority scores generated by the four methods and to prevent the prioritisation method from being bias towards the method generating larger range of priority values, we perform the min-max normalisation of the priority scores generated by E_R, F_R, TF-IDF_R, and -(SC_R) (Patro & Sahu, 2015).

6.1.1.3 Multi-Criteria Heuristic Function

Multi-criteria based heuristic functions are seen as cognitive tools that assist significantly in solving complex problems by generating approximate solution that is dependent on multiple criteria, and further assist in balancing the trade-off between the solution generation time, optimal nature of the solution, and the accuracy of the solution complemented by its completeness (Dasgupta et al., 2013; Filcek et al., 2017). For instance, the heuristic based approach has been proved successful to generate an optimal solution (out of 'n' possible solutions) for the traveling salesperson problem (NP-hard) (Lin & Kernighan, 1973). Since our objective is to generate an approximate optimal priority of R using the prominent methods mentioned in the previous sub-sections, we incorporate all the methods into a multi-criteria heuristic function ***f*** and represent those methods as variables of the function. Thus, the overall priority P_R of R is given as

$$P_R = \boldsymbol{f}:\ \alpha E_R + \beta F_R + \gamma \text{TF-IDF}_R + \delta(-(SC_R)) \tag{19}$$

In (19), E_R represents the priority of R generated by the entropy variable, F_R indicates the priority of R generated by the frequency variable, TF-IDF_R indicates the priority of R generated by the TF-IDF variable and -(SC_R) indicates the priority generated by the sentiment variable. In addition, we introduce four constants α, β, γ, and δ in the multi-criteria heuristic function to support the future prospects of performing manual or automated optimisation of the function to improve its efficiency as required (i.e., reducing computation time, increasing accuracy of prioritisation, prioritising useful reviews based on business requirements, and so on) (Blot et al., 2017; Marler & Arora, 2004). However, while performing the optimisation, the values of α, β, γ, and δ should be set in such a way that it satisfies the constraint $\alpha + \beta + \gamma + \delta = 1$. Currently, following the conventions of recommended settings, the individual values of α, β, γ, and δ are set to 0.25 as default (seed) values (Arcuri & Fraser, 2013). Figure 23 illustrates the computation of P_R pertaining to useful reviews. Initially, the keywords of interest are identified from the corpus of useful reviews. For a particular useful review, E_R is generated using equation (13), F_R is generated using equation (14) and TF-IDF_R is generated using equation (18). As noted earlier, these respective variables operate on the appropriate keywords information made available through the identified keywords of interest. To generate -(SC_R) the entire non pre-processed useful review is processed by VADER. After E_R, F_R, TF-IDF_R, and -(SC_R) of all the useful reviews have been computed,

E_R, F_R, TF-IDF$_R$, and -(SC_R) are subjected to min-max normalisation respectively. That is, all the priority scores generated by E_R are normalised followed by those generated by F_R, TF-IDF$_R$, and -(SC_R). Next, the respective weights of α, β, γ, and δ are multiplied with the normalised E_R, F_R, TF-IDF$_R$, and -(SC_R) of the useful reviews after which the final P_R of each useful review is generated using equation (19).

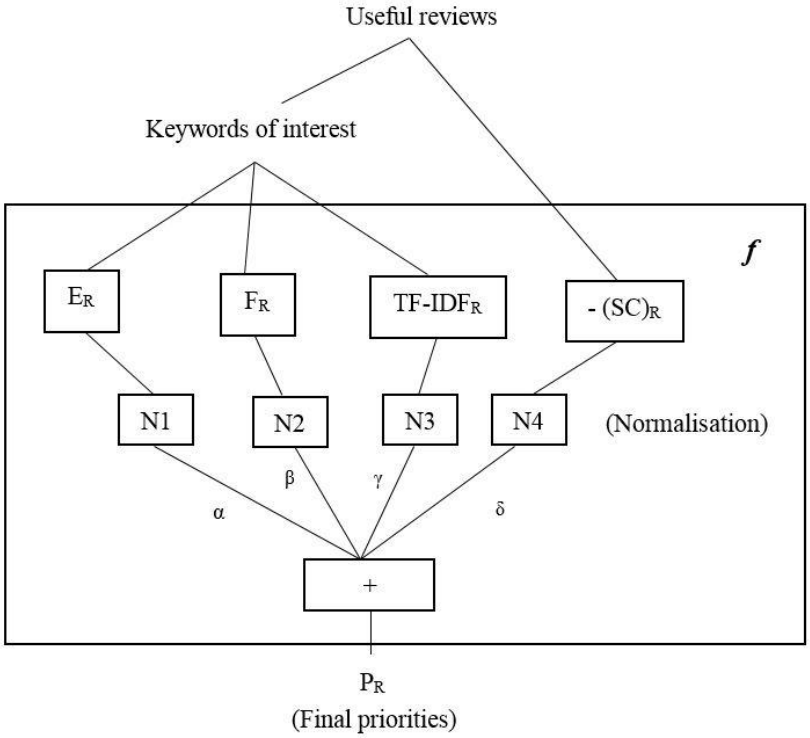


Figure 23. Diagrammatic representation of heuristic function *f* generating priorities of useful reviews

6.1.1.4 Group Priority

In statistics, weighted average method is utilised by researchers to measure the centre of a frequency distribution which is influenced by all the samples within a population and the result generated by the weighted average is termed as a reliable measure of central tendency when generating inferences from a general population (James et al., 2013). For instance, in Geology weighted average is considered as a significant statistical measure when determining the overall intensity of earthquake for a particular region based on the earthquake's previous frequency of occurrences, and Richter scale readings (Allen, 1986). In this study, we chose the weighted average method to generate the priority of a group (G). For instance, if a group has three useful reviews with individual P_R of 0.80, 0.90 and 0.90, then the group priority will be 0.87. Doing so will enable the app developers to gain insights on the overall magnitude of the priorities that are generated for useful reviews residing within a group. In addition, the groups with higher weighted average priorities would reflect alarming useful reviews and this would aid towards the standardisation of addressing the useful reviews based on the computed group priority. For

example, a standard approach that can be followed by app developers is to give preference to addressing the useful reviews of a group that has the highest priority over the others (Peng et al., 2012).

6.1.1.5 Elimination of Duplicate Useful Reviews

It is to be noted that based on the taxonomy that is generated, a single useful review can be classified into multiple groups. This is due to the app features being interlinked through the means of common requests, bugs or enhancements (Li et al., 2012). For example, consider the useful review 'map keeps blurring and generates inaccurate distance', this particular useful review will be classified into two groups i.e., *map* and *distance* as the taxonomy generates map and distance as the groups and the classification of useful reviews is accomplished though the means of keyword lookup classifying mechanism.

As mentioned earlier in Chapter 2 (refer to section 2.1) it becomes necessary to eliminate the duplicate instances of the useful reviews spread across multiple groups as they create confusion regarding the different priorities of the same useful reviews generated across different groups (Chen et al., 2014). Hence, we take our hybrid prioritisation method a step further to eliminate the duplicate useful reviews. Peng et al. (2012) suggest that the product developers address the requirement groups based on the descending order of their priorities. In addition, the NRP states that in every requirements addressing cycle, the product developers always tend to address the requirements with higher priorities (Bagnall et al., 2001). Based on these two studies, we develop the process to eliminate duplicate useful reviews which is as follows; the duplicate useful reviews spread across multiple groups are initially identified after which the priorities of their respective groups are compared. If a group has the highest priority over the other groups, then the duplicate instances of the particular useful review are eliminated from the groups having lower priorities. If the groups have equal priorities, then the useful review within a group with the highest priority is retained and the duplicate instances of the useful review are eliminated from the other groups. The listing of the elimination process is as follows; consider a useful review R being classified into groups G1, G2, and G3. After the prioritisation process is complete, let us assume that R has priority p1 in G1, p2 in G2, and p3 in G3. Then, if *priority of G1 > priority of G2 > priority of G3* then R is eliminated from G2, and G3, and if *priority of G1 = priority of G2 = priority of G3*, and if $p1 < p2 > p3$, then the R is eliminated from G1, and G3. For example, consider the useful review 'The signal drops and so no proper GPS that causes battery wastage' being classified into groups 'signal', 'GPS' and 'battery'. Within the signal group having a priority Low, the priority of useful review is Medium, whereas within the GPS group having a priority Medium, the priority of useful review is Low and within the battery group having a priority High, the priority of useful review is High. In such case, the entry to the useful review would be eliminated from the signal and GPS groups and would be retained in the battery group for app developers' to address. In addition, if all the groups have the same

priority, then the useful review would be retained in the battery group and eliminated from the others as the useful review holds the highest priority in the battery group in comparison to the others.

6.1.2 Experimental Settings (Group-based Prioritisation Method)

In this sub-section, we provide the details regarding the procedure that was followed to validate the primary outcome of the group-based prioritisation phase. First, we provide a brief description of the dataset that was used for the pilot experimentation purpose. We then provide the details of the pre-processing and POS tagging operations that were performed. Thereafter, we provide details regarding the evaluation procedure followed to validate the performance of the group-based prioritisation method.

6.1.2.1 Dataset

To demonstrate and evaluate the proposed group-based prioritisation method we utilise the My Tracks dataset that was part of the taxonomy generation pilot study conducted in Phase 3 of this undertaken research.

6.1.2.2 Useful Reviews Pre-processing and POS Tagging

We performed the basic useful reviews pre-processing operations mentioned in sub-section 4.3.1. That said, the first objective was to identify nouns, adjectives, and verbs from the pre-processed useful reviews to identify the keywords of interest. To achieve this goal, we repeated the POS tagging operation as mentioned in sub-section 5.4.2. Finally, after the necessary keywords of interest were identified, the useful reviews were prioritised using the group-based prioritisation method. It is to be noted that the pre-processing operation is performed to obtain the necessary keywords of interest and compute the priorities based on E_R, F_R and TF-IDF$_R$ variables. We apply VADER on the original form of useful reviews to compute -(SC_R).

6.1.2.3 Group-based Prioritisation Method Evaluation

After classifying the useful reviews into groups of interest using the automatically generated taxonomy, we initiated the group-based prioritisation method to generate the priorities of the useful reviews and their associated groups. We then benchmarked the performance of the proposed group-based prioritisation method using the commonly utilised time and accuracy dimensions (Bebensee et al., 2010; McZara et al., 2015). For the purpose of this thesis, we cover accuracy, time and operational demonstration dimensions. The other dimensions identified via the systematic mapping study such as requirements dependency, requirements updates and computational complexity are beyond the scope of this thesis and could be potentially part of the future work. It is to be noted that both proposed prioritisation methods i.e., group-based and individual are influenced by a requirements prioritisation method from a study from the systematic mapping phase (Peng et al., 2012) that addresses highest

number of requirements compared to the others, thus assuring the scalability dimension of prioritisation for both of our proposed methods.

6.1.2.3.1 Time

As useful reviews tend to be numerous, we compute the total time (seconds) required to prioritise a given set of useful reviews as time is of the essence when app developers have to address important useful reviews in the limited intervals of app maintenance and evolution cycles (Bebensee et al., 2010; Fabio et al., 2015; Pagano & Maalej, 2013). Benchmarking time required for prioritisation is also crucial when the app developers are driven by NRP as it helps them to determine the suitability of the utilisation of a particular prioritisation method based on the total time required for prioritisation (Bagnall et al., 2001). Moreover, failing to do so, negatively affects the business value of the app in the app market as quickly responsive app updates addressing end-users' requests, bugs or enhancements are crucial to keep the end-users engaged with the app or attract more end-users. For instance, app login problems should be fixed immediately as the end-users are unable to use the app. Thus, addressing critical bugs, requests or enhancements (accurately prioritised) on a timely basis allows the app to sustain in the competitive market. Hence, this thesis emphasises on the time dimension which is crucial for app developers towards fixing of prominent app concerns. Thus, it becomes necessary to benchmark the time required for prioritisation as it enables the app developers to determine the suitability of the utilisation of a particular prioritisation method based on the method's total time required for prioritisation.

6.1.2.3.2 Accuracy

We evaluate the accuracy of the group-based prioritisation method based on the priorities assigned by the stakeholders i.e., cross-validating the priorities of useful reviews generated by the solution against those assigned by the stakeholders (Bebensee et al., 2010; McZara et al., 2015). The empirical studies on requirements prioritisation evaluating accuracy dimension show that stakeholders preferences (i.e., requirements priority preferences of humans) are the reliable source to validate the priorities of requirements generated by a method, as these preferences reflect the actual order in which the requirements need to be fulfilled from the stakeholders perspective (Achimugu et al., 2016; Bebensee et al., 2010; Laurent et al., 2007).

Concerning useful reviews, the suitable candidates for stakeholders would be regular end-users of the app who are familiar with the day-to-day use of the apps and accordingly log the requests, bugs or enhancements related to the apps in the form of useful reviews (Maalej et al., 2016a; Pagano & Maalej, 2013; Panichella et al., 2015). Hence, such end-users have a better knowledge of the logged contents. Moreover, addressing of prioritised useful reviews assists app developers in launching essential app updates with end-users solicited requests, enhancements or rectified app bugs (Maalej et al., 2016a;

Robillard et al., 2014). Therefore, it is crucial that the priorities of the useful reviews match with those of the app's end-users.

That said, we performed an internal evaluation of the accuracy of the prioritisation method where we assumed the role of the stakeholders (end-users) as we are familiar with apps' experience (i.e., being regular app users and software developers ourselves) and are aware of the importance of a useful review from an end-user's perspective (Licorish et al., 2017). To achieve this evaluation, we first had to convert the priorities generated in numerical range (0 to1) to three intervals (Low, Medium, High). This in turn leads towards the ease of simplifying the generation or assignment of the priorities, and allows us to measure the reliability of the prioritisation results in alignment with the widely followed software engineering convention (Boehm & Port, 2001; Diebold et al., 2018). Thus, we map the 10 numerical priorities (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1) onto interval priorities (Low, Medium, High) using the class interval approach (Evans, 1977). Based on the computed class intervals, we map numerical priorities in range 0-0.3 onto Low priority interval, numerical priorities in range 0.4-0.6 as Medium priority interval and numerical priorities in range 0.7-1.0 are mapped onto High priority interval.

After achieving the necessary conversion, our primary objective is to assist the stakeholders to assign priorities to the useful reviews based on the defined priority intervals, and for this we have developed a guideline inherited from the priority assignment codes mentioned in (Licorish et al., 2017). Following the guidelines proposed by Licorish et al. (2017), we were able to map the authors' code 2 assignment guideline onto Low priority interval, code 3 assignment guideline onto Medium priority interval and code 4 assignment priority interval onto High priority interval. The authors' code 1 assignment guideline was discarded as it indicated non-useful reviews and we have already figured out an approach to automatically filter useful reviews in phase 2. Our guideline regarding the priority generation and assignment towards useful reviews is provided in Table 6.1 where Low priority interval corresponding to code 2 assignment guideline indicates the contents within the useful reviews that are not essential towards the functionality or performance of the app. The code 3 assignment guideline mapped onto Medium priority interval indicates useful reviews that directly affect the functionality or performance of the app. Finally, High priority interval inferred from code 4 assignment guideline indicates severe app concerns within useful reviews that require immediate attention of the app developers. In Table 6.1 along with the priority assignment guideline, we provide some examples of the useful reviews that fall under the Low, Medium, and High priority intervals. The priorities of the useful reviews generated by the group-based prioritisation method will be compared against those assigned by the stakeholders to determine the accuracy of the group-based prioritisation method.

Table 6.1 Priority assignment guideline

Numerical range and priority	Justification for the priority assignment	Useful reviews examples
0 - 0.3 Low	Useful reviews that reflect requests, bugs or enhancements pertaining to an app that seem optional (not obligatory) towards the app's functionalities or performance.	1. "Love the material design. Dark mode and Chromebook optimisation would be awesome." 2. "I like it when you can get free add-ons sometimes on your Vodafone app but make it regular please."
0.4 - 0.6 Medium	Useful reviews that reflect requests, bugs or enhancements pertaining to an app that seem mandatory (imperative) towards the app's functionalities or performance.	1. "The first few times I turned it on the graphics were great but now the butterflies are just coloured squares along with the writing is messed up." 2. "It's okay, good for basic use but some options are not available on the app so sometimes I need to use the full website on a computer."
0.7 - 1.0 High	Useful reviews that reflect requests, bugs or enhancements pertaining to an app that seem severe (critical) towards the app's functionalities or performance.	1. "The only streaming app on my Samsung note that won't work. Crashes frequently. Always gives 'unexpectedly stopped working' notice." 2. "I can't seem to download the app due to "Error: 941" and it says "My Vodafone can't be downloaded". Please fix this!"

It is to be noted that in the previous study from which the priority assignment guideline mentioned in Table 6.1 was derived, the authors had filtered reviews from the My Tracks dataset using the ratings criteria (i.e., retaining reviews whose ratings were less than or equal to 3) and later manually labelled them according to their developed coding scheme, and thus, only 855 useful reviews labelled as High, Medium and Low based on the new guideline mentioned in Table 6.1 were retained for evaluation (Licorish et al., 2017). We use these useful reviews to evaluate the accuracy and time of the group-based prioritisation method.

We provide results of this pilot study in the Results section (refer to section 6.2) and present the details related to the individual prioritisation method in the next sub-section.

6.1.3 Individual Prioritisation Method

While the group-based prioritisation method classifies useful reviews into groups of interest, we noticed that within a group, different useful reviews might have different priorities. While some studies (Chen et al., 2014; Peng et al., 2012) have considered group-based priorities, others (Asghar et al., 2013; Voola & Babu, 2013; Chopra et al., 2016) have considered individual requirements for prioritisation. The individual requirement-based prioritisation is also a more fine-grained method. Therefore, our work

also considered the individual-based prioritisation method. In this approach we decided to discard the classification approach before prioritisation (i.e., directly prioritising useful reviews after filtering) (Asghar et al., 2013; Voola & Babu, 2013). This was mainly due to the group-based prioritisation method removing majority of group information associated with the useful reviews because of the elimination of the duplicate useful reviews; as we observed that only few groups and their related priorities were retained and thus, the method missed out on the other important groups of interest. In addition, based on pertinent studies we had an intuition that prioritising useful reviews directly after the filtering process would generate better results as it would avoid the complexities involved in classification approach that hampered the performance of the prioritisation method because of factors such as handling of redundant information, computational time, and so on (Asghar et al., 2017; Sadiq et al., 2009; Zhang et al., 2014).

Thus, in this phase we conducted a pilot study in which we directly applied the multi-criteria heuristic function (refer to equation (19)) on the previously mentioned set of 855 useful reviews of the My Tracks dataset to prioritise them individually without classification. By doing so, we found out that such individual prioritisation method generated better results than the group-based prioritisation method (refer to sub-section 6.2.2). Hence, we conducted a full-scale experimentation of this method in this phase and we highlight the details regarding this experimentation below. The RQ related to benchmarking the performance of the prioritisation method during the pilot and full-scale study is similar to that of RQ4.1 and is

RQ4.2 What is the performance of the developed individual prioritisation method?

6.1.4 Experimental Settings (Individual Prioritisation Method)

In this sub-section, we provide the details regarding the datasets, the pre-processing operations and the evaluation approach used to empirically validate the individual prioritisation method.

6.1.4.1 Datasets

To demonstrate general relevance of the individual prioritisation method we extracted the latest reviews (i.e., reviews logged up to November 2019) of four apps hosted on the public apps distribution platform Google Play Store using a web crawler (refer to Appendices, section B). These four apps belonged to Casual (App 1), Entertainment (App 2), Shopping (App 3) and Tool (App 4) categories, and comprised of 5044, 3683, 4559 and 6583 reviews respectively. The average length of these reviews from these four apps ranged from 112 to 137 words and the average ratings of these apps ranged from 1.5 to 4.2. For anonymity purpose, we do not reveal the names of the apps. We provide the summary of these extracted datasets in Table 6.2, where the first column corresponds to the identifier of the particular app, followed by the number of reviews logged for the app, the maximum length of a review pertaining

to the app, the minimum length of the review pertaining to the app, the average length of the total reviews of the app, the app's average end-user rating and the app's category.

Table 6.2 Extracted datasets summary

App ID	Total number of reviews logged	Maximum review length (characters)	Minimum review length (characters)	Average length of review	Average app rating	Category
App 1	5044	2110	2	126	4.2	Casual
App 2	3683	1483	2	137	1.5	Entertainment
App 3	4559	1732	3	112	3.2	Shopping
App 4	6583	1434	2	123	2.4	Tool

These extracted reviews were then independently labelled as useful or non-useful by the two supervisors and the PhD candidate using the filtering rules mentioned in (Chen et al., 2014). Next, we utilised Fleiss Kappa to perform the reliability assessments to support our evaluations. The Fleiss co-efficient was found to be 0.78 (substantial agreement), 0.65 (substantial agreement), 0.68 (substantial agreement) and 0.71 (substantial agreement) for App 1, App 2, App 3 and App 4 respectively (Landis & Koch, 1977). Follow up discussions were conducted among us to resolve any conflicts and establish consensus for achieving a reliable manual labelling process, where we converged on 100% agreement. After performing the necessary tasks (i.e., reliability assessments and manual filtering) App 1, App 2, App 3 and App 4 indicated 1138, 1760, 1154 and 1120 useful reviews respectively. That said, we performed the basic useful reviews pre-processing operations and POS tagging to identify keywords of interest mentioned in sub-section 6.1.2.2. Furthermore, we make the datasets[17] (i.e., both raw and labelled) used in this study publicly available for the research community.

6.1.4.2 Individual Prioritisation Method Evaluation

We followed the evaluation approach mentioned in sub-section 6.1.2.3 to benchmark the performance of the individual prioritisation method using the time and accuracy dimensions. In this phase, we evaluated the accuracy of the method at two levels. Initially, we performed internal evaluation of the method i.e., comparing the priorities of the useful reviews generated by the method against those assigned by us. Next, to perform the external evaluation of the individual prioritisation method we recruited 10 participants from the department of Information Science at the University of Otago. To conduct the external evaluation, we initially had to get an ethics application approved from the Human Ethics Committee of the University of Otago (refer to Appendices section C for its complete details). Furthermore, the participants of the external evaluation are regular apps' users and have experience

[17] https://tinyurl.com/yy6nsurh

with apps along with software development. For both evaluations i.e., internal and external, the stakeholders were made familiar with descriptions and usage of the four apps.

That said, based on the number of useful reviews belonging to each app, there was a cognitive overhead associated with the limited human resources available for internal and external evaluation. Therefore, for internal evaluation, we used random sampling method (i.e., 95% confidence interval, 5% error margin) to determine the appropriate sample of useful reviews from each app that had to be evaluated (Morse, 2000). The random sampling method returned 288, 316, 289 and 287 useful reviews from App 1, App 2, App 3 and App 4 respectively. Using the guidelines mentioned in Table 6.1, the two supervisors and the PhD candidate independently prioritised the randomly sampled useful reviews. Next, we performed the required reliability assessments and the Fleiss coefficients were found to be 0.54 (moderate agreement), 0.45 (moderate agreement), 0.61 (substantial agreement) and 0.66 (substantial agreement) for App 1, App 2, App 3 and App 4 respectively (Landis & Koch, 1977). Later, follow up discussions among the team were held to resolve any conflicts on the priorities of the app reviews and this lead to 100% convergence (i.e., establishment of consensus) essential towards the evaluation of accuracy.

Finally, to perform the external evaluation, the 10 recruited participants were subjected to a 30 minute study designed based on the participant cognitive load limitation guideline (De Jong, 2010; Katsanos, et al., 2009). Katsanos et al. (2009) have shown that a sample size of 10 participants is reliable enough to evaluate the outcomes of software engineering research or application. Next, to address the requirement towards external evaluation, we performed stratified random sampling of the total useful reviews that were part of internal evaluation to get the necessary useful reviews for the participants to evaluate (Kadilar & Cingi, 2003). Stratified random sampling prevents the sampling process from being dominated by the useful reviews of a particular app(s) by returning the approximate equal number of useful reviews from each app. Out of the total 1,180 useful reviews (i.e., 288 - App 1, 316 - App 2, 289 - App3 and 287 - App 4) which were the part of internal evaluation, the stratified random sampling returned 71, 73, 74, and 72 (total 290: 95 % confidence interval, 5% error margin) from App 1, App 2, App 3 and App 4 respectively (Kadilar & Cingi, 2003). During external evaluation, each participant evaluated a non-identical set of 29 useful reviews, wherein each set comprised of approximately equal number of app reviews from four apps. Initially, using the guideline from the cognitive load theory, we estimated an approximate set of 30 useful reviews would be adequate for evaluation for each participant who had requested a maximum participation time of 30 minutes (De Jong, 2010). Based on the guideline, each of the internal participants (i.e., two supervisors and the PhD candidate) initially had independently recorded the average time required to evaluate useful reviews (i.e., time required to analyse each review and assign it a priority). It was found out that, on average it takes around 1 minute to perform the evaluation of a single useful review. After establishing an informal agreement on the

average time required to evaluate a useful review, it seemed appropriate that each participant evaluate the non-identical set of 29 useful reviews (i.e., 290 useful reviews among 10 participants) in a span of 30 minutes. The additional 1 minute would assist in getting the participant's mind frame ready to perform manual evaluation after the necessary briefing on evaluation was conducted. Furthermore, to establish a common understanding, the external participants were briefed in detail on the objective of the evaluation and priority assignment guideline mentioned in Table 6.1 prior to the conduct of the individual external evaluations. The details pertaining to the objective of external evaluation, the necessary briefings and an external participant evaluation sheet are provided in the Appendices (refer to section D).

We provide results of the pilot and full-scale experimentation study performed in this phase in the Results section.

6.2 Results

In this section, we report the results of the pilot study pertaining to the group-based prioritisation method and the individual prioritisation method. Later, we report the results of the conducted full-scale study on the individual prioritisation method.

6.2.1 Group-based Prioritisation Results

Initially, we evaluated the performance of the group-based prioritisation method on My Tracks dataset. Table 6.3 indicates the overall performance of the group-based prioritisation method based on time and accuracy dimensions.

Table 6.3. Performance of group-based prioritisation method on My Tracks dataset

Number of useful reviews	Time (seconds)	Accuracy (%)
855	347.6	58.0%

The group-based prioritisation method required 347.6 seconds to prioritise 855 useful reviews of the My Tracks dataset and exhibited an accuracy of 58.0%. Furthermore, out of 152 app features, only 84 app features were retained after prioritisation. This was because of the duplicate reviews elimination process. A useful review could get classified into several groups because of the keyword lookup classifying mechanism and thus, its duplicate instances might exist in several groups. After the elimination process is initiated, the duplicate instances of useful reviews in the groups having low priorities are eliminated and is retained in the group having the highest priority. However, certain low priority groups do not retain any useful reviews and thus are discarded. Also, out of these 84 app features only 4 app features (i.e., accuracy, distance, download and package) were found common with those

reported in the previous study for priorities cross-validation purpose (Licorish et al., 2017). The priorities of these app features did not match with those presented in the previous study, further reducing the suitability of the group-based prioritisation method for useful reviews prioritisation.

We discuss the results of the undertaken pilot study on the prioritisation of useful reviews using the group-based prioritisation method and the considerations of their implications in the Discussion section (refer to Section 6.3). In the next sub-section, we present the results related to the individual prioritisation method.

6.2.2 Individual Prioritisation Method

In this sub-section, we report the results of the pilot and full-scale study conducted using the individual prioritisation method. Firstly, we report the results of the pilot study in Table 6.4.

Table 6.4. Performance of individual prioritisation method on My Tracks dataset

Number of useful reviews	Time (seconds)	Accuracy (%)
855	24.4	65.0%

From Table 6.4 it is observed that the individual prioritisation method required 24.4 seconds to prioritise 855 useful reviews of the My Tracks dataset and exhibited an accuracy of 65.0%. When the performance of the individual prioritisation method was compared with that of the group-based prioritisation method (refer to Table 4.9), a reduction of 92.98% was observed in case of the time required for prioritisation and an increase of 7% was observed in case of accuracy. This confirmed our intuition that stated the performance of the individual prioritisation method would be better than group-based prioritisation method (refer to sub-section 6.1.3).

Secondly, we report the results of the full-scale study that dealt with the prioritisation of the useful reviews belonging to four apps: App 1, App 2, App 3 and App 4). Table 6.5 indicates the total time required by the individual prioritisation method to prioritise the useful reviews. The useful reviews of App 4 required the least time (17.1 seconds) for prioritisation whereas the useful reviews of App 2 required the most time (24.6 seconds). The useful reviews of App 3 and App 1 required 17.80 seconds and 19.33 seconds for prioritisation respectively.

Table 6.5 Total time required for prioritisation

App ID.	Number of useful reviews	Time (seconds)
App 1	1138	19.3
App 2	1760	24.6
App 3	1154	21.8
App 4	1120	17.1

Thirdly, we present the accuracy of the individual prioritisation method after completing the full-scale internal evaluation in Table 6.6. Based on the priorities of useful reviews manually assigned by us against those generated by the individual prioritisation method, the method exhibited highest accuracy in prioritising the useful reviews of App 3 (81.3%) followed by App 1 (77.43%), App 4 (76.7%) and App 2 (73.3%).

Table 6.6 Accuracy of individual prioritisation method (internal evaluation)

App ID.	Number of useful reviews	Accuracy (%)
App 1	288	77.4
App 2	316	73.3
App 3	289	81.3
App 4	287	76.7

Finally, we report the results of the external evaluation. Table 6.7 indicates the accuracy results obtained from the external evaluation. Based on the priorities of useful reviews manually assigned by the participants against those generated by the individual prioritisation method, the method exhibited highest accuracy in prioritising the useful reviews of App 1 (85.9%) followed by App 4 (81.9%), App 3 (74.3%) and App 2 (74.0%).

Table 6.7. Accuracy of individual prioritisation method (external evaluation)

App ID.	Number of useful reviews	Accuracy (%)
App 1	71	85.9
App 2	73	74.0
App 3	74	74.3
App 4	72	81.9

We had performed internal and external evaluation of the individual prioritisation method to determine its accuracy (as shown in Tables 6.6 and 6.7). To achieve this, we had involved humans based evaluation approach as such approach provides the necessary reliable ground truth for cross-validation purposes (Stumpf et al., 2007). That said, the Pearson correlation between the priority assignment judgments of us and the participants was found to be 0.8 (p-value < 0.01) which indicates that there was a substantial level of agreement between the internal and external participants on the subjectivity involved in assigning priorities to the useful reviews. The internal evaluation reported an average accuracy of 77.17%, whereas an average accuracy of 79.04% was reported in external evaluation for all the four apps. Even though the sample selected for external evaluation was representative of the total population (i.e., useful reviews) that was a part of the internal evaluation, the average accuracy results are approximately similar with marginal difference (~1.9%) among them indicating promising results and show significant level of alignment among the priority assignment perception of humans.

We discuss the results of the undertaken studies on the prioritisation of useful reviews using the individual prioritisation method and the considerations of their implications in the Discussion section (refer to section 6.3).

That said, the operational demonstration of the classification, group-based prioritisation and individual prioritisation methods comprising of the filtering of useful reviews using the best performing variant IV can be assessed by accessing the web tool[18]. A set of sample reviews have also been provided for demonstration purpose[19]. The walkthrough towards operational demonstration with the support of essential relevant screenshots are provided in the Appendices (refer to section F) of this thesis.

In the next section, we present the discussion and implications along with the threats to validity related to the relevant phases.

6.3 Discussion

The research study that was conducted in this phase shows that it is possible to develop an automated prioritisation method that can prioritise numerous useful reviews. The two proposed prioritisation methods (i.e., group-based and individual) generate the required priorities by directly operating on the end-users' requests, bugs or enhancements contained in the useful reviews, and is thus independent of domain knowledge and priority preferences of stakeholders. With regards to this, it is possible for the developed prioritisation methods to accommodate new useful reviews and generate updated priorities of useful reviews during the prioritisation process. Such methods hold promise in supporting software maintenance and evolution cycles of apps, where there is often a necessity to convert numerous useful reviews into actionable knowledge (i.e., classification or prioritisation) in regular short intervals (Fabio et al., 2015; Groen et al., 2015). That said, based on the findings presented in this study, our intuition is that the foremost aspect of developing an automated prioritisation method is the identification and assembly of relevant prominent prioritisation methods. The fulfilment of this aspect is dependent on the type of prioritisation research problem. For instance, if the prioritisation problem demanded to prioritise useful reviews based only on the frequency of occurrences of the keywords of interest present in those useful reviews, then frequency method would be appropriate in such case. For solving the problem related to the prioritisation of numerous useful reviews, we conducted an extensive search for the appropriate prioritisation methods and assessed their suitability when the four methods (i.e., E_R, F_R, TF-IDF$_R$ and - (SC_R)) were assembled by means of multi-criteria heuristic function. The multi-criteria heuristic function provides flexibility towards prioritisation of useful reviews based on specific objectives (del Campo et al., 2016). For instance, business manager of an app could set the value of δ (refer to equation (19)) to a larger value if the requirement for prioritisation is based on end-users

[18] https://recptool.otagointeractive.nz
[19] https://f2h.io/buxdrhm7dnsf

satisfaction levels of app usage. Subsequently, app developers could perform prioritisation based on the level of information conveyed by useful reviews by increasing the value of α. Moreover, it would be easy to incorporate any additional methods, modify the existing ones or remove the unnecessary ones via the multi-criteria heuristic function depending upon the requirement of prioritisation research or application. For instance, app developers can develop a method that prioritises useful reviews based on the geographical location of the app's end-users and incorporate this method as a variable of the multi-criteria heuristic function. The sub-sections below discuss the results and implications of RQ4.1 and 4.2

6.3.1 RQ4.1 What is the performance of the developed group-based prioritisation method?

On default seed values of α, β, γ, and δ (i.e., 0.25), the group-based prioritisation method exhibited accuracy of 58% and required 347.6 seconds to prioritise 855 useful reviews and their associated groups in the conducted pilot study. Concerning time, the method prioritised 147 useful reviews per minute given that the time dimension also considered the classification of useful reviews based on the automatically generated taxonomy and elimination of duplicate useful reviews for the group-based prioritisation method. The total time required for the classification phase (i.e., taxonomy generation and classifying useful reviews into the dynamically generated groups of interest) within the group-based prioritisation method was 323.10 seconds. The POS tagging operation required 20 seconds and the taxonomy generation along with the classification of useful reviews into groups of interest took 303.10 seconds. That said, the actual time required to prioritise the useful reviews and their associated groups along with the elimination of duplicate useful reviews was 24.5 seconds. Thus, the proposed group-based prioritisation method could benefit from a timely optimised POS technique, providing a scope for future research. However, given the number of useful reviews prioritised per minute, in terms of the time dimension, the proposed method performs better than most of the requirements prioritisation methods presented in Table 3.9 (refer to Chapter 3, sub-section 3.5.6) with only ReproTizer outperforming our proposed group-based prioritisation method.

Concerning accuracy, the group-based prioritisation method performed fairly when compared to the requirements prioritisation methods mentioned in Table 3.6 (refer to Chapter 3, sub-section 3.5.6), but intuitively the result based on accuracy was not noteworthy. Moreover, the group-based prioritisation method was unable to retain the majority of groups (app features) after the duplicate review elimination process. With regards to this, only 4 app features matched with those present in the prioritised dataset that was used as ground truth to validate the outcome of the group-based prioritisation method (Licorish et al., 2017). Since the priorities of these app features did not match, an informal accuracy of 0% was noted for the prioritised group, making this group-based prioritisation method ineffective. Such findings open avenues for potential research whose primary objective would be to improve the accuracy of the

group-based prioritisation method by developing approaches specialised in efficient elimination of duplicate useful reviews and retaining the majority of prioritised groups.

It is to be noted that in this and further sub-sections, we have compared the performances of the group-based prioritisation method and the individual prioritisation method with those of other requirements prioritisation methods based on the accuracy and time dimensions (refer to Chapter 3, sub-section 3.5.6). Even though the empirical studies on requirements prioritisation methods covering accuracy and time dimensions have non-identical experimental settings (i.e., research methodology, number of requirements, type of requirements - dependent or independent, validation procedures and so on) the comparisons made are fitting for general summarisations. In addition, as mentioned earlier, the empirical studies from the app domain did not benchmark the performance of their proposed prioritisation methods based on any dimension, hence, we are unable to make any general comparison with those studies. That said, the studies from the app domain focusing on prioritisation of app reviews or the empirical studies on requirements prioritisation lacking the dimensions could benefit from the utilisation of suitable dimensions that have been identified through means of our conducted systematic mapping study on requirements prioritisation.

In addition, it is to be noted that both of our proposed prioritisation method (i.e., group-based and individual) are not dependent on domain knowledge and the priority preferences of the stakeholders to generate priorities of useful reviews. This contrasts with requirements prioritisation methods which utilise domain knowledge and priority preferences of the stakeholders to gain better prioritisation results in terms of accuracy. For instance, AHP or BPL are known to generate accurate priorities of the requirements when the priority preferences provided by the various stakeholders are closely related to each other and are in close proximity with the criteria used for validating accuracy of the particular requirements prioritisation method (Bebensee et al., 2010; Chopra et al., 2016). One common example of such criteria is a validation dataset consisting of already prioritised requirements. However, on default seed values of the parameters α, β, γ, and δ, both the proposed prioritisation methods have shown promising results and there is a potential scope to improve the methods performance based on parameter tuning in future. We discuss some of the parameter tuning concepts in sub-section 6.3.3.

To conclude, albeit a pilot study, the results based on accuracy and time dimensions of group-based prioritisation method that was evaluated in phase 4 did not seem satisfactory (accuracy: 58.0% and time: 347.6 seconds), we decided to discard the classification approach before prioritisation (i.e., directly prioritising useful reviews after filtering) (Asghar et al., 2013; Voola & Babu, 2013). This was mainly due to the group-based prioritisation method removing the majority of group information (i.e., only few groups and their related priorities were retained, and thus, missing out on the other important groups of interest) associated with the useful reviews. This was the result of the elimination process that removes duplicate (useful) reviews. In addition, based on pertinent studies we previously had an intuition that

prioritising useful reviews directly after the filtering process would generate better results as it would avoid the complexities involved (e.g. redundant information, computational time, and so on) in the classification approach that hampered the performance of the prioritisation method (Asghar et al., 2017; Sadiq et al., 2009; Zhang et al., 2014). The next sub-section addresses this issue.

6.3.2 What is the performance of the developed individual prioritisation method?

In the pilot study, the individual prioritisation method outperformed the group-based prioritisation method by exhibiting an accuracy of 65.0% and requiring only 24.4 seconds to prioritise the same set of 855 useful reviews. Concerning time, the individual prioritisation method prioritised 2085 useful reviews per minute. This is a significant improvement over the group-based prioritisation method that could only prioritise 147 useful reviews, albeit the group-based prioritisation method constituted the classification phase. It was observed that the POS tagging operation performed to identify the keywords of interest, which dominated the prioritisation time by 82% (20 seconds), whereas the actual time to prioritise the useful reviews was minimal at 4.4 seconds. In addition, given the number of useful reviews prioritised per minute, in terms of the time dimension, the proposed method performs better than most of the requirements prioritisation methods presented in Table 3.9 (refer to Chapter 3, sub-section 3.5.6), with only ReproTizer outperforming our proposed individual prioritisation method.

Concerning accuracy, the individual prioritisation method performed satisfactorily when compared to the requirements prioritisation methods mentioned in Table 3.6 (refer to Chapter 3, sub-section 3.5.6). This confirmed our intuition to directly prioritise useful reviews after they were filtered from a pool of reviews as it prevented the complexities of the classification phase from hampering the performance of the multi-criteria heuristic function that is the core driving force of our proposed prioritisation methods. As the results of the individual prioritisation method based on accuracy and time dimensions were more promising than the group-based prioritisation method, in the conducted pilot study, we performed a full-scale evaluation of the individual prioritisation method. The evaluation was performed on datasets pertaining to four new apps belonging to different categories and external participants were included to assure a rigorous validation of the proposed individual prioritisation method.

In the pilot and full-scale evaluation study, it was observed that the individual prioritisation method took less than half a minute to prioritise 855 useful reviews, while the traditional manually driven requirements prioritisation methods like AHP, BPL, NA and Weiger took much longer time (i.e., measures in minutes or hours) to prioritise a small set of requirements (refer to Table 3.9). Concerning the time results reported in the full-scale evaluation study, we observed that the POS technique that identified the keywords of interest dominated the prioritisation time by 80-85% for App 1 (16.43 seconds), App 2 (20.40 seconds), App 3 (14.24 seconds) and App 4 (14.24 seconds) respectively. This shows that the individual prioritisation method requires less time to perform prioritisation of numerous

useful reviews belonging to the four apps (App 1 - 2.9 seconds, App 2 - 4.18 seconds, App 3 - 3.56 seconds and App 4 - 3.25 seconds). Similar to the group-based prioritisation method, these findings reveal that our proposed individual prioritisation method could benefit from a timely optimised POS technique.

Concerning accuracy, during the full-scale evaluation study we performed external evaluation along with the internal evaluation of the individual prioritisation method's prioritisation outcomes to determine its overall accuracy. The outcome sample selected for external evaluation was representative of the total number of reviews that were a part of the internal evaluation. Even though a representative sample, on average basis, the accuracy results (77.17 % and 79.04%) are approximately close, with marginal difference (~1.9%) indicating promising results. In addition, the significant Pearson correlation coefficient indicated that there was substantial level of agreement between the internal and external participants on the subjectivity involved in assigning priorities to the useful reviews. That said, while evaluating the accuracy of the group-based prioritisation method and individual prioritisation method we utilised human evaluators (internal and external evaluators) who had experience with apps and software development, and these human evaluators provided the necessary ground truth for cross-validation purposes (Stumpf et al., 2007). Moreover, the utilisation of such evaluation practise was in alignment with the guidelines provided by requirements prioritisation studies that suggested stakeholders priority preferences are reliable source to validate the priorities of requirements generated by a particular requirements prioritisation method (Achimugu et al., 2016; Asghar et al., 2013; Bebensee et al., 2010; McZara et al., 2015).

Furthermore, both of our proposed prioritisation methods are novel and they are in their elementary stage. We have not experimented with the fine tuning of α, β, γ, and δ parameters for useful reviews belonging to different apps. It is unclear what results of the proposed methods would be generated on different parameter settings, which in turn would assist in deciding the best, average and worst case scenario in terms of accuracy and time required to prioritise numerous useful reviews belonging to a particular app. However, researching this aspect is beyond of the scope of the current study and could be planned as potential future work. Nevertheless, some of the accuracy and time results related to the prioritisation of numerous useful reviews of the individual prioritisation method reported in this study are substantial (Accuracy: 65.0%, 85.9%, 81.9%, 81.3%, 74.0% and Time: 24.4 seconds, 19.33 seconds, 24.58 seconds, 17.80 seconds, and 17.10 seconds). Thus, the proposed prioritisation methods, and specifically the individual prioritisation method, holds promise for prioritising useful reviews to support app maintenance and evolution cycles. In the next sub-section, we discuss some of the concepts that would assist with the automation process leading to fine tuning of parameters of the multi-criteria heuristic function.

6.3.3 Automated Parameter Fine Tuning

In this sub-section, we propose and discuss abstracts of few preliminary concepts that would assist in performing automated fine tuning of the parameters (i.e., α, β, γ, and δ) pertaining to the multi-criteria heuristic function to potentially generate optimal prioritisation results. This in turn will potentially allow the proposed methods to exhibit better performance in terms of accuracy and time dimensions. In addition, this would transform the multi-criteria heuristic function into an evolutionary multi-criteria heuristic function that automatically fine tunes the parameters to prioritise numerous useful reviews to generate optimal prioritisation results (Wessing et al., 2017).

6.3.3.1 Surrogate Modelling Approach

One approach to perform automated fine tuning of the parameters is to subject the multi-criteria heuristic function to a surrogate model (Forrester et al., 2007). The multi-criteria heuristic function works in a feed forward fashion i.e., takes the useful reviews and keywords of interest as input, processes the input with the particular prioritisation methods which are incorporated as variables of the multi-criteria heuristic function and generates the list of prioritised useful reviews. However, the prioritisation results produced by the proposed multi-criteria heuristic function are linear and such results are sometimes known to be insignificant over time in comparison to the non-linear results generated by multi-criteria heuristic functions that are driven by feedback mechanisms (Solow, 2007). This is because, linear results contribute towards tactical growth, whereas non-linear results contribute towards the strategic growth of the respective multi-criteria heuristic functions. Therefore, we propose a feedback mechanism based multi-criteria heuristic function derived from surrogate modelling (Forrester et al., 2008). Surrogate modelling iteratively creates optimal results over time through means of evolutionary computing (Forrester et al., 2007). Figure 24 illustrates the proposed surrogate model to prioritise numerous useful reviews, wherein, the causal useful reviews prioritisation results can be used to create virtual models of the numerous useful reviews prioritisation problem, and with the assistance of these virtual models, optimal results can be generated that could be applied to the real world numerous useful reviews prioritisation problem reflecting one of the virtual model. The causal prioritised useful reviews indicate the outcome of a specific prioritisation operation. Such causal prioritised useful reviews act as solutions to the relevant prioritisation problems which can be virtually formulated. These virtually formulated problems represent the different versions of the prioritisation problem that are formulated to address the prioritisation of useful reviews. For instance, a specific version of the problem can be efficiently solved using entropy and sentiment methods based on setting appropriate α and δ values, and another version of the problem may only require the β value to be set to 1 for efficient prioritisation. These problems would assist in determining the right combination of parameters to prioritise useful reviews pertaining to a specific prioritisation problem. Such combination

of parameters would act as surrogates of the default parameter values to generate optimal prioritisation results pertaining to real world useful reviews.

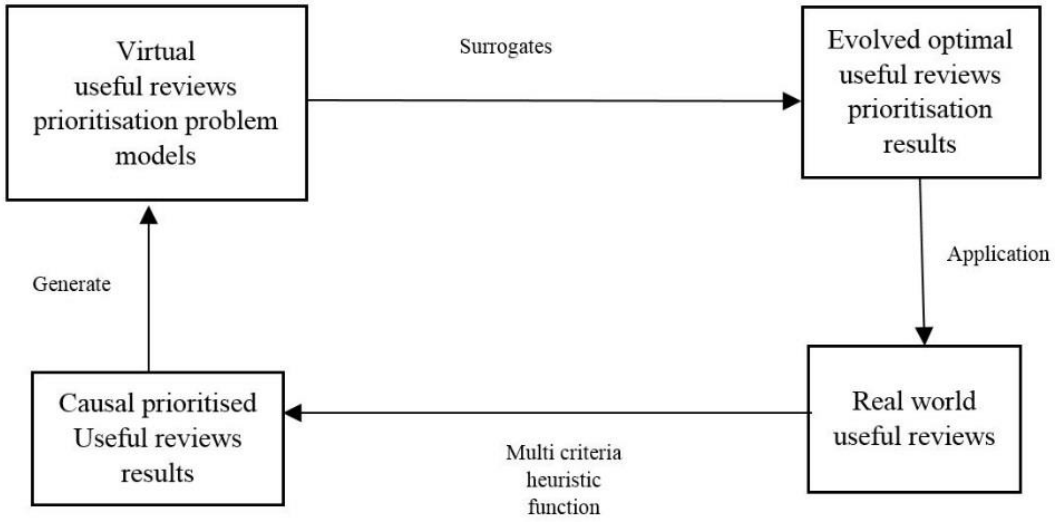


Figure 24. Surrogate model of the multi-criteria heuristic function towards numerous useful reviews prioritisation problem

However, implementing the proposed surrogate model of the multi-criteria heuristic function is not a straightforward task. There are few challenges that need to be addressed to achieve this model. We represent these challenges in Figure 25 using the 'What, Why, and How' research methodology (Fuchs & Fuchs, 2006). Therefore, in this case 'What' addresses the object of examination (i.e., numerous useful reviews) that causally predicts the 'Why', which reflects the priorities of the useful reviews, which in turn assists in answering the 'How' aspect i.e., how we can evolve and test optimal results through the means of virtual useful reviews prioritisation problem models.

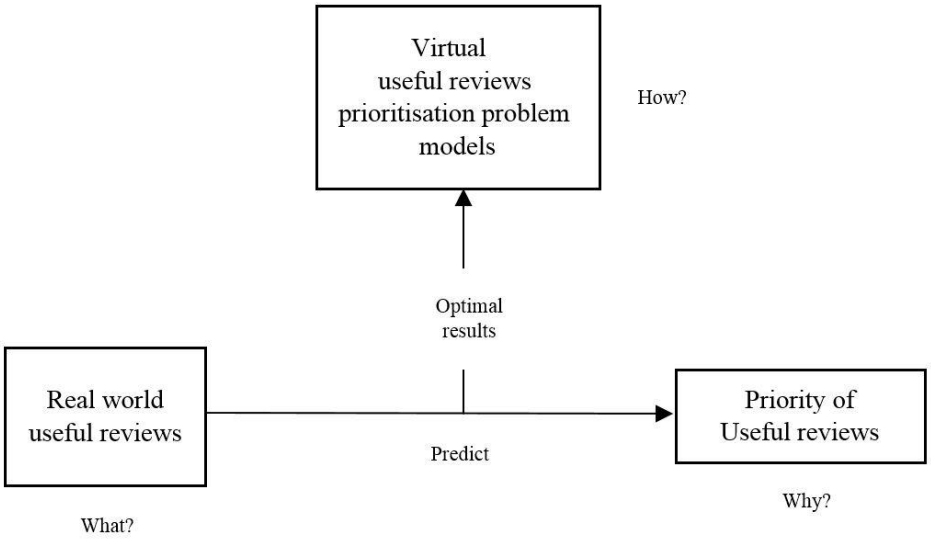


Figure 25. Challenges represented in the form of questions to implement the proposed surrogate model

6.3.3.2 Parameter Sweeping Approach

Another approach to find the optimal set of values for the parameters α, β, γ, and δ is by performing parameter sweeping so that the multi-criteria heuristic function can optimally solve the numerous useful reviews prioritisation problem (Wibisono et al., 2008). The prime objective of the parameter sweeping methods in case of prioritisation of useful reviews is to identify appropriate values for the parameters that would produce an optimal multi-criteria heuristic function which could potentially minimise a predefined cost function on given useful reviews (Bergstra & Bengio, 2012). The predefined cost function in this scenario would be a function that would map the acceptable accuracy and time results related to prioritisation onto a real number. The overall objective would be to minimise the cost function (Bergstra & Bengio, 2012). Later, by means of an objective function (generating accurate and timely prioritised useful reviews) multiple combinations of values pertaining to the parameters can be evaluated to check for several values returned by the cost function (Bergstra & Bengio, 2012). The cost function with minimum value would determine the optimal set of parameter values (Bergstra & Bengio, 2012). There are some prominent parameter sweeping methods specialised in such tasks. One common method is Bayesian method that generates a probabilistic model of the function mapping from the parameter values to the evaluated objective function (Snoek et al., 2012). The method generates a set of parameter values based on the current probabilistic model and simultaneously updates the model at every iteration with the objective of identifying the optimal parameter values. Such a method generates the values pertaining to the parameters and shortlists those that are close to the optimal ones (Snoek et al., 2012). Another method is to utilise evolutionary algorithm that would initially generate random sets of parameter values to later evaluate these values and obtain their fitness function (e.g., accuracy or time results of the multi-criteria heuristic function with those parameter values) (Bergstra et al., 2011). Later, the sets of parameter values would be ranked based on their relative fitness to substitute the sets of values of the parameters generating worst results with new sets of values of the parameters computed through crossover and mutation. The evolutionary algorithm runs iteratively until the algorithm is no longer generating any optimal parameters (Bergstra et al., 2011).

6.3.3.3 Orthogonal Procrustes Problem Approach

In addition, another potential approach to generate the optimal parameter values can be considered. The approach would require the final set of normalised priority values generated by each variable E_R, F_R, TF-IDF$_R$ and - (SC_R) to be represented in the form of a matrix (X) where each column represents the priority value generated by each variable respectively. The values in each row would represent the priority values generated by each variable for a particular useful review. In another matrix (Y) having a single column, P_R indicating the actual priorities (i.e., ground truth obtained from domain experts) of the useful reviews would reside. The arrangement of these two matrices can be seen as orthogonal

Procrustes problem (matrix approximation problem), where the objective would be to compute an orthogonal matrix (Z) which would closely map X to Y (Gower & Dijksterhuis, 2004) given as

$$Z = \text{argmin}_{\Omega} ||\Omega X - Y||_F \quad (20)$$

Where (20) is subject to $\Omega^T\Omega = I$ and $||\Omega X - Y||_F$ is derived from Frobenius norm (Storjohann, 2001). This is equivalent to finding the nearest orthogonal matrix to given matrix (M) such that $M = YX^T$ and to find the orthogonal matrix Z, singular value decomposition is utilised such that

$M = U\sum V^T$ to derive

$$Z = UV^T \quad (21)$$

In (21), U is an m x m real unitary matrix whereas V is an n x n real unitary matrix and $\sum$ is an m x n rectangular diagonal matrix with non-negative real numbers on the diagonal.

Once Z is computed, the respective optimal values of parameters could be identified by dividing the values of X by 4, since there are four variables in the multi-criteria heuristic function. One potential solution to generate Z would be the utilisation of Kabsch algorithm that would generate the optimal Z by minimising the root mean squared deviation between X and Y (Blatov et al., 2019). However, other solutions pertaining to the generation of Z can be investigated and evaluated.

In the next sub-section, we present the threats to validity.

6.4 Threats to Validity

In this section, we present the threats to validity that can potentially affect the outcomes reported in this prioritisation study. This study was focused on the prioritisation of useful reviews that are expressed in natural language, and hence our developed automated prioritisation methods were only evaluated for their appropriateness at prioritising useful reviews.

6.4.1 Internal Validity

We have mitigated several threats related to the subjectivity involved in manually assigning the priorities to informative reviews by: (a) inheriting essential guidelines from the pertinent prioritisation study (Licorish et al., 2017), (b) rigorously studying what types of useful reviews the actual app developers are concerned with and (c) making efficient use of the feedback provided by the app developers. All the essential information including the priority assigning guidelines (refer to table 6.1) were discussed among the three labellers for common understanding, before the reliability assessments were conducted which returned fair to substantial agreements. Follow up discussions were held to establish consensus before generating the appropriate results and finalising the particular outcomes. In

addition, we have performed external evaluation in the final phase (i.e., prioritisation of useful reviews) and have achieved substantial results that confirm the valid construction of ground truth (i.e., labelled datasets) in phases 2 and 4. Furthermore, we have selected four prominent methods for performing the prioritization of informative reviews. However, app developers (and other stakeholders) may also favour other methods for prioritization purpose (e.g., end-users' geographic location). The impact of such methods is not investigated in this study. However, we believe that our multi-criteria heuristic function is flexible. It would thus be easy to incorporate any additional methods, modify any existing ones or remove the unnecessary ones depending upon the objective of prioritisation.

6.4.2 External Validity

The external evaluation participants may have assigned priorities based on their intuition and experience of apps usage; however, these individuals were properly introduced to the work and guided accordingly on the assigning of priorities. The application of the developed individual prioritisation method was tested on useful reviews of four apps. Hence, the generalisability of the method could be further evaluated through the use of additional useful reviews from several apps. However, the accuracy and time requirements of the proposed individual prioritisation method is substantial in terms of the validation of the method. We used a computer with specific hardware configuration (refer to Section 4.5), which may limit the generalisability of the reported time results, however the pattern of results were consistent across the datasets and so this was not a threat to the pattern of results observed. Furthermore, the objective of the proposed prioritisation method is to generate prioritised list of the useful reviews but the decision of addressing certain prioritised useful reviews is totally dependent on the judgements of app developers for the given app maintenance and evolution cycle. This is because, only app developers are aware of the constraints such as feasibility, cost, time and so on that are imposed upon them to influence such decision.

6.4.3 Construct Validity

To construct the ground truth data to prioritise useful reviews we followed the well-established rules from the pertinent study to label the app reviews and recommended practices from the software engineering discipline (consensus formation). However, another alternative to construct this ground truth data would be to approach the app developers of the respective apps to obtain the prioritised set of reviews for evaluating the performance of the prioritisation method.

In the next chapter, we provide the concluding remarks of each phase, research contributions and summary of implications along with the potential future work.

7 Conclusion

In this chapter, we present the conclusions related to the four phases of research that were conducted in this research study, research contributions, along with a summary of implications and potential future work. As stated in the chapters 1, 2 and 3, the findings of the first phase influenced the next three phases that are linked subsequently. In section 7.1 we provide the conclusions drawn from the conducted systematic mapping study on requirements prioritisation (Phase 1), which is followed by the conclusions of the pilot study on automated filtering of useful reviews (Phase 2). Next, the conclusions for the pilot study on the approach towards automated taxonomy generation (Phase 3), and the pilot studies on the group-based prioritisation method and individual prioritisation method along with the full-scale study on individual prioritisation method are provided (Phase 4). In the subsequent sections we highlight the research contributions (Section 7.2) and the summary of implications along with the potential future work (Section 7.3).

7.1 Summary of Outcomes

This section provides a summary of outcomes for the four research phases of the study.

7.1.1 Phase 1 - Systematic Mapping Study (RQ1)

Stakeholders often provide requirements before the development of a product begins, and log feedback containing feature requests, bugs or enhancements for post-release product improvements. Product developers at times face challenges in terms of deciding which requirements or feedback to address and in what order during the product development or the product maintenance and evolution cycles. This is particularly evident when stakeholders are provided with an online platform to provide their requirements or feedback pertaining to software products. Therefore, software developers are on the lookout for efficient and reliable prioritisation methods to aid in deciding which crucial requirements or feedback to address initially. Numerous prioritisation methods exist, and these are utilised based on the orientation of a particular prioritisation application as these methods have their own merits and demerits.

While requirements prioritisation methods assist with the requirements prioritisation process under several conditions, challenges are encountered when there are numerous requirements to prioritise. Some of the prominent challenges are: lack of scalability of the particular requirements prioritisation method or the dependency of the particular requirements prioritisation method on domain knowledge and priority preferences of stakeholders to perform prioritisation. In addition, it is established that such methods demand much from stakeholders when the number of requirements to prioritise increase significantly, and particularly in crowdsourced contexts such as app reviews. However, our proposed prioritisation methods have addressed these challenges. That said, previous review studies did not

perform evaluations across full range of requirements prioritisation methods that are present in the studies on requirements prioritisation belonging to different disciplines. Thus, in the first phase of the study, we have exploited this opportunity and conducted a comprehensive systematic mapping study on requirements prioritisation that highlights the strength of evidence that is available on requirements prioritisation. To achieve this, we answered six research questions, analysing the interest in requirements prioritisation over time, the publication venues of the studies on requirements prioritisation and the disciplines of these studies (RQ1.1). We next investigated the approaches that are used for studying requirements prioritisation (RQ1.2) and the types of contributions that are provided for addressing the requirements prioritisation challenge (RQ1.3). Next, we identified the actual requirements prioritisation methods (RQ1.4) and the dimensions that were evaluated for requirements prioritisation methods (RQ1.5). Finally, we examined the performance outcomes of the various evaluations, and evidence of relationships between attributes of the requirements prioritisation methods and their performance outcomes (RQ1.6).

To summarise the outcomes of phase 1, the findings show that there has been steady interest in requirements prioritisation over the years. We observed that most of the studies are published in conferences and journals, in the discipline of software engineering, with product manufacturing also featuring eminently. Moreover, we found out that the majority of the studies focused on requirements prioritisation targeted evaluated solutions. We observed that researchers have also often proposed solutions (i.e., solutions that were not evaluated) or provided some type of simulation. The contributions towards addressing requirements prioritisation challenges ranged from hybrid methods to tools, and some hybrid methods harnessed the strengths of multiple methods while attempting to avoid the methods' drawbacks. That said, we identified eight dimensions that were evaluated for empirical requirements prioritisation studies, with requirements prioritisation methods largely evaluated for their operational demonstration, while the examined attributes had limited effects on requirements prioritisation methods' outcomes. We also observed that out of the 157 requirements prioritisation methods, 67 of these methods were part of multiple studies. AHP, CV, QFD, NA and PG were among the top ten requirements prioritisation methods that were researched. While there exists an opportunity to perform further evaluations on requirements prioritisation studies, our findings reveal that the development of new methods may efficiently address the encountered requirements prioritisation challenge if they are inspired by hybrid methods. The performance trade-offs of such methods are to be expected based on their performance targets. In the next sub-section, we provide the summary of outcomes related to the filtering of useful reviews.

7.1.2 Phase 2 - Useful Reviews Filtering (RQ2)

In the pilot study, we investigated the Multinomial Naïve Bayes variants for their feasibility and utility towards the filtering of useful reviews. Previously, many studies have proposed filtering approaches to

extract reviews of interest for app developers. However, the approach involving Expectation Maximization of Multinomial Naïve Bayes had shown the most promise. Therefore, in our pilot study, we investigated the performances of six variants of Multinomial Naïve Bayes. The results of this pilot study suggest that Expectation Maximisation Multinomial Naïve Bayes with Laplace smoothing (variant IV) and Complement Naïve Bayes with Laplace Smoothing (variant VI) may be best suited for filtering useful reviews for further app maintenance and evolution operations such as meaningful data analysis and visualisation, classification or prioritisation. In the next sub-section we provide the summary of outcomes related to the classification of useful reviews based on the automatically generated taxonomy.

7.1.3 Phase 3 - Classification of Useful Reviews (RQ3)

The need to generate an automated taxonomy for grouping useful reviews was a requirement of the group-based prioritisation method. Hence, we conducted a pilot study to validate the feasibility of our proposed approach. By doing so, we found out that previous studies on classification of reviews pertaining to apps have worked towards classifying and analysing numerous reviews in support of app maintenance and evolution cycles. Generally, the proposed classification approaches utilise a taxonomy that is manually derived from domain knowledge to classify reviews having similar characteristics into specific groups. However, such domain knowledge needs to be made available from experts and is often generalised (shallow), which forces app developers to manually analyse each review after the classification process is completed. Moreover, as the number of reviews increase, scalability challenges are encountered for classification approaches that are driven by manually derived taxonomies. We addressed these drawbacks in this pilot study and developed a novel approach that automatically generates a taxonomy to group reviews into dynamically created groups of interest without being dependent on the availability of domain knowledge. Based on the empirical evaluation conducted in this pilot study, the outcome of our proposed approach compares substantially to the one that was manually derived, and thus, could be useful for grouping useful reviews. In the next sub-section we provide the summary of outcomes related to the prioritisation of useful reviews.

7.1.4 Phase 4 - Automated Prioritisation of Useful Reviews (RQ4)

Previous studies on requirements prioritisation have attempted to address the challenge to prioritise numerous requirements but we observed that these methods were dependent on stakeholders' preferences and domain knowledge to prioritise the requirements or lacked scalability. Subsequently, only two studies from the app reviews domain were aimed at prioritisation of reviews pertaining to the apps, but these works lacked essential dimensions to measure their prioritisation performance further bringing into question their suitability. In the pilot study, we addressed the limitations of the previous studies by proposing novel automated prioritisation methods (i.e., group-based and individual) for prioritising numerous useful reviews. Based on the empirical evaluations reported in the pilot study,

our proposed methods are completely automated in comparison to manual ones that are dependent on the availability of domain knowledge or priority preferences of the stakeholders. In addition, dimensions such as accuracy and time are found to be crucial in benchmarking the prioritisation performance of these methods as app developers have to address several critical useful reviews in time constrained app maintenance and evolution cycles. As the individual prioritisation method outperformed the group-based prioritisation method in terms of the accuracy and time dimensions, we performed a full-scale evaluation of the individual prioritisation method. Our outcomes show that the results generated by the individual prioritisation method for useful reviews belonging to different sets of app are promising. Therefore, this method could be of potential use to app developers who are bound by time constraints to identify and address issues from numerous useful reviews in the app maintenance and evolution cycles.

7.2 Contributions

In this section, we highlight the key contributions that are provided for the software engineering community.

Firstly, we contribute a systematic mapping study protocol for studies related to requirements prioritisation, which provides classification schemes to categorise the studies on requirements prioritisation or similar work for meaningful interpretations. The proposed classification schemes on research approaches and contributions have been specifically developed for the studies on requirements prioritisation. The protocol also assisted in uncovering research interest in requirements prioritisation along with the different venues of publications and disciplines in which requirements prioritisation is considered. Along with these, we were able to uncover several requirements prioritisation methods (empirical and non-empirical) and identify the essential dimensions that are crucial towards the evaluation of requirements prioritisation methods.

Secondly, we contribute an approach to automatically filter useful reviews using a set of predefined rules and a recommended Multinomial Naïve Bayes variant. The recommendation related to the variant being, the semi supervised variant Expectation Maximisation of Multinomial Naïve Bayes with Laplace Smoothing (variant IV) is best suited overall. However, app developers can utilise the supervised variant Complement Naïve Bayes with Laplace smoothing (variant VI) if the app developers have substantial number of reviews whose labels (useful or non-useful) are imbalanced.

Thirdly, we contribute a preliminary approach that automatically generates a taxonomy from useful reviews for classification purpose. The approach is best suited when there is unavailability of domain knowledge (e.g., predefined manual taxonomy) to perform classification. It can also be used when app developers need to generate a fine-grained taxonomy reflecting prioritised list of app features and their association with requests, bugs or enhancements.

Fourthly, we contribute through the development and evaluation of two automated prioritisation methods for prioritising useful reviews. These methods driven by a multi-criteria heuristic function are independent of the stakeholders' priority preferences and domain knowledge to prioritise useful reviews. In addition, the multi-criteria heuristic function provides the flexibility to add, modify or remove prioritisation methods to support application-oriented prioritisation. For example, the prioritisation application requiring the useful reviews being prioritised based on the prominent end-users of the app.

Finally, the empirically evaluated and developed requirements prioritisation solution is demonstrated as a web-based tool available at: https://recptool.otagointeractive.nz/

7.3 Implications and Future Work

In this sub-section, we summarise implications and potential future work related to the four phases of the study. With the identification of requirements prioritisation studies from multiple disciplines, researchers from one discipline may seek guidelines from studies from other disciplines to effectively solve the particular encountered requirements prioritisation problem. Moreover, with the knowledge of the developed classification schemes and identified requirements prioritisation methods, researchers could work towards the development of a hybrid requirements prioritisation method that harnesses the strengths of multiple methods while avoiding their drawbacks. With regards to this, there is scope to develop a taxonomy for the comparison of the requirements prioritisation methods across different disciplines. Furthermore, the development of an application specific requirements prioritisation method could benefit from utilisation of relevant dimensions. For instance, researchers aiming to accurately and rapidly prioritise a product's requirements or feedback based on the dependencies that exist among the requirements or feedback could benefit from studies covering the accuracy, time and requirements dependency dimensions and utilise these dimensions to their advantage. Practitioners may also be able to use our insights when addressing the requirements prioritisation challenge.

In addition, software engineering practitioners could benefit from the developed automated filtering approach (i.e., information retrieval) as this approach identifies and extracts logged requests, bugs or enhancements related to software products (e.g., app) logged by the products' stakeholders (e.g., product's end-users). In addition, there exists a research opportunity to investigate and evaluate techniques that generate discriminative features (i.e., words) for learning purpose that can help increase the prediction accuracy and F-Measure of the Multinomial Naïve Bayes variants IV (Expectation Maximisation of Multinomial Naïve Bayes with Laplace Smoothing) and VI (Complement Naïve Bayes with Laplace Smoothing) along with the addressing of the problem of independence assumption made by the Multinomial Naïve Bayes variants.

The practitioners can also benefit from the proposed automated taxonomy generation approach to build a taxonomy which indicates requests, bugs or enhancements associated with the prominent product's features. The application of such an approach is best suited when stakeholders log bugs, requests or enhancements pertaining to product features that are contextually similar. Furthermore, with regards to the COALS method that was utilised to automatically generate the taxonomy used the default threshold value. However, outcomes of COALS can be evaluated using different threshold settings. In addition, COALS could be integrated with SVD on the appropriate SVD parameter value to generate potential optimal data necessary towards the generation of taxonomy.

Finally, through means of this conducted PhD study, practitioners can gain insights on how automated prioritisation methods can be developed to prioritise logged requests, bugs or enhancements pertaining to a software product. The key aspect in such a scenario being the identification of essential criteria required to drive the prioritisation process and developing the relevant methods to fulfil the criteria along with the utilisation of appropriate dimensions. That said, further research towards automated tuning of the parameters of the multi-criteria heuristic function to generate potential optimal prioritisation results reflecting increases in accuracy and reduction in time required for prioritisation can be conducted. In addition, the utility of other dimensions (e.g., computational complexity) towards prioritisation could also be investigated and evaluated. Beyond useful reviews, the validity of the prioritisation methods could also be investigated on bugs and requests that are logged on software repositories such as Jira, GitHub and so on. Such follow up research holds promise for the continuous evolution of prioritisation methods.

References

Aasem, M., Ramzan, M., & Jaffar, A. (2010, 14-16 June 2010). *Analysis and optimization of software requirements prioritization techniques.* Paper presented at the 2010 International Conference on Information and Emerging Technologies. DOI: 10.1109/ICIET.2010.5625687

Abou-Elseoud, M. A., Nasr, E. S., & Hefny, H. A. (2016, 20-21 Dec. 2016). *Enhancing requirements prioritization based on a hybrid technique.* Paper presented at the 2016 11th International Conference on Computer Engineering & Systems (ICCES). DOI: 10.1109/ICCES.2016.7822009

Abran, A., Moore, J. W., Bourque, P., Dupuis, R., & Tripp, L. (2004). Software engineering body of knowledge. *IEEE Computer Society, Angela Burgess*. DOI: 10.1109/52.805471

Achimugu, P., & Selamat, A. (2015). A Hybridized Approach for Prioritizing Software Requirements Based on K-Means and Evolutionary Algorithms. In A. T. Azar & S. Vaidyanathan (Eds.), *Computational Intelligence Applications in Modeling and Control* (pp. 73-93). Springer International Publishing. DOI: https://doi.org/10.1007/978-3-319-11017-2_4

Achimugu, P., Selamat, A., & Ibrahim, R. (2014). *A Clustering Based Technique for Large Scale Prioritization during Requirements Elicitation*. DOI: https://doi.org/10.1007/978-3-319-07692-8_59

Achimugu, P., Selamat, A., & Ibrahim, R. (2016). *ReproTizer: A Fully Implemented Software Requirements Prioritization Tool*. Paper presented at the Transactions on Computational Collective Intelligence XXII, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-662-49619-0_5

Achimugu, P., Selamat, A., Ibrahim, R., & Mahrin, M. N. (2014a). An Adaptive Fuzzy Decision Matrix Model for Software Requirements Prioritization. In J. Sobecki, V. Boonjing, & S. Chittayasothorn (Eds.), *Advanced Approaches to Intelligent Information and Database Systems* (pp. 129-138). Springer International Publishing. DOI: https://doi.org/10.1007/978-3-319-05503-9_13

Achimugu, P., Selamat, A., Ibrahim, R., & Mahrin, M. N. (2014b). A systematic literature review of software requirements prioritization research. *Information and Software Technology, 56*(6), 568-585. DOI: https://doi.org/10.1016/j.infsof.2014.02.001

Aggarwal, C., & Zhai, C. (2012). *Mining Text Data*: Springer Science Business Media. DOI: https://doi.org/10.1007/978-1-4614-3223-4

Ahmad, A., Shahzad, A., Padmanabhuni, V. K., Mansoor, A., Joseph, S., & Arshad, Z. (2011, 10-12 June 2011). *Requirements prioritization with respect to Geographically Distributed Stakeholders.* Paper presented at the 2011 IEEE International Conference on Computer Science and Automation Engineering. DOI: 10.1109/CSAE.2011.5952853

Allan, J., Carbonell, J. G., Doddington, G., Yamron, J., & Yang, Y. (1998). Topic detection and tracking pilot study final report. DOI: https://doi.org/10.1007/978-1-4615-0933-2_1

Allen, J. R. L. (1986). Earthquake magnitude-frequency, epicentral distance, and soft-sediment deformation in sedimentary basins. *Sedimentary Geology, 46*(1), 67-75. DOI: https://doi.org/10.1016/0037-0738(86)90006-0

Aly, M. (2005). Survey on multiclass classification methods. *Neural Networks, 19*, 1-9. Retrieved from: https://tinyurl.com/y6ah82s9

Aral, S. (2014). The problem with online ratings. *MIT Sloan Management Review, 55*(2), 47. Retrieved from: https://tinyurl.com/yxbzjb4h

Archak, N., Ghose, A., & Ipeirotis, P. G. (2007). *Show me the money!: deriving the pricing power of product features by mining consumer reviews.* Paper presented at the 13th ACM SIGKDD international conference on Knowledge discovery and data mining. DOI: 10.1145/1281192.1281202

Arcuri, A., & Fraser, G. (2013). Parameter tuning or default values? An empirical investigation in search-based software engineering. *Empirical Software Engineering, 18*(3), 594-623. DOI: https://doi.org/10.1007/s10664-013-9249-9

Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statist. Surv., 4*, 40-79. DOI: 10.1214/09-SS054

Armacost, R. L., Componation, P. J., Mullens, M. A., & Swart, W. W. (1994). An AHP Framework For Prioritizing Customer Requirements In QFD: An Industrial Housing Application. *IIE Transactions, 26*(4), 72-79. DOI: 10.1080/07408179408966620

Asghar, A. R., Tabassum, A., Bhatti, S. N., & Jadi, A. M. (2017, 19-21 April 2017). *Impact and challenges of requirements elicitation & prioritization in quality to agile process: Scrum as a case scenario.* Paper presented at the 2017 International Conference on Communication Technologies (ComTech). DOI: 10.1109/COMTECH.2017.8065749

Asghar, M. W., Marchetto, A., Susi, A., & Scanniello, G. (2013, 5-8 March 2013). *Maintainability-Based Requirements Prioritization by Using Artifacts Traceability and Code Metrics*. Paper presented at the 2013 17th European Conference on Software Maintenance and Reengineering. DOI: 10.1109/CSMR.2013.62

Atukorala, N. L., Chang, C. K., & Oyama, K. (2016). *Situation-Oriented Evaluation and Prioritization of Requirements*, Singapore. DOI: https://doi.org/10.1007/978-981-10-3256-1_2

Babar, M. I., Ramzan, M., & Ghayyur, S. A. K. (2011, 11-13 July 2011). *Challenges and future trends in software requirements prioritization*. Paper presented at the International Conference on Computer Networks and Information Technology. DOI: 10.1109/ICCNIT.2011.6020888

Bacchelli, A., Sasso, T. D., D'Ambros, M., & Lanza, M. (2012). *Content classification of development emails*. Paper presented at the 34th International Conference on Software Engineering, Zurich, Switzerland. DOI: 10.5555/2337223.2337268

Bagnall, A. J., Rayward-Smith, V. J., & Whittley, I. M. (2001). The next release problem. *Information and Software Technology*, *43*(14), 883-890. DOI: https://doi.org/10.1016/S0950-5849(01)00194-X

Bajaj, P., & Arora, V. (2013). Multi-person decision-making for requirements prioritization using fuzzy AHP. *SIGSOFT Softw. Eng. Notes*, *38*(5), 1-6. DOI: 10.1145/2507288.2507302

Bebensee, T., van de Weerd, I., & Brinkkemper, S. (2010). *Binary Priority List for Prioritizing Software Requirements*, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-642-14192-8_8

Bennett, K. H., & Rajlich, V. T. (2000). *Software maintenance and evolution: a roadmap*. Paper presented at the Conference on The Future of Software Engineering, Limerick, Ireland. DOI: https://doi.org/10.1145/336512.336534

Berander, P., & Andrews, A. (2005). Requirements Prioritization. In A. Aurum & C. Wohlin (Eds.), *Engineering and Managing Software Requirements* (pp. 69-94). Berlin, Heidelberg: Springer Berlin Heidelberg. DOI: https://doi.org/10.1007/3-540-28244-0_4

Berander, P., & Jonssen, P. (2006). Hierarchical Cumulative Voting (HCV) — Prioritization of Requirements In Hierarchies. *International Journal of Software Engineering and Knowledge Engineering*, *16*(06), 819-849. DOI: 10.1142/s0218194006003026

Berander, P., & Svahnberg, M. (2009). Evaluating two ways of calculating priorities in requirements hierarchies – An experiment on hierarchical cumulative voting. *Journal of Systems and Software, 82*(5), 836-850. DOI: https://doi.org/10.1016/j.jss.2008.11.841

Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. *The Journal of Machine Learning Research, 13*(1), 281-305. DOI: 10.5555/2188385.2188395

Bergstra, J. S., Bardenet, R., Bengio, Y., & Kégl, B. (2011). *Algorithms for hyper-parameter optimization.* Paper presented at the Advances in Neural Information Processing Systems. DOI: 10.5555/2986459.2986743

Blatov, I. A., Kitaeva, E. V., Shevchenko, A. P., & Blatov, V. A. (2019). A universal algorithm for finding the shortest distance between systems of points. *Acta Crystallographica Section A: Foundations and Advances, 75*(6). DOI: 10.1107/S2053273319011628

Blot, A., Pernet, A., Jourdan, L., Kessaci-Marmion, M.-É., & Hoos, H. H. (2017). *Automatically Configuring Multi-objective Local Search Using Multi-objective Optimisation*. DOI: https://doi.org/10.1007/978-3-319-54157-0_5

Boehm, B., & Port, D. (2001). *Educating software engineering students to manage risk.* Paper presented at the 23rd International Conference on Software Engineering. DOI: 10.1109/ICSE.2001.919133

Bollegala, D., Matsuo, Y., & Ishizuka, M. (2011). A web search engine-based approach to measure semantic similarity between words. *IEEE Transactions on Knowledge and Data Engineering, 23*(7), 977-990. DOI: 10.1109/TKDE.2010.172

Boullé, M. (2006). MODL: A Bayes optimal discretization method for continuous attributes. *Machine Learning, 65*(1), 131-165. DOI: 10.1007/s10994-006-8364-x

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software, 80*(4), 571-583. DOI: https://doi.org/10.1016/j.jss.2006.07.009

Broder, A. (2002). A taxonomy of web search. *SIGIR Forum, 36*(2), 3-10. DOI: 10.1145/792550.792552

Brunetti, G., & Golob, B. (2000). A feature-based approach towards an integrated product model including conceptual design information. *Computer-Aided Design, 32*(14), 877-887. DOI: https://doi.org/10.1016/S0010-4485(00)00076-2

Burgess, C. (1998). From simple associations to the building blocks of language: Modeling meaning in memory with the HAL model. *Behavior research methods, instruments, & computers, 30*(2), 188-198. DOI: https://doi.org/10.3758/BF03200643

Calders, T., & Verwer, S. (2010). Three naive Bayes approaches for discrimination-free classification. *Data Mining and Knowledge Discovery, 21*(2), 277-292. DOI: https://doi.org/10.1007/s10618-010-0190-x

Carod, N. M., & Cechich, A. (2010, 22-27 Aug. 2010). *Cognitive Profiles in Understanding and Prioritizing Requirements: A Case Study.* Paper presented at the 2010 Fifth International Conference on Software Engineering Advances. DOI: 10.1109/ICSEA.2010.58

Caruana, R., & Niculescu-Mizil, A. (2006). *An empirical comparison of supervised learning algorithms.* Paper presented at the 23rd international conference on Machine learning. DOI: 10.1145/1143844.1143865

Chea, R. S., Morris, J., & Prabhu, G. (2009). System for dynamic product summary based on consumer-contributed keywords: Google Patents. Retrieved from: https://tinyurl.com/y6fdcqmz

Chen, N., Lin, J., Hoi, S. C. H., Xiao, X., & Zhang, B. (2014). *AR-miner: mining informative reviews for developers from mobile app marketplace*. Paper presented at the 36th International Conference on Software Engineering, Hyderabad, India. DOI: https://doi.org/10.1145/2568225.2568263

Chen, Y. Z., & Yu, Q. (2014, 17-19 Aug. 2014). *A fuzzy game approach to prioritize customer requirements in Quality Function Deployment.* Paper presented at the 2014 International Conference on Management Science & Engineering 21th Annual Conference Proceedings. DOI: 10.1109/ICMSE.2014.6930230

Chopra, R. K., Gupta, V., & Chauhan, D. S. (2016). Experimentation on accuracy of non functional requirement prioritization approaches for different complexity projects. *Perspectives in Science, 8*, 79-82. DOI: https://doi.org/10.1016/j.pisc.2016.04.001

Ciurumelea, A., Panichella, S., & Gall, H. C. (2018, 27 May-3 June 2018). *Poster: Automated User Reviews Analyser.* Paper presented at the 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion). DOI: https://doi.org/10.1145/3183440.3194988

Ciurumelea, A., Schaufelbühl, A., Panichella, S., & Gall, H. C. (2017, 20-24 Feb. 2017). *Analyzing reviews and code of mobile apps for better release planning.* Paper presented at the 2017 IEEE

24th International Conference on Software Analysis, Evolution and Reengineering (SANER). DOI: 10.1109/SANER.2017.7884612

Cleland-Huang, J., & Mobasher, B. (2008). *Using data mining and recommender systems to scale up the requirements process*. Paper presented at the 2nd international workshop on Ultra-large-scale software-intensive systems, Leipzig, Germany. DOI: 10.1109/RE.2008.47

Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2007). Automated classification of non-functional requirements. *Requir. Eng., 12*(2), 103-120. DOI: 10.1007/s00766-007-0045-1

Collins, M. (2012). The Naive Bayes Model, Maximum-Likelihood Estimation, and the EM algorithm. *Lecture Notes*. Retrieved from: https://tinyurl.com/y5zpfooy

Cysneiros, L. M., & do Prado Leite, J. C. S. (2004). Nonfunctional requirements: From elicitation to conceptual models. *IEEE Transactions on Software Engineering, 30*(5), 328-350. DOI: 10.1109/TSE.2004.10

Das, A., Bandyopadhyay, S., & Gambäck, B. (2012). *Sentiment analysis: what is the end user's requirement?* Paper presented at the 2nd International Conference on Web Intelligence, Mining and Semantics. DOI: https://doi.org/10.1145/2254129.2254173

Dasgupta, P., Chakrabarti, P., & DeSarkar, S. (2013). *Multiobjective heuristic search: An introduction to intelligent search methods for multicriteria optimization*: Springer Science & Business Media. DOI: 10.1007/978-3-322-86853-4

De Jong, T. (2010). Cognitive load theory, educational research, and instructional design: some food for thought. *Instructional Science, 38*(2), 105-134. DOI: https://doi.org/10.1007/s11251-009-9110-0

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science, 41*(6), 391-407. DOI: https://doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9

del Campo, C., Pauser, S., Steiner, E., & Vetschera, R. (2016). Decision making styles and the use of heuristics in decision making. *Journal of Business Economics, 86*(4), 389-412. DOI: 10.1007/s11573-016-0811-y

Delia Ilie, U. L., & Andreas Kain. (2009). *Evaluation And Prioritization Of Cross Linked Requirements In The Automotive Development Process.* Paper presented at the International Design

Engineering Technical Conferences & Computers and Information in Engineering Conference, San Diego, California, USA. DOI: 10.1115/DETC2009-87249

Dhinakaran, V. T., Pulle, R., Ajmeri, N., & Murukannaiah, P. K. (2018, 20-24 Aug. 2018). *App Review Analysis Via Active Learning: Reducing Supervision Effort without Compromising Classification Accuracy*. Paper presented at the 2018 IEEE 26th International Requirements Engineering Conference (RE). DOI: 10.1109/RE.2018.00026

Di Sorbo, A., Panichella, S., Alexandru, C. V., Shimagaki, J., Visaggio, C. A., Canfora, G., & Gall, H. C. (2016). *What would users change in my app? summarizing app reviews for recommending software changes*. Paper presented at 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. DOI: http://dx.doi.org/10.1145/2950290.2950299

Di Sorbo, A., Panichella, S., Alexandru, C. V., Visaggio, C.A., & Canfora, G., (2017) SURF: Summarizer of User Reviews Feedback, Paper presented at 39th International Conference on Software Engineering Companion (ICSE-C): Buenos Aires, Argentina. DOI: 10.1109/ICSE-C.2017.5

Diebold, P., Schmitt, A., & Theobald, S. (2018). *Scaling agile: how to select the most appropriate framework*. Paper presented at the 19th International Conference on Agile Software Development: Companion, Porto, Portugal. DOI: https://doi.org/10.1145/3234152.3234177

Dillon, J. T. (1984). The Classification of Research Questions. *Review of Educational Research, 54*(3), 327-361. DOI: 10.3102/00346543054003327

Elsood, M. A. A., Hefny, H. A., & Nasr, E. S. (2014, 15-17 Dec. 2014). *A goal-based technique for requirements prioritization.* Paper presented at the 2014 9th International Conference on Informatics and Systems. DOI: 10.1109/INFOS.2014.7036697

Erk, K. (2010). *What is word meaning, really?:(and how can distributional models help us describe it?).* Paper presented at the 2010 Workshop on Geometrical Models of Natural Language Semantics. DOI: 10.5555/1870516.1870519

Evangelopoulos, N., Zhang, X., & Prybutok, V. R. (2012). Latent Semantic Analysis: five methodological recommendations. *European Journal of Information Systems, 21*(1), 70-86. DOI: 10.1057/ejis.2010.61

Evans, I. S. (1977). The selection of class intervals. *Transactions of the Institute of British Geographers*, 98-124. DOI: 10.2307/622195

Fabio, P., M. L.-V., G. Bavota, R. Oliveto, M. Di Penta, D. Poshyvanyk, A. De Lucia. (2015). *User reviews matter! Tracking crowdsourced reviews to support evolution of successful apps*. Paper presented at the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME), Bremen, Germany. DOI: 10.1109/ICSM.2015.7332475

Fadhl Hujainah, Rohani Binti. A. B., Basheer Al-Haimi and Abdullah B. (2016). Analyzing Requirement Prioritization Techniques Based on the Used Aspects. *Research Journal of Applied Sciences*, *11*, 327-332. DOI: 10.3923/rjasci.2016.327.332

Fang, X., & Zhan, J. (2015). Sentiment analysis using product review data. *Journal of Big Data, 2*(1), 5. DOI: https://doi.org/10.1186/s40537-015-0015-2

Felfernig, A., & Ninaus, G. (2012, 4-4 June 2012). *Group recommendation algorithms for requirements prioritization.* Paper presented at the 2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE). DOI: 10.1109/RSSE.2012.6233412

Filcek, G., Hojda, M., & Żak, J. (2017). A heuristic algorithm for solving a Multiple Criteria Carpooling Optimization (MCCO) problem. *Transportation Research Procedia, 27*, 656-663. DOI: https://doi.org/10.1016/j.trpro.2017.12.108

Fleiss, J. L., & Cohen, J. (1973). The equivalence of weighted kappa and the intraclass correlation coefficient as measures of reliability. *Educational and psychological measurement, 33*(3), 613-619. DOI: https://doi.org/10.1177/001316447303300309

Forouzani, S., Ahmad, R., & Gazerani, N. (2012, 24-26 April 2012). *Design of a teaching framework for software requirement prioritization.* Paper presented at the 2012 8th International Conference on Computing Technology and Information Management (NCM and ICNIT). Retrieved from: https://ieeexplore.ieee.org/document/6268608

Forrester, A., Sobester, A., & Keane, A. (2008). *Engineering design via surrogate modelling: a practical guide*: John Wiley & Sons. DOI: 10.1002/9780470770801

Forrester, A. I., Sóbester, A., & Keane, A. J. (2007). Multi-fidelity optimization via surrogate modelling. *Proceedings of the royal society a: mathematical, physical and engineering sciences*, *463*(2088), 3251-3269. DOI: https://doi.org/10.1098/rspa.2007.1900

Franceschini, F., Galetto, M., Maisano, D., & Mastrogiacomo, L. (2015). Prioritisation of engineering characteristics in QFD in the case of customer requirements orderings. *International Journal of Production Research, 53*(13), 3975-3988. DOI: 10.1080/00207543.2014.980457

Fu, B., Lin, J., Li, L., Faloutsos, C., Hong, J., & Sadeh, N. (2013). *Why people hate your app: making sense of user feedback in a mobile app store*. Paper presented at the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, Illinois, USA. DOI: https://doi.org/10.1145/2487575.2488202

Fuchs, D., & Fuchs, L. S. (2006). Introduction to response to intervention: What, why, and how valid is it? *Reading research quarterly, 41*(1), 93-99. DOI: 10.1598/RRQ.41.1.4

Fung, R. Y. K., Shouju, R., & Jinxing, X. (1996, 14-17 Oct 1996). *The prioritisation of attributes in customer requirement management.* Paper presented at the 1996 IEEE International Conference on Systems, Man and Cybernetics. Information Intelligence and Systems (Cat. No.96CH35929). DOI: 10.1109/ICSMC.1996.571198

Galvis Carreño, L. V., & Winbladh, K. (2013). *Analysis of user comments: an approach for software requirements evolution.* Paper presented at the 2013 International Conference on Software Engineering. DOI: 10.5555/2486788.2486865

Ganu, G., Elhadad, N., & Marian, A. (2009). *Beyond the stars: improving rating predictions using review text content.* Paper presented at the WebDB. Retrieved from: https://tinyurl.com/y5ns2rk7

Gao, C., Wang, B., He, P., Zhu, J., Zhou, Y., & Lyu, M. R. (2015). Paid: Prioritizing app issues for developers by tracking user reviews over versions. Paper presented at the 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE). DOI: 10.1109/ISSRE.2015.7381797

Gao, C., Zeng, J., Lo, D., Lin, C.-Y., Lyu, M. R., & King, I. (2018). INFAR: Insight extraction from app reviews. Paper presented at the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. DOI: https://doi.org/10.1145/3236024.3264595

Garg, N., Sadiq, M., & Agarwal, P. (2017). *GOASREP: Goal Oriented Approach for Software Requirements Elicitation and Prioritization Using Analytic Hierarchy Process,* Singapore. DOI: https://doi.org/10.1007/978-981-10-3156-4_28

Garg, U., & Singhal, A. (2017, 12-13 Jan. 2017). *Software requirement prioritization based on non-functional requirements.* Paper presented at the 2017 7th International Conference on Cloud Computing, Data Science & Engineering - Confluence. DOI: 10.1109/CONFLUENCE.2017.7943258

Gärtner, S., & Schneider, K. (2012, 2-2 June 2012). *A method for prioritizing end-user feedback for requirements engineering.* Paper presented at the 2012 5th International Workshop on Co-operative and Human Aspects of Software Engineering (CHASE). DOI: 10.1109/CHASE.2012.6223020

Ghose, A., & Ipeirotis, P. G. (2011). Estimating the Helpfulness and Economic Impact of Product Reviews: Mining Text and Reviewer Characteristics. *IEEE Transactions on Knowledge and Data Engineering, 23*(10), 1498-1512. DOI: 10.1109/TKDE.2010.188

Goul, M., Marjanovic, O., Baxley, S., & Vizecky, K. (2012, 4-7 Jan. 2012). *Managing the Enterprise Business Intelligence App Store: Sentiment Analysis Supported Requirements Engineering.* Paper presented at the 2012 45th Hawaii International Conference on System Sciences. DOI: 10.1109/HICSS.2012.421

Gower, J. C., & Dijksterhuis, G. B. (2004). *Procrustes problems* (Vol. 30): Oxford University Press on Demand. DOI: 10.1093/acprof:oso/9780198510581.001.0001

Grafton, J., Lillis, A. M., Ihantola, E. M., & Kihn, L. A. (2011). Threats to validity and reliability in mixed methods accounting research. *Qualitative Research in Accounting & Management*. DOI: https://doi.org/10.1108/11766091111124694

Greer, D., & Ruhe, G. (2004). Software release planning: an evolutionary and iterative approach. *Information and Software Technology, 46*(4), 243-253. DOI: https://doi.org/10.1016/j.infsof.2003.07.002

Groen, E., Doerr, J., & Adam, S. (2015). *Towards Crowd-Based Requirements Engineering: A Research Preview*. DOI: https://doi.org/10.1007/978-3-319-16101-3_16

Hajič, J., Raab, J., & Spousta, M. (2009). *Semi-supervised training for the averaged perceptron POS tagger.* Paper presented at the 12th Conference of the European Chapter of the Association for Computational Linguistics. DOI: 10.3115/1609067.1609152

Hofmann, T. (1999). *Probabilistic latent semantic analysis.* Paper presented at the Fifteenth Conference on Uncertainty in Artificial Intelligence. DOI: https://doi.org/10.1145/312624.312649

Hoon, L., Vasa, R., Schneider, J.G., & Grundy, J. (2013). An analysis of the mobile app review landscape: trends and implications. *Faculty of Information and Communication Technologies, Swinburne University of Technology, Tech. Rep*. Retrieved from: http://hdl.handle.net/1959.3/352848

Hosseini, M., Shahri, A., Phalp, K., Taylor, J., Ali, R., & Dalpiaz, F. (2015, 13-15 May 2015). *Configuring crowdsourcing for requirements elicitation.* Paper presented at the 2015 IEEE 9th International Conference on Research Challenges in Information Science (RCIS). DOI: 10.1109/RCIS.2015.7128873

Htay, S. S., & Lynn, K. T. (2013). Extracting product features and opinion words using pattern knowledge in customer reviews. *The Scientific World Journal, 2013*. DOI: https://doi.org/10.1155/2013/394758

Hutto, C. J., & Gilbert, E. (2015). *VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text*. Retrieved from: http://eegilbert.org/papers/icwsm14.vader.hutto.pdf

Iacob, C., & Harrison, R. (2013, 18-19 May 2013). *Retrieving and analyzing mobile apps feature requests from online reviews.* Paper presented at the 2013 10th Working Conference on Mining Software Repositories (MSR). DOI: 10.1109/MSR.2013.6624001

Iacob, C., Harrison, R., & Faily, S. (2014). *Online Reviews as First Class Artifacts in Mobile App Development*. DOI: https://doi.org/10.1007/978-3-319-05452-0_4

Inoki, M., Kitagawa, T., & Honiden, S. (2014, 26-26 Aug. 2014). *Application of requirements prioritization decision rules in software product line evolution.* Paper presented at the 2014 IEEE 5th International Workshop on Requirements Prioritization and Communication (RePriCo). DOI: 10.1109/RePriCo.2014.6895216

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112): Springer. DOI: https://doi.org/10.1007/978-1-4614-7138-7

Jiang, H., Zhang, J., Li, X., Ren, Z., Lo, D., Wu, X., & Luo, Z. (2019). Recommending new features from mobile app descriptions. ACM Transactions on Software Engineering and Methodology (TOSEM), 28(4), 1-29. DOI: https://doi.org/10.1145/3344158

Jiang, J. J., & Conrath, D. W. (1997). Semantic similarity based on corpus statistics and lexical taxonomy. *arXiv preprint cmp-lg/9709008*. Retrieved from: https://arxiv.org/pdf/cmp-lg/9709008.pdf

John, G. H., & Langley, P. (1995). *Estimating continuous distributions in Bayesian classifiers.* Paper presented at the Eleventh Conference on Uncertainty in Artificial Intelligence. DOI: https://arxiv.org/abs/1302.4964

Jung, Y. G., Kim, K. T., Lee, B., & Youn, H. Y. (2016, 19-21 Oct. 2016). *Enhanced Naive Bayes Classifier for real-time sentiment analysis with SparkR.* Paper presented at the 2016 International Conference on Information and Communication Technology Convergence (ICTC). DOI: 10.1109/ICTC.2016.7763455

Kadilar, C., & Cingi, H. (2003). Ratio estimators in stratified random sampling. *Biometrical Journal: Journal of Mathematical Methods in Biosciences, 45*(2), 218-225. DOI: https://doi.org/10.1007/978-3-642-71581-5_4

Kamvysi, K., Gotzamani, K., Andronikidis, A., & Georgiou, A. C. (2014). Capturing and prioritizing students' requirements for course design by embedding Fuzzy-AHP and linear programming in QFD. *European Journal of Operational Research, 237*(3), 1083-1094. DOI: https://doi.org/10.1016/j.ejor.2014.02.042

Karov, Y., & Edelman, S. (1998). Similarity-based word sense disambiguation. *Comput. Linguist., 24*(1), 41-59. DOI: 10.5555/972719.972722

Katsanos, C., Tselios, N., & Avouris, N. (2009). *Are Ten Participants Enough for Evaluating Information Scent of Web Page Hyperlinks?* Paper presented at the Human-Computer Interaction – INTERACT 2009, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-642-03655-2_45

Keertipati, S., Savarimuthu, B. T. R., & Licorish, S. A. (2016). *Approaches for prioritizing feature improvements extracted from app reviews*. Paper presented at the 20th International Conference on Evaluation and Assessment in Software Engineering, Limerick, Ireland. DOI: https://doi.org/10.1145/2915970.2916003

Khalid, M., Shehzaib, U., & Asif, M. (2015). A case of mobile app reviews as a crowdsource. *Int. J. Inf. Eng. Electron. Business, 7*(5). DOI: 0.5815/ijieeb.2015.05.06

Kim, J., Kim, C., Park, Y., & Lee, H. (2012). *Trends and relationships of smartphone application services: Analysis of apple app store using text mining-based network analysis.* Paper presented at the 4th ISPIM Innovation Symposium, Wellington, New Zealand. Retrieved from: https://tinyurl.com/yxfzbb8h

Kim, S. M., Pantel, P., Chklovski, T., & Pennacchiotti, M. (2006). *Automatically assessing review helpfulness.* Paper presented at the 2006 Conference on Empirical Methods in Natural Language Processing. DOI: 10.5555/1610075.1610135

Kiremire, A. R. (2011). The application of the pareto principle in software engineering. *Consulted January, 13*, 2016. Retrieved from: https://tinyurl.com/y5g3o8vp

Kitchenham. (2007). Guidelines for performing Systematic Literature Reviews in Software Engineering. Retrieved from: https://tinyurl.com/y682jy48

Ko, Y., Park, S., & Seo, J. (2000). *Web-based requirements elicitation supporting system using requirements categorization.* Paper presented at the Twelfth International Conference on Software Engineering and Knowledge Engineering (SEKE 2000), Chicago, USA. Retrieved from: http://home.donga.ac.kr/yjko/papers/ic2.pdf

Kohavi, R. (1995). *A study of cross-validation and bootstrap for accuracy estimation and model selection.* Paper presented at the IJCAI. DOI: 10.5555/1643031.1643047

Konkol, M., Brychcín, T., & Konopík, M. (2015). Latent semantics in named entity recognition. *Expert Systems with Applications, 42*(7), 3470-3479. DOI: https://doi.org/10.1016/j.eswa.2014.12.015

Košmerlj, A., Belyaeva, E., Leban, G., Grobelnik, M., & Fortuna, B. (2015). *Towards a complete event type taxonomy*. Paper presented at the 24th International Conference on World Wide Web. DOI: https://doi.org/10.1145/2740908.2742005

Kozima, H., & Furugori, T. (1993). *Similarity between words computed by spreading activation on an English dictionary*. Paper presented at the sixth conference on European chapter of the Association for Computational Linguistics, Utrecht, The Netherlands. DOI: 10.3115/976744.976772

Kravchenko, T.K., & Sergey, B. (2017). Prioritization of requirements for effective support of the communication process with customers of a commercial bank. *Business Informatics, 2*(40), 7-16. DOI: 10.17323/1998-0663.2017.2.7.16

Kukreja, N., Boehm, B., Payyavula, S. S., & Padmanabhuni, S. (2012, 24-28 Sept. 2012). *Selecting an appropriate framework for value-based requirements prioritization.* Paper presented at the 2012 20th IEEE International Requirements Engineering Conference (RE). DOI: 10.1109/RE.2012.6345819

Landauer, T. K., & Dumais, S. (2008). Latent semantic analysis. *Scholarpedia, 3*(11), 4356. DOI: https://doi.org/10.1002/0470018860.s00561

Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *biometrics*, 159-174. DOI: 10.2307/2529310

Laurent, P., Cleland-Huang, J., & Duan, C. (2007, 15-19 Oct. 2007). *Towards Automated Requirements Triage.* Paper presented at the 15th IEEE International Requirements Engineering Conference (RE 2007). DOI: 10.1109/RE.2007.63

Leacock, C., & Chodorow, M. (1998). Combining local context and WordNet similarity for word sense identification. *WordNet: An electronic lexical database*, *49*(2), 265-283. Retrieved from: https://ieeexplore.ieee.org/document/6287675

Lehtola, L. (2017). *Towards a Research Framework on Requirements Prioritization*. Retrieved from: https://tinyurl.com/y4jxamh8

Lehtola, L., & Kauppinen, M. (2006). Suitability of requirements prioritization methods for market-driven software product development. *Software Process: Improvement and Practice*, *11*(1), 7-19. DOI: 10.1002/spip.249

Leksin, V., & Vorontsov, K. (2008). The overfitting in probabilistic latent semantic models. *Pattern Recognition and Image Analysis: new information technologies (PRIA-9-2008): Nizhni Novgorod, Russian Federation*, *1*, 393-396. Retrieved from: https://tinyurl.com/yygx7cft

Li, J., Jeffery, R., Fung, K. H., Zhu, L., Wang, Q., Zhang, H., & Xu, X. (2012). *A Business Process-Driven Approach for Requirements Dependency Analysis*, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-642-32885-5_16

Licorish, S. A., Savarimuthu, B. T. R., & Keertipati, S. (2017). *Attributes that Predict which Features to Fix: Lessons for App Store Mining*. Paper presented at the 21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden. DOI: https://doi.org/10.1145/3084226.3084246

Lin, S., & Kernighan, B. W. (1973). An effective heuristic algorithm for the traveling-salesman problem. *Operations research*, *21*(2), 498-516. DOI: https://doi.org/10.1287/opre.21.2.498

Liu, K. (2000). *Semiotics in information systems engineering*: Cambridge University Press. DOI: 10.1017/CBO9780511543364

Lorigo, L., Haridasan, M., Brynjarsdóttir, H., Xia, L., Joachims, T., Gay, G., & Pan, B. (2008). Eye tracking and online search: Lessons learned and challenges ahead. *Journal of the American Society for Information Science and Technology*, *59*(7), 1041-1052. DOI: 10.1002/asi.20794

Lowd, D., & Domingos, P. (2005). *Naive Bayes models for probability estimation.* Paper presented at the 22nd International Conference on Machine learning. DOI: https://doi.org/10.1145/1102351.1102418

Lund, K., & Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments, & computers, 28*(2), 203-208. DOI: https://doi.org/10.3758/BF03204766

Luo, Q., Xu, W., & Guo, J. (2014). *A Study on the CBOW Model's Overfitting and Stability.* Paper presented at the 5th International Workshop on Web-scale Knowledge Representation Retrieval & Reasoning. DOI: https://doi.org/10.1145/2663792.2663793

Maalej, W., Kurtanović, Z., Nabil, H., & Stanik, C. (2016a). On the automatic classification of app reviews. *Requirements Engineering, 21*(3), 311-331. DOI: https://doi.org/10.1007/s00766-016-0251-9

Maalej, W., Nayebi, M., Johann, T., & Ruhe, G. (2016b). Toward Data-Driven Requirements Engineering. *IEEE Software, 33*(1), 48-54. DOI: 10.1109/MS.2015.153

Maedche, A., & Staab, S. (2000). *Discovering conceptual relations from text.* Paper presented at the ECAI. DOI: 10.5555/3006433.3006501

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization, 26*(6), 369-395. DOI: https://doi.org/10.1007/s00158-003-0368-6

Mayring, P. (2004). Qualitative content analysis. *A companion to qualitative research, 1*, 159-176. Retrieved from: https://tinyurl.com/yyzkulsq

McCallum, A., & Nigam, K. (2001). A Comparison of Event Models for Naive Bayes Text Classification, In *AAAI-98 workshop on learning for text categorization* (Vol. 752, No. 1, pp. 41-48). Retrieved from: https://tinyurl.com/h3z2hxe

Mccallum, M. L., & Bury, G. W. (2013). Google search patterns suggest declining interest in the environment. *Biodiversity and conservation, 22*(6-7), 1355-1367. DOI: https://doi.org/10.1007/s10531-013-0476-6

McIlroy, S., Ali, N., Khalid, H., & E. Hassan, A. (2016). Analyzing and automatically labelling the types of user issues that are raised in mobile app reviews. *Empirical Software Engineering, 21*(3), 1067-1106. DOI: 10.1007/s10664-015-9375-7

McIlroy, S., Shang, W., Ali, N., & Hassan, A. E. (2015). Is it worth responding to reviews? studying the top free apps in google play. *IEEE Software, 34*(3), 64-71. DOI: 10.1109/MS.2015.149

McZara, J., Sarkani, S., Holzer, T., & Eveleigh, T. (2015). Software requirements prioritization and selection using linguistic tools and constraint solvers—a controlled experiment. *Empirical Software Engineering, 20*(6), 1721-1761. DOI: 10.1007/s10664-014-9334-8

Michie, D., Spiegelhalter, D. J., & Taylor, C. (1994). Machine learning. *Neural and Statistical Classification, 13*. DOI: 10.2307/1269742

Mihalcea, R., Corley, C., & Strapparava, C. (2006). *Corpus-based and knowledge-based measures of text semantic similarity.* Paper presented at the AAAI. DOI: 10.5555/1597538.1597662

Misaghian, N., & Motameni, H. (2016). An approach for requirements prioritization based on tensor decomposition. *Requirements Engineering*. DOI: 10.1007/s00766-016-0262-6

Morse, J. M. (2000). Determining sample size: Sage Publications Sage CA: Thousand Oaks, CA. DOI: https://doi.org/10.1177/104973200129118183

Myers, L., & Sirois, M. J. (2004). Spearman correlation coefficients, differences between. *Encyclopedia of Statistical Sciences, 12*. DOI: https://doi.org/10.1002/0471667196.ess5050.pub2

Nepal, B., Yadav, O. P., & Murat, A. (2010). A fuzzy-AHP approach to prioritization of CS attributes in target planning for automotive product development. *Expert Systems with Applications, 37*(10), 6775-6786. DOI: https://doi.org/10.1016/j.eswa.2010.03.048

Ng, A. Y., & Jordan, M. I. (2002). *On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes.* Paper presented at the Advances in Neural Information Processing Systems. DOI: 10.5555/2980539.2980648

Nidhra, S., Satish, L. P. K., & Ethiraj, V. S. (2012, 5-7 Sept. 2012). *Analytical Hierarchy Process issues and mitigation strategy for large number of requirements.* Paper presented at the 2012 CSI Sixth International Conference on Software Engineering (CONSEG). DOI: 10.1109/CONSEG.2012.6349467

Nigam, K., Mccallum, A. K., Thrun, S., & Mitchell, T. (2000). Text Classification from Labeled and Unlabeled Documents using EM. *Machine Learning, 39*(2), 103-134. DOI: 10.1023/a:1007692713085

Ninaus, G. (2012). *Using group recommendation heuristics for the prioritization of requirements*. Paper presented at the sixth ACM conference on Recommender Systems, Dublin, Ireland. DOI: https://doi.org/10.1145/2365952.2366034

Nuseibeh, B., & Easterbrook, S. (2000). *Requirements engineering: a roadmap*. Paper presented at the Conference on The Future of Software Engineering, Limerick, Ireland. DOI: https://doi.org/10.1145/336512.336523

Oliveira, R. P., & Almeida, E. S. (2015, 21-22 Sept. 2015). *Requirements Evolution in Software Product Lines: An Empirical Study*. Paper presented at the 2015 IX Brazilian Symposium on Components, Architectures and Reuse Software. DOI: 10.1109/SBCARS.2015.11

Pagano, D., & Maalej, W. (2013, 15-19 July 2013). *User feedback in the appstore: An empirical study.* Paper presented at the 2013 21st IEEE International Requirements Engineering Conference (RE). DOI: 10.1109/RE.2013.6636712

Palma, F., Susi, A., & Tonella, P. (2011). *Using an SMT solver for interactive requirements prioritization*. Paper presented at the 19th ACM SIGSOFT Symposium and the 13th European Conference on Foundations of software engineering, Szeged, Hungary. DOI: 10.1145/2025113.2025124

Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., & Gall, H. C. (2016). *Ardoc: App reviews development oriented classifier*. Paper presented at the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. DOI: 10.1145/2950290.2983938

Panichella, S., Sorbo, A. D., Guzman, E., Visaggio, C. A., Canfora, G., & Gall, H. C. (2015, Sept. 29 2015-Oct. 1 2015). *How can i improve my app? Classifying user reviews for software maintenance and evolution.* Paper presented at the 2015 IEEE International Conference on Software Maintenance and Evolution (ICSME). DOI: 10.1109/ICSM.2015.7332474

Patro, S., & Sahu, K. K. (2015). Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*. DOI: 10.17148/IARJSET.2015.2305

Peng, W., Sun, T., Revankar, S., & Li, T. (2012). Mining the Voice of the Customer for Business Prioritization. *ACM Trans. Intell. Syst. Technol.*, *3*(2), 1-17. DOI: 10.1145/2089094.2089114

Pergher, M., & Rossi, B. (2013, 15-15 July 2013). *Requirements prioritization in software engineering: A systematic mapping study*. Paper presented at the 2013 3rd International Workshop on Empirical Requirements Engineering (EmpiRE). DOI: 10.1109/EmpiRE.2013.6615215

Perini, A., Susi, A., & Avesani, P. (2013). A Machine Learning Approach to Software Requirements Prioritization. *IEEE Transactions on Software Engineering, 39*(4), 445-461. DOI: 10.1109/TSE.2012.52

Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). *Systematic mapping studies in software engineering*. Paper presented at the 12th International Conference on Evaluation and Assessment in Software Engineering, Italy. DOI: 10.5555/2227115.2227123

Petrakis, E. G., Varelas, G., Hliaoutakis, A., & Raftopoulou, P. (2006). X-similarity: Computing semantic similarity between concepts from different ontologies. *Journal of Digital Information Management, 4*(4). DOI: https://tinyurl.com/y5p3b5vw

Philip Achimugu. (2014). Applying Fuzzy-TOPSIS Algorithm in Prioritizing Software Requirements. *New Trends in Software Methodologies, Tools and Techniques* (pp. 659-671): IOS Press. DOI: 10.3233/978-1-61499-434-3-659

Popli, R., Chauhan, N., & Sharma, H. (2014, 7-8 Feb. 2014). *Prioritising user stories in agile environment.* Paper presented at the 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT). DOI: 10.1109/ICICICT.2014.6781336

Potts, C. (1993). Software-engineering research revisited. *IEEE Software, 10*(5), 19-28. DOI: https://doi.org/10.1109/52.232392

Reisinger, J., & Mooney, R. J. (2010). *Multi-prototype vector-space models of word meaning*. Paper presented at the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, California. DOI: 10.5555/1857999.1858012

Ren, J., Lee, S. D., Chen, X., Kao, B., Cheng, R., & Cheung, D. (2009). *Naive bayes classification of uncertain data.* Paper presented at the 2009 Ninth IEEE International Conference on Data Mining. DOI: 10.1109/ICDM.2009.90

Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. (2003). *Tackling the poor assumptions of naive bayes text classifiers.* Paper presented at the 20th International Conference on Machine Learning (ICML-03). DOI: 10.5555/3041838.3041916

Rényi, A. (1961). *On measures of entropy and information.* Paper presented the Fourth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Contributions to the Theory of Statistics. Retrieved from: https://projecteuclid.org/euclid.bsmsp/1200512181

Robillard, M. P., Maalej, W., Walker, R. J., & Zimmermann, T. (2014). *Recommendation Systems in Software Engineering*: Springer Publishing Company, Incorporated. DOI: 10.1007/978-3-642-45135-5

Rodrigues, P., Silva, I. S., Barbosa, G. A. R., Coutinho, F. R. D. S., & Mourão, F. (2017). *Beyond the stars: towards a novel sentiment rating to evaluate applications in web stores of mobile apps.* Paper presented at the 26th International Conference on World Wide Web Companion. DOI: 10.1145/3041021.3054139

Rohde, D. L., Gonnerman, L. M., & Plaut, D. C. (2006). An improved model of semantic similarity based on lexical co-occurrence. *Communications of the ACM, 8*(627-633), 116. Retrieved from: https://tinyurl.com/y4r27zz4

Roma, P., & Ragaglia, D. (2016). Revenue models, in-app purchase, and the app performance: Evidence from Apple's App Store and Google Play. *Electronic Commerce Research and Applications, 17*, 173-190. DOI: https://doi.org/10.1016/j.elerap.2016.04.007

Rowley, J., & Slack, F. (2004). Conducting a literature review. *Management research news*. DOI: 10.1108/01409170410784185

Ryan, K., & Karlsson, J. (1997). *Prioritizing software requirements in an industrial setting*. Paper presented at the 19th International Conference on Software Engineering, Boston, Massachusetts, USA. DOI: https://doi.org/10.1145/253228.253453

Sadiq, M. (2017). A Fuzzy Set-Based Approach for the Prioritization of Stakeholders on the Basis of the Importance of Software Requirements. *IETE Journal of Research, 63*(5), 616-629. DOI: 10.1080/03772063.2017.1313140

Sadiq, M., Ghafir, S., & Shahid, M. (2009, 27-28 Oct. 2009). *An Approach for Eliciting Software Requirements and its Prioritization Using Analytic Hierarchy Process.* Paper presented at the 2009 International Conference on Advances in Recent Technologies in Communication and Computing. DOI: 10.1109/ARTCom.2009.58

Sadiq, M., Hassan, T., & Nazneen, S. (2017, 9-10 Feb. 2017). *AHP_GORE_PSR: Applying analytic hierarchy process in goal oriented requirements elicitation method for the prioritization of software requirements.* Paper presented at the 3rd International Conference on Computational Intelligence & Communication Technology (CICT). DOI: 10.1109/CIACT.2017.7977366

Sánchez, D., Batet, M., & Isern, D. (2011). Ontology-based information content computation. *Knowledge-Based Systems, 24*(2), 297-303. DOI: https://doi.org/10.1016/j.knosys.2010.10.001

Santos, R., Albuquerque, A., & Pinheiro, P. R. (2016). Towards the Applied Hybrid Model in Requirements Prioritization. *Procedia Computer Science, 91*, 909-918. DOI: https://doi.org/10.1016/j.procs.2016.07.109

Schaffer, C. (1993). Selecting a classification method by cross-validation. *Machine Learning, 13*(1), 135-143. DOI: https://doi.org/10.1007/BF00993106

Shah, F. A., Sirts, K., & Pfahl, D. (2018). *Simple App Review Classification with Only Lexical Features.* Paper presented at the ICSOFT. DOI: 10.5220/0006855901120119

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell system technical journal*, 27(3), 379-423. DOI: 10.1002/j.1538-7305.1948.tb00917.x

Shao, P. (2008). *Sample selection: an algorithm for requirements prioritization*. Paper presented at the 46th Annual Southeast Regional Conference, Auburn, Alabama. DOI: 10.1145/1593105.1593248

Sharma, J. R. (2007). Prioritizing customers requirements in QFD by integrating their interrelationship with the raw weights. *Journal of the Institution of Engineers, 88*, 7-11. DOI: https://doi.org/10.1080/10686967.2007.11918046

Sher, F., Jawawi, D. N. A., Mohamad, R., & Babar, M. I. (2014, 23-24 Sept. 2014). *Requirements prioritization techniques and different aspects for prioritization a systematic literature review protocol.* Paper presented at the 2014 8th. Malaysian Software Engineering Conference (MySEC). DOI: 10.1109/MySec.2014.6985985

Sheskin, D. J. (2003). *Handbook of parametric and nonparametric statistical procedures*: Chapman and Hall/CRC. DOI: https://doi.org/10.1201/9781420036268

Snoek, J., Larochelle, H., & Adams, R. P. (2012). *Practical bayesian optimization of machine learning algorithms.* Paper presented at the Advances in Neural Information Processing Systems. DOI: 10.5555/2999325.2999464

Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management, 45*(4), 427-437. DOI: https://doi.org/10.1016/j.ipm.2009.03.002

Solemon, B., Sahibuddin, S., & Abdul Azim Abd, G. (2008, 26-28 Aug. 2008). *Requirements engineering problems in 63 software companies in Malaysia.* Paper presented at the 2008 International Symposium on Information Technology. DOI: 10.1109/ITSIM.2008.4631911

Solow, D. (2007). Linear and nonlinear programming. *Wiley Encyclopedia of Computer Science and Engineering*. DOI: https://doi.org/10.1002/9780470050118.ecse219

Sommervile, I. (2009). *Software Engineering* (9 ed.). DOI: http://doi.ieeecomputersociety.org/10.1109/MC.1987.1663532

Somprasertsri, G., & Lalitrojwong, P. (2008). *A maximum entropy model for product feature extraction in online customer reviews.* Paper presented at the 2008 IEEE Conference on Cybernetics and Intelligent Systems. DOI: 10.1109/ICCIS.2008.4670882

Sorbo, A. D., Panichella, S., Alexandru, C. V., Shimagaki, J., Visaggio, C. A., Canfora, G., & Gall, H. C. (2016). *What would users change in my app? summarizing app reviews for recommending software changes*. Paper presented at the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Seattle, WA, USA. DOI: https://doi.org/10.1145/2950290.2950299

Storjohann, A. (2001). *Deterministic computation of the Frobenius form.* Paper presented at the 42nd IEEE Symposium on Foundations of Computer Science. DOI: https://doi.org/10.1109/SFCS.2001.959911

Stumpf, S., Rajaram, V., Li, L., Burnett, M., Dietterich, T., Sullivan, E., & Herlocker, J. (2007). *Toward harnessing user feedback for machine learning.* Paper presented at the 12th international conference on Intelligent user interfaces.DOI: https://doi.org/10.1145/1216295.1216316

Sundaram, S. K., Hayes, J. H., & Dekhtyar, A. (2005). Baselines in requirements tracing. *SIGSOFT Softw. Eng. Notes, 30*(4), 1-6. DOI: 10.1145/1082983.1083169

Sureka, A. (2014, 7-10 April 2014). *Requirements Prioritization and Next-Release Problem under Non-additive Value Conditions.* Paper presented at the 2014 23rd Australian Software Engineering Conference. DOI: 10.1109/ASWEC.2014.12

Syarifah Fazlin, & Seyed Fadzir. (2016). Requirement Prioritization Approaches and Evaluation Strategies: A Systematic Literature Review. *Journal of Engineering and Applied Sciences, 11*(6), 1201-1205. DOI: 10.1109/ACCESS.2018.2881755

Takahira, R., Tanaka-Ishii, K., & Dębowski, Ł. (2016). Entropy rate estimates for natural language—a new extrapolation of compressed large-scale corpora. *Entropy, 18*(10), 364. DOI: https://doi.org/10.3390/e18100364

Thabane, L., Ma, J., Chu, R., Cheng, J., Ismaila, A., Rios, L. P., Goldsmith, & C. H. (2010). A tutorial on pilot studies: the what, why and how. *BMC medical research methodology, 10*(1), 1. DOI: 10.1186/1471-2288-10-1

Thakurta, R. (2013). A framework for prioritization of quality requirements for inclusion in a software project. *Software Quality Journal, 21*(4), 573-597. DOI: 10.1007/s11219-012-9188-5

Thew, S., & Sutcliffe, A. (2017). Value-based requirements engineering: method and experience. *Requirements Engineering*. DOI: 10.1007/s00766-017-0273-y

Tian, J., Rudraraju, S., & Li, Z. (2004). Evaluating web software reliability based on workload and failure data extracted from server logs. *IEEE Transactions on Software Engineering, 30*(11), 754-769. DOI: 10.1109/TSE.2004.87

Turner, C. R., Fuggetta, A., Lavazza, L., & Wolf, A. L. (1999). A conceptual basis for feature engineering. *Journal of Systems and Software, 49*(1), 3-15. DOI: https://doi.org/10.1016/S0164-1212(99)00062-X

Voola, P., & Babu, A., V. (2017). Study of aggregation algorithms for aggregating imprecise software requirements' priorities. *European Journal of Operational Research, 259*(3), 1191-1199. DOI: https://doi.org/10.1016/j.ejor.2016.11.040

Voola, P., & Babu, A. V. (2013). Comparison of requirements prioritization techniques employing different scales of measurement. *SIGSOFT Softw. Eng. Notes, 38*(4), 1-10. DOI: 10.1145/2492248.2492278

Vu, P. M., Nguyen, T. T., & Nguyen, T. T. (2019). On building an automated responding system for app reviews: What are the characteristics of reviews and their responses?. arXiv preprint arXiv:1908.10816.

Walid Maalej., & Haader Nabil. (2015). *Bug report, feature request, or simply praise? On automatically classifying app reviews*. Paper presented at the 2015 IEEE 23rd International Requirements Engineering Conference (RE), Ottawa, Canada. DOI: 10.1109/RE.2015.7320414

Wang, C., Zhang, F., Liang, P., Daneva, M., & Sinderen, M. V. (2018). *Can app changelogs improve requirements classification from app reviews?: an exploratory study*. Paper presented at the

12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, Oulu, Finland. DOI: https://doi.org/10.1145/3239235.3267428

Wessing, S., Pink, R., Brandenbusch, K., & Rudolph, G. (2017). *Toward Step-Size Adaptation in Evolutionary Multiobjective Optimization*. DOI: https://doi.org/10.1007/978-3-319-54157-0_45

Wibisono, A., Zhao, Z., Belloum, A., & Bubak, M. (2008). *A Framework for Interactive Parameter Sweep Applications*, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-540-69389-5_55

Wieringa, R., Maiden, N., Mead, N., & Rolland, C. (2005). Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requir. Eng.*, *11*(1), 102-107. DOI: 10.1007/s00766-005-0021-6

Wilcox, R. R. (2011). *Introduction to robust estimation and hypothesis testing*: Academic press. DOI: https://doi.org/10.1016/C2010-0-67044-1

Wnuk, K., Regnell, B., & Schrewelius, C. (2009). *Architecting and Coordinating Thousands of Requirements – An Industrial Case Study*, Berlin, Heidelberg. DOI: https://doi.org/10.1007/978-3-642-02050-6_10

Wu, H. C., Luk, R. W. P., Wong, K. F., & Kwok, K. L. (2008). Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, *26*(3), 13. DOI: https://doi.org/10.1145/1361684.1361686

Xuemei, S., Dakun, Z., Xinming, D., & You, Z. (2008, 12-15 Oct. 2008). *Prioritizing design requirement in fuzzy quality function deployment.* Paper presented at the 2008 IEEE International Conference on Service Operations and Logistics, and Informatics. DOI: 10.1109/SOLI.2008.4682981

Yang, H., & Liang, P. (2015). *Identification and Classification of Requirements from App User Reviews*. DOI: 10.18293/SEKE2015-063

Yuan, Q., Cong, G., & Thalmann, N. M. (2012). *Enhancing naive bayes with various smoothing methods for short text classification.* Paper presented at the 21st International Conference on World Wide Web. DOI: 10.1145/2187980.2188169

Yutao Ma, Y. L., Haisu Zhang, & Guisheng Chen. (2012). A Hybrid Method for Prioritizing Software Requirements in terms of Use Cases. *Journal of Convergence Information Technology(JCIT)*, *7*(5). DOI: 10.4156/jcit.vol7.issue5.3

Zha, Z. J., Yu, J., Tang, J., Wang, M., & Chua, T.-S. (2014). Product aspect ranking and its applications. *IEEE Transactions on Knowledge and Data Engineering, 26*(5), 1211-1224. DOI: 10.1109/TKDE.2013.136

Zhang, C., Zhang, X., & Halstead-Nussloch, R. (2014). Assessment Metrics, Challenges And Strategies For Mobile Health Apps. *Issues in Information Systems, 15*(2). Retrieved from: https://tinyurl.com/y3v76su7

Zhang, L., & Liu, B. (2011). *Identifying noun product features that imply opinions.* Paper presented at the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2. DOI: 10.5555/2002736.2002849

Zhang, R., & Tran, T. (2008, 9-12 Dec. 2008). *An Entropy-Based Model for Discovering the Usefulness of Online Product Reviews.* Paper presented at the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology. DOI: 10.1109/WIIAT.2008.149

Zhaoling, L., Dongling, Z., & Qisheng, G. (2009, 17-19 June 2009). *A grey method of prioritizing engineering characteristics in QFD.* Paper presented at the 2009 Chinese Control and Decision Conference. DOI: 10.1109/CCDC.2009.5191557

Zhu, J., Wang, H., & Zhang, X. (2006). *Discrimination-based feature selection for multinomial naïve bayes text classification.* Paper presented at the International Conference on Computer Processing of Oriental Languages. DOI: https://doi.org/10.1007/11940098_15

Zowghi, D., & Coulin, C. (2005). Requirements elicitation: A survey of techniques, approaches, and tools. *Engineering and managing software requirements* (pp. 19-46): Springer. DOI: https://doi.org/10.1007/3-540-28244-0_2

Appendices

A. List of shortlisted studies on requirements prioritisation

ID	Authors	Title	Type	Year
1	Peng Shao	Sample Selection: An Algorithm for Requirements Prioritization	Conference: Proceedings of the 46th Annual Southeast Regional Conference on XX	2008
2	Jane Cleland-Huang, Bamshad Mobasher	Using data mining and recommender systems to scale up the requirements process	Workshop: Proceedings of the 2nd international workshop on Ultra-large-scale software-intensive systems	2008
3	Zornitza Racheva and Maya Daneva and Andrea Herrmann	A Conceptual Model of Client-driven Agile Requirements Prioritization: Results of a Case Study	Symposium: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement	2010
4	Francis Palma and Angelo Susi and Paolo Tonella	Using an SMT Solver for Interactive Requirements Prioritization	Conference: Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering	2011
5	Gerald Ninaus	Using Group Recommendation Heuristics for the Prioritization of Requirements	Conference: Proceedings of the sixth ACM conference on Recommender systems	2012
6	A. Felfernig and G. Ninaus	Group Recommendation Algorithms for Requirements Prioritization	Workshop: Proceedings of the Third International Workshop on Recommendation Systems for Software Engineering	2012
7	Abdel Ejnioui and Carlos Otero and Luis Otero	Simulation-based Fuzzy Multi-attribute Decision Making for Prioritizing Software Requirements	Conference: Proceedings of the 1st Annual conference on Research in information technology	2012
8	Stefan Gartner and Kurt Schneider	A Method for Prioritizing End-user Feedback for Requirements Engineering	Workshop: Proceedings of the 5th International Workshop on Co-operative and Human Aspects of Software Engineering	2012
9	Nupul Kukreja and Barry Boehm	Integrating Collaborative Requirements Negotiation and Prioritization Processes: A Match Made in Heaven	Conference: Proceedings of the 2013 International Conference on Software and System Process	2013
10	Persis Voola and A. Vinaya Babu	Comparison of Requirements Prioritization Techniques Employing Different	Journal: ACM SIGSOFT Software Engineering Notes	2013

ID	Authors	Title	Type	Year
		Scales of Measurement		
11	Punam Bajaj and Vineet Arora	Multi-person Decision-making for Requirements Prioritization Using Fuzzy AHP	Journal: ACM SIGSOFT Software Engineering Notes	2013
12	Richard R. Maiti and Frank J. Mitropoulos	Prioritizing Non-Functional Requirements in Agile Software Engineering	Conference: Proceedings of the SouthEast Conference	2017
13	J. Karlsson	Software requirements prioritizing	Conference: Proceedings of the Second International Conference on Requirements Engineering	1996
14	P. Berander	Using students as subjects in requirements prioritization	Symposium: 2004 International Symposium on Empirical Software Engineering	2004
15	P. Avesani; C. Bazzanella; A. Perini; A. Susi	Facing scalability issues in requirements prioritization with machine learning techniques	Conference: 13th IEEE International Conference on Requirements Engineering	2005
16	J. Cleland-Huang, M. Denne	Financially informed requirements prioritization	Conference: 27th International Conference on Software Engineering	2005
17	A. Perini; A. Susi; F. Ricca; C. Bazzanella Fondazione	An Empirical Study to Compare the Accuracy of AHP and CBRanking Techniques for Requirements Prioritization	Workshop: 2007 Fifth International Workshop on Comparative Evaluation in Requirements Engineering	2007
18	P. Laurent; J. Cleland-Huang; C. Duan	Towards Automated Requirements Triage	Conference: 15th IEEE International Requirements Engineering Conference	2007
19	D. Port; A. Olkov; T. Menzies	Using Simulation to Investigate Requirements Prioritization Strategies	Conference: 2008 23rd IEEE/ACM International Conference on Automated Software Engineering	2008
20	R. Beg; Q. Abbas; R. P. Verma	Using Simulation to Investigate Requirements Prioritization Strategies	Conference : 2008 First International Conference on Emerging Trends in Engineering and Technology	2008
21	T. M. Fehlmann	New Lanchester Theory for Requirements Prioritization	Workshop: 2008 Second International Workshop on Software Product Management	2008
22	A. Herrmann; M. Daneva	Requirements Prioritization Based on Benefit and Cost Prediction: An	Conference : 2008 16th IEEE International Requirements Engineering Conference	2008

ID	Authors	Title	Type	Year
		Agenda for Future Research		
23	Xuemei Sun, Dakun Zhang, Xinming Duan, You Zhao	Prioritizing design requirement in fuzzy quality function deployment	Conference: 2008 IEEE International Conference on Service Operations and Logistics, and Informatics	2008
24	Zhaoling Li; Dongling Zhang; Qisheng Gao	A grey method of prioritizing engineering characteristics in QFD	Conference : 2009 Chinese Control and Decision Conference	2009
25	M. R. Beg; R. P. Verma; A. Joshi	Reduction in number of comparisons for requirement prioritization using B-Tree	Conference: 2009 IEEE International Advance Computing Conference	2009
26	M. Sadiq; S. Ghafir; M. Shahid	An Approach for Eliciting Software Requirements and its Prioritization Using Analytic Hierarchy Process	Conference: 2009 International Conference on Advances in Recent Technologies in Communication and Computing	2009
27	M. Ramzan; M. A. Jaffar; M. A. Iqbal; S. Anwar; A. A. Shahid	Value Based Fuzzy Requirement Prioritization and Its Evaluation Framework	Conference: 2009 Fourth International Conference on Innovative Computing, Information and Control	2009
28	Amir Seyed Danesh, Soolmaz Mir Mortazavi, Seyed Yahya Seyed Danesh	Requirements Prioritization in On-line Banking Systems: Using Value-Oriented Framework	Conference: 2009 International Conference on Computer Technology and Development	2009
29	Aaron K. Massey, Paul N. Otto, Annie I. Antón	Prioritizing Legal Requirements	Workshop: 2009 Second International Workshop on Requirements Engineering and Law	2009
30	P. Fitsilis; V. Gerogiannis; L. Anthopoulos; I. K. Savvas	Supporting the Requirements Prioritization Process Using Social Network Analysis Techniques	Workshop: 2010 19th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises	2010
31	S. A. Marjaie; V. Kulkarni	Recognition of Hidden Factors in Requirements Prioritization Using Factor Analysis	Conference: 2010 International Conference on Computational Intelligence and Software Engineering	2010
32	C. E. Otero; E. Dell; A. Qureshi; L. D. Otero	A Quality-Based Requirement Prioritization Framework Using Binary Inputs	Conference: 2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation	2010
33	M. Sadiq; J. Ahmed; M. Asim; A. Qureshi; R. Suman	More on Elicitation of Software Requirements and	Conference: 2010 International Conference on Data Storage and Data Engineering	2010

ID	Authors	Title	Type	Year
		Prioritization Using AHP		
34	P. Chatzipetrou; L. Angelis; P. Rovegard; C. Wohlin	Prioritization of Issues and Requirements by Cumulative Voting: A Compositional Data Analysis Framework	Conference : 2010 36th EUROMICRO Conference on Software Engineering and Advanced Applications	2010
35	N. M. Carod	Cognitive-driven requirements prioritization: A case study	Conference: 2010 9th IEEE International Conference on Cognitive Informatics (ICCI)	2010
36	N. M. Carod	Cognitive Profiles in Understanding and Prioritizing Requirements: A Case Study	Conference: 2010 Fifth International Conference on Software Engineering Advances	2010
37	Z. Racheva; M. Daneva; A. Herrmann; R. J. Wieringa	A conceptual model and process for client-driven agile requirements prioritization	Conference: 2010 Fourth International Conference on Research Challenges in Information Science	2010
38	Z. Racheva; M. Daneva; K. Sikkel; A. Herrmann; R. Wieringa	Do We Know Enough about Requirements Prioritization in Agile Projects: Insights from a Case Study	Conference : 2010 18th IEEE International Requirements Engineering Conference	2010
39	M. Aasem; M. Ramzan; A. Jaffar	Analysis and optimization of software requirements prioritization techniques	Conference: 2010 International Conference on Information and Emerging Technologies	2010
40	M. A. Iqbal; A. M. Zaidi; S. Murtaza	A New Requirement Prioritization Model for Market Driven Products Using Analytical Hierarchical Process	Conference: 2010 International Conference on Data Storage and Data Engineering	2010
41	P. Tonella; A. Susi; F. Palma	Using Interactive GA for Requirements Prioritization	Symposium : 2nd International Symposium on Search Based Software Engineering	2010
42	A. Ahmad; A. Shahzad; V. K. Padmanabhuni; A. Mansoor; S. Joseph; Z. Arshad	Requirements prioritization with respect to Geographically Distributed Stakeholders	Conference: 2011 IEEE International Conference on Computer Science and Automation Engineering	2011
43	R. B. Svensson; T. Gorschek; B. Regnell; R. Torkar; A. Shahrokni; R. Feldt; A. Aurum	Prioritization of quality requirements: State of practice in eleven companies	Conference: 2011 IEEE 19th International Requirements Engineering Conference	2011

ID	Authors	Title	Type	Year
44	M. I. Babar; M. Ramzan; S. A. K. Ghayyur	Challenges and future trends in software requirements prioritization	Conference: International Conference on Computer Networks and Information Technology	2011
45	A. Ejnioui; C. E. Otero; A. A. Qureshi	Software requirement prioritization using fuzzy multi-attribute decision making	Conference: 2012 IEEE Conference on Open Systems	2012
46	N. Kukreja; B. Boehm; S. S. Payyavula; S. Padmanabhuni	Selecting an appropriate framework for value-based requirements prioritization	Conference : 2012 20th IEEE International Requirements Engineering Conference	2012
47	S. Nidhra; L. P. Kelapanda Satish; V. S. Ethiraj	Analytical Hierarchy Process issues and mitigation strategy for large number of requirements	Conference: 2012 CSI Sixth International Conference on Software Engineering	2012
48	S. Forouzani; R. Ahmad; S. Forouzani; N. Gazerani	Design of a teaching framework for software requirement prioritization	Conference: 2012 8th International Conference on Computing Technology and Information Management	2012
49	Hans Christian Benestad, Jo E. Hannay	Does the prioritization technique affect stakeholders' selection of essential software product features?	Symposium: Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement	2012
50	M. Pergher; B. Rossi	Requirements prioritization in software engineering: A systematic mapping study	Workshop : 2013 3rd International Workshop on Empirical Requirements Engineering	2013
51	M. W. Asghar; A. Marchetto; A. Susi; G. Scanniello	Maintainability-Based Requirements Prioritization by Using Artifacts Traceability and Code Metrics	Conference: 2013 17th European Conference on Software Maintenance and Reengineering	2013
52	A. Perini; A. Susi; P. Avesani	A Machine Learning Approach to Software Requirements Prioritization	Journal: IEEE Transactions on Software Engineering	2013
53	Nupul Kukreja	Decision theoretic requirements prioritization A two-step approach for sliding towards value realization	Conference: 2013 35th International Conference on Software Engineering (ICSE)	2013
54	R. Ahmed; D. Musleh; M. Ahmed; M. El-Attar	Use case prioritization using fuzzy logic system	Conference: 2014 IEEE 5th International Conference on	2014

ID	Authors	Title	Type	Year
			Software Engineering and Service Science	
55	Y. Z. Chen; Q. Yu	A fuzzy game approach to prioritize customer requirements in Quality Function Deployment	Conference : 2014 International Conference on Management Science & Engineering 21th Annual Conference Proceedings	2014
56	F. Fellir; K. Nafil; R. Touahni	System requirements prioritization based on AHP	Conference: 2014 Third IEEE International Colloquium in Information Science and Technology (CIST)	2014
57	R. Popli; N. Chauhan; H. Sharma	Prioritising user stories in agile environment	Conference: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT)	2014
58	A. Sureka	Requirements Prioritization and Next-Release Problem under Non-additive Value Conditions	Conference: 2014 23rd Australian Software Engineering Conference	2014
59	M. Inoki; T. Kitagawa; S. Honiden	Application of requirements prioritization decision rules in software product line evolution	Workshop: 2014 IEEE 5th International Workshop on Requirements Prioritization and Communication	2014
60	B. A. Mustafa; A. Zainuddin	An experimental design to compare software requirements prioritization techniques	Conference : 2014 International Conference on Computational Science and Technology	2014
61	R. Easmin; A. U. Gias; S. M. Khaled	A partial order assimilation approach for software requirements prioritization	Conference : 2014 International Conference on Informatics, Electronics & Vision	2014
62	F. Sher; D. N. A. Jawawi; R. Mohamad; M. I. Babar	Multi-aspects based requirements priortization technique for value-based software developments	Conference: 2014 International Conference on Emerging Technologies	2014
63	F. Sher; D. N. A. Jawawi; R. Mohamad; M. I. Babar	Requirements prioritization techniques and different aspects for prioritization a systematic literature review protocol	Conference : 2014 8th. Malaysian Software Engineering Conference	2014
64	N. Condori-Fernandez; P. Lago	Can we know upfront how to prioritize	Workshop: 2015 IEEE Fifth International Workshop on	2015

ID	Authors	Title	Type	Year
		quality requirements?	Empirical Requirements Engineering	
65	J. M. Fernandes; S. P. Rodrigues; L. A. Costa	Comparing AHP and ELECTRE I for prioritizing software requirements	Conference : 2015 IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing	2015
66	N. Garg; P. Agarwal; S. Khan	Recent advancements in requirement elicitation and prioritization techniques	Conference : 2015 International Conference on Advances in Computer Engineering and Applications	2015
67	R. R. Maiti; F. J. Mitropoulos	Capturing, eliciting, predicting and prioritizing (CEPP) non-functional requirements metadata during the early stages of agile software development	Conference: SoutheastCon 2015	2015
68	M. A. Abou-Elseoud; E. S. Nasr; H. A. Hefny	Enhancing requirements prioritization based on a hybrid technique	Conference: 2016 11th International Conference on Computer Engineering & Systems	2016
69	J. R. F. D. Santos; A. B. Albuquerque; P. R. Pinheiro	Requirements Prioritization in Market-Driven Software: A Survey Based on Large Numbers of Stakeholders and Requirements	Conference: 2016 10th International Conference on the Quality of Information and Communications Technology	2016
70	M. Yousuf; M. U. Bokhari; M. Zeyauddin	An analysis of software requirements prioritization techniques: A detailed survey	Conference : 2016 3rd International Conference on Computing for Sustainable Global Development	2016
71	R. M. Liaqat; M. A. Ahmed; F. Azam; B. Mehboob	A Majority Voting Goal Based technique for Requirement Prioritization	Conference : 2016 22nd International Conference on Automation and Computing	2016
72	S. Dhingra; Savithri G; M. Madan; Manjula R	Selection of prioritization technique for software requirement using Fuzzy Logic and Decision Tree	Conference: 2016 Online International Conference on Green Engineering and Technologies	2016
73	M. Sadiq; T. Hassan; S. Nazneen	AHP_GORE_PSR: Applying analytic hierarchy process in goal oriented requirements	Conference: 2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)	2017

ID	Authors	Title	Type	Year
		elicitation method for the prioritization of software requirements		
74	Morales-Ramirez; D. MuÃ±ante; F. Kifetew; A. Perini; A. Susi; A. Siena	Exploiting User Feedback in Tool-Supported Multi-criteria Requirements Prioritization	Conference: 2017 IEEE 25th International Requirements Engineering Conference	2017
75	B. B. Jawale; G. K. Patnaik; A. T. Bhole	Requirement Prioritization Using Adaptive Fuzzy Hierarchical Cumulative Voting	Conference: 2017 IEEE 7th International Advance Computing Conference	2017
76	U. Garg; A. Singhal	Software requirement prioritization based on non-functional requirements	Conference : 2017 7th International Conference on Cloud Computing, Data Science & Engineering	2017
77	A. R. Asghar; A. Tabassum; S. N. Bhatti; A. M. Jadi	Impact and challenges of requirements elicitation & prioritization in quality to agile process: Scrum as a case scenario	Conference: 2017 International Conference on Communication Technologies	2017
78	Emitza Guzman, Mohamed Ibrahim, Martin Glinz	Prioritizing User Feedback from Twitter: A Survey Report	Workshop : 2017 IEEE/ACM 4th International Workshop on CrowdSourcing in Software Engineering (CSI-SE)	2017
79	Perini, Anna; Ricca, Filippo; Susi, Angelo	Tool-supported requirements prioritization: Comparing the AHP and CBRank methods	Journal: Information and Software Technology	2009
80	Berander, Patrik; Svahnberg, Mikael	Evaluating two ways of calculating priorities in requirements hierarchies – An experiment on hierarchical cumulative voting	Journal: Journal of Systems and Software	2009
81	Bimal Nepal, Om P.Yadav, Alper Muratc	A fuzzy-AHP approach to prioritization of CS attributes in target planning for automotive product development	Journal: Expert Systems with Applications	2010
82	Riņķevičs, K.; Torkar, R.	Equality in cumulative voting: A systematic review with an improvement proposal	Journal: Information and Software Technology	2013

ID	Authors	Title	Type	Year
83	Tonella, Paolo; Susi, Angelo; Palma, Francis	Interactive requirements prioritization using a genetic algorithm	Journal: Information and Software Technology	2013
84	AL-Ta'ani, Rami Hasan; Razali, Rozilawati	Prioritizing Requirements in Agile Development: A Conceptual Framework	Conference: 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013	2013
85	Kukreja, Nupul; Payyavula, Sheetal Swaroop; Boehm, Barry; Padmanabhuni, Srivinas	Value-Based Requirements Prioritization: Usage Experiences	Conference: 2013 Conference on Systems Engineering Research	2013
86	Liu, Yuanyuan; Zhou, Jian; Chen, Yizeng	Using fuzzy non-linear regression to identify the degree of compensation among customer requirements in QFD	Journal : Neurocomputing	2014
87	Konstantina Kamvysi, Katerina Gotzamani, Andreas Andronikidis, Andreas C.Georgiou	Capturing and prioritizing students' requirements for course design by embedding Fuzzy-AHP and linear programming in QFD	Journal:European Journal of Operational Research	2014
88	Pitangueira, Antônio Mauricio; Maciel, Rita Suzana P.; Barros, Márcio	Software requirements selection and prioritization using SBSE approaches: A systematic review and mapping of the literature	Journal: Journal of Systems and Software	2015
89	Santos, Rômulo; Albuquerque, Adriano; Pinheiro, Plácido Rogerio	Towards the Applied Hybrid Model in Requirements Prioritization	Conference : Promoting Business Analytics and Quantitative Management of Technology: 4th International Conference on Information Technology and Quantitative Management	2016
90	Raj KumarChopra, Varun Gupta, Durg Singh Chauhan	Experimentation on accuracy of non functional requirement prioritization approaches for different complexity projects	Journal: Perspectives in Science	2016
91	Hosna Pakizehkar, Mohammad Mirmohammadi Sadrabadi, Rasool ZareMehrjardi,	The Application of Integration of Kano's Model, AHP Technique and QFD Matrix in Prioritizing	Conference: 3rd International Conference on New Challenges in Management and Business: Organization and Leadership	2016

ID	Authors	Title	Type	Year
	Amir Ehsan Eshaghieh	the Bank's Substructions		
92	Shao, Fei; Peng, Rong; Lai, Han; Wang, Bangchao	DRank: A semi-automated requirements prioritization method based on preferences and dependencies	Journal: Journal of Systems and Software	2017
93	Achimugu, Philip; Selamat, Ali; Ibrahim, Roliana; Mahrin, Mohd Naz'ri	A systematic literature review of software requirements prioritization research	Journal: Information and Software Technology	2017
94	Daneva, Maya; van der Veen, Egbert; Amrit, Chintan; Ghaisas, Smita; Sikkel, Klaas; Kumar, Ramesh; Ajmeri, Nirav; Ramteerthkar, Uday; Wieringa, Roel	Agile requirements prioritization in large-scale outsourced system projects: An empirical study	Journal:Journal of Systems and Software	2017
95	Anand, R. Vijay; Dinakaran, M.	andling stakeholder conflict by agile requirement prioritization using Apriori technique	Journal: Computers & Electrical Engineering	2017
96	Persis Voola, Vinaya Babu	Study of aggregation algorithms for aggregating imprecise software requirements' priorities	Journal:European Journal of Operational Research	2017
97	Wasserman, G.S.	On how to prioritize design requirements during the qfd planning process	Journal: IIE Transactions (Institute of Industrial Engineers)	1993
98	Armacost, R.L., Componation, P.J., Mullens, M.A., Swart, W.W.	An AHP framework for prioritizing customer requirements in QFD: An industrialized housing application	Journal:IIE Transactions (Institute of Industrial Engineers)	1994
99	Franceschini, F., Rossetto, S.	QFD: The problem of comparing technical/engineering design requirements	Journal: Research in Engineering Design	1995
100	Fung, Richard Y.K., Ren, Shouju, Xie, Jinxing	Prioritisation of attributes in customer requirement management	Conference:Proceedings of the IEEE International Conference on Systems, Man and Cybernetics	1996
101	Park, Taeho, Kim, Kwang-Jae	Integrative prioritization process in QFD with	Conference: Proceedings - Annual Meeting of the Decision Sciences Institute	1996

ID	Authors	Title	Type	Year
		modified house of quality		
102	Ryan, Kevin, Karlsson, Joachim	Prioritizing software requirements in an industrial setting	Conference: Proceedings - International Conference on Software Engineering	1997
103	Wang, H., Xie, M., Goh, T.N.	A comparative study of the prioritization matrix method and the analytic hierarchy process technique in quality function deployment	Journal: Total Quality Management	1998
104	Avesani, P., Bazzanella, C., Perini, A., Susi, A.	Exploiting domain knowledge in requirements prioritization	Conference: 17th International Conference on Software Engineering and Knowledge Engineering	2005
105	Karlsson, L., Host, M., Regnell, B.	Evaluating the practical use of different measurement scales in requirements prioritisation	Symposium:Proceedings of the 5th ACM-IEEE International Symposium on Empirical Software Engineering	2006
106	Lehtola, L., Kauppinen, M.	Suitability of requirements prioritization methods for market-driven software product development	Journal:Software Process Improvement and Practice	2006
107	Sharma, J.R., Pimplapure, S.R., Rawani, A.M.	Prioritizing customers requirements in QFD by integrating their interrelationship with the raw weights	Journal:Journal of the Institution of Engineers (India), Part PR: Production Engineering Division	2007
108	Karlsson, L., Thelin, T., Regnell, B., Berander, P., Wohlin, C.	Pair-wise comparisons versus planning game partitioning-experiments on requirements prioritisation techniques	Journal:Empirical Software Engineering	2007
109	Mohamed, A.S.I., El-Maddah, B.I.A., Wahba, C.A.M	Criteria-based requirements prioritization for software product management	Journal:Proceedings of the 2008 International Conference on Software Engineering Research and Practice	2008
110	Hoff, G., Fruhling, A., Ward, K.	Requirement prioritization decision factors for agile development environments	Conference: 14th Americas Conference on Information Systems	2008
111	Li, Y.-L., Tang, J.-F., Yao, J.-M., Luo, X.-G., Jiao, M.-H., Xu, J.	Integration methodology for prioritizing customer requirements in house of quality for product improvement	Journal:Computer Integrated Manufacturing Systems, CIMS	2008

ID	Authors	Title	Type	Year
112	Svahnberg, M., Karasira, A.	A study on the importance of order in requirements prioritisation	Workshop: 2009 3rd International Workshop on Software Product Management, IWSPM 2009	2009
113	Sharma, J.R., Rawani, A.M.	Linking company with customers and competitors: A comprehensive QFD model and its post-matrix analysis	Journal: International Journal of Modelling and Simulation	2009
114	Danesh, A.S., Ahmad, R.	Study of prioritization techniques using students as subject	Conference: Proceedings - 2009 International Conference on Information Management and Engineering, ICIME 2009	2009
115	Berander, P., Svahnberg, M.	Evaluating two ways of calculating priorities in requirements hierarchies - An experiment on hierarchical cumulative voting	Journal:Journal of Systems and Software	2009
116	Eben, K.G.M., Daniilidis, C., Lindemann, U.	Interrelating and prioritising requirements on multiple hierarchy levels	Conference: 11th International Design Conference	2010
117	Agheri, E., Asadi, M., Gasevic, D., Soltani, S.	Stratified analytic hierarchy process: Prioritization and selection of software features	Journal: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	2010
118	Bebensee, T., Van De Weerd, I., Brinkkemper, S.	Binary priority list for prioritizing software requirements	Journal: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	2010
119	Cordeiro, A.G., Freitas, A.L.P.	Prioritizaion of requirements and evaluation of software quality according to the users' perspective	Journal: Ciencia da Informacao	2011
120	Koziolek, A.	Architecture-driven quality requirements prioritization	Workshop: 2012 1st IEEE International Workshop on the Twin Peaks of Requirements and Architecture	2012
121	Nidhra, S., Kelapanda Satish, L.P., Ethiraj, V.S.	Analytical Hierarchy Process issues and mitigation strategy for large number of requirements - An experimental study	Conference: 2012 CSI 6th International Conference on Software Engineering	2012

ID	Authors	Title	Type	Year
122	Sharma, A.K., Sharma, J., Mehta, I.C.	A novel fuzzy integrated technical requirements prioritization software system for quality function deployment	Journal: International Journal of Computers and Applications	2012
123	Ejnioui, A., Otero, C.E., Otero, L.D.	A Simulation-based fuzzy multi-attribute decision making for prioritizing software requirements	Conference: Proceedings of the ACM Research in Information Technology	2012
124	Reichel, T., Rünger, G.	Prioritization of product requirements using the analytic hierarchy process	Conference:Proceedings of the 14th International Conference on Enterprise Information Systems	2012
125	Ma, Y., Liu, Y., Zhang, H., Chen, G.	A hybrid method for prioritizing software requirements in terms of use cases	Journal: Journal of Convergence Information Technology	2012
126	Koziolek, A	Research preview: Prioritizing quality requirements based on software architecture evaluation feedback	Conference: 18th Working Conference on Requirements Engineering: Foundation for Software Quality	2012
127	Peng, W., Sun, T., Revankar, S., Li, T.	Mining the "voice of the customer" for business prioritization	Journal:ACM Transactions on Intelligent Systems and Technology	2012
128	Kassab, M.	An integrated approach of AHP and NFRs framework	Conference: Proceedings - International Conference on Research Challenges in Information Science	2013
129	Ejnioui, A., Otero, C.E., Otero, L.D.	Prioritisation of software requirements using grey relational analysis	Journal: International Journal of Computer Applications in Technology	2013
130	Sadiq, M., Jain, S.K.	A fuzzy based approach for requirements prioritization in goal oriented requirements elicitation process	Conference: Proceedings of the International Conference on Software Engineering and Knowledge Engineering	2013
131	Achimugu, P., Selamat, A., Azar, A.T., Vaidyanathan, S.	A hybridized approach for prioritizing software requirements based on k-means and evolutionary algorithms	Journal:Studies in Computational Intelligence	2014
132	Kamvysi, K., Gotzamani, K., Andronikidis, A., Georgiou, A.C.	Capturing and prioritizing students' requirements for course design by embedding Fuzzy-	Journal: European Journal of Operational Research	2014

ID	Authors	Title	Type	Year
		AHP and linear programming in QFD		
133	Achimugu, P., Selamat, A., Ibrahim, R	A clustering based technique for large scale prioritization during requirements elicitation	Journal:Advances in Intelligent Systems and Computing	2014
134	Achimugu, P., Selamat, A., Ibrahim, R.	Applying fuzzy-TOPSIS algorithm in prioritizing software requirements	Chapter:Frontiers in Artificial Intelligence and Applications	2014
135	Singh, D.E, Sharma, A.	Software requirement prioritization using machine learning	Conference:Proceedings of the International Conference on Software Engineering and Knowledge Engineering	2014
136	Achimugu, P., Selamat, A., Ibrahim, R	A preference weights model for prioritizing software requirements	Journal:Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	2014
137	Parthasarathy, S., Daneva, M.	Customer requirements based ERP customization using AHP technique	Journal:Business Process Management	2014
138	Babar, M.I., Ghazali, M., Jawawi, D.N.A., Shamsuddin, S.M., Ibrahim, N.	PHandler: An expert system for a scalable software requirements prioritization process	Journal: Knowledge-Based Systems	2015
139	Franceschini, F.Email Author, Galetto, M., Maisano, D., Mastrogiacomo, L.	Prioritisation of engineering characteristics in QFD in the case of customer requirements orderings	Journal:International Journal of Production Research	2015
140	Valsala, S., Nair, A.R.	Requirement prioritization in software release planning using enriched genetic algortihm	Journal:International Journal of Applied Engineering Research	2015
141	Jakub Duda, Maciej Rostański, Wojciech Borczyk, Krzysztof Grochla	Applying Kano model into goal/requirements elicitation for crossplatform mobile content technology	Conference:Proceedings of the 11th International Conference on Strategic Management and Its Support by Information Systems 2015, SMSIS 2015	2015
142	Devulapalli, S., Rao, O., Khare, A.	Requirements prioritization-parameters of relevance-An empirical study across 3 datasets	Conference:ACM International Conference Proceeding Series	2016

ID	Authors	Title	Type	Year
143	Achimugu, P., Selamat, A., Ibrahim, R.	ReproTizer: A fully implemented software requirements prioritization tool	Journal: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)	2016
144	Hujainah, F., Abu Bakar, R.B., Al-Haimi, B., Nasser, A.B.	Analyzing requirement prioritization techniques based on the used aspects	Journal: Research Journal of Applied Sciences	2016
145	Al-Ta'ani, R.H., Razali, R.	A framework for requirements prioritisation process in an agile software development environment: Empirical study	Journal: International Journal on Advanced Science, Engineering and Information Technology	2016
146	Devulapalli, S., Rao, O.R.S., Khare,	Mathematical treatment of ABC framework for requirements prioritization	Conference: Smart Innovation, Systems and Technologies	2016
147	Seyed Fadzir, S.F., Ibrahim, S., Mahrin, M.N	Requirement prioritization approaches and evaluation strategies: A systematic literature review	Journal: Journal of Engineering and Applied Sciences	2016
148	Thakurta, R.	A specification of principles governing the design of requirement prioritisation approaches	Journal: International Journal of Business Information Systems	2016
149	Viswanathan, A., Nair, S.R., Krishnan, S.M.	Solution model for requirement prioritization	Journal: International Journal of Control Theory and Applications	2016
150	Keertipati, S., Savarimuthu, B.T.R., Licorish, S.A.	Approaches for prioritizing feature improvements extracted from app reviews	Conference:20th International Conference on Evaluation and Assessment in Software Engineering	2016
151	Sadiq, M.	A Fuzzy Set-Based Approach for the Prioritization of Stakeholders on the Basis of the Importance of Software Requirements	Journal: IETE Journal of Research	2017
152	Dhir, S., Kumar, D., Singh, V.B.	Requirement paradigms to implement the software projects in agile development using analytical hierarchy process	Journal: International Journal of Decision Support System Technology	2017

ID	Authors	Title	Type	Year
153	Anand, R.V., Dinakaran, M.	Multi-voting and binary search tree-based requirements prioritisation for e-service software project development	Journal: Electronic Government	2017
154	Barbosa, P.A.M., De Vasconcelos Silveira, F.R., Pinheiro, P.R., Filho, M.S.	Selection and prioritization of software requirements using the Verbal Decision Analysis paradigm	Journal: Proceedings of the International Conference on Software Engineering and Knowledge Engineering	2017
155	Garg, N., Sadiq, M., Agarwal, P.	GOASREP: Goal oriented approach for software requirements elicitation and prioritization using analytic hierarchy process	Chapter: Advances in Intelligent Systems and Computing	2017
156	Nazir, F., Butt, W.H., Anwar, M.W., Khan Khattak, M.A.	The applications of natural language processing (NLP) for software requirement engineering - A systematic literature review	Chapter: Lecture Notes in Electrical Engineering	2017
157	Khan, S.U.R., Lee, S.P., Dabbagh, M., Tahir, M., Khan, M., Arif, M.	RePizer: a framework for prioritization of software requirements	Journal: Frontiers of Information Technology and Electronic Engineering	2017
158	Busetta, P., Kifetew, F.M., Munante, D., Perini, A., Siena, A., Susi, A.	Tool-Supported Collaborative Requirements Prioritisation	Conference: Proceedings - International Computer Software and Applications	2017
159	Sadiq, M., Hassan, T., Nazneen, S	AHP-GORE-PSR: Applying analytic hierarchy process in goal oriented requirements elicitation method for the prioritization of software requirements	Conference: 3rd IEEE International Conference on "Computational Intelligence and Communication Technology"	2017
160	Licorish, S.A., Savarimuthu, B.T.R., Keertipati, S.	Attributes that predict which features to fix: Lessons for app store mining	Conference: 21st International Conference on Evaluation and Assessment in Software Engineering	2017
161	Vinodh, S, Manjunatheshwara, K.J., Karthik Sundaram, , Kirthivasan, V	Application of fuzzy quality function deployment for sustainable design of consumer electronics products: a case study	Journal: Clean Technologies and Environmental Policy	2017

ID	Authors	Title	Type	Year
162	Kifetew, F.M., Susi, A., Muñante, D., Perini, A., Siena, A., Busetta, P.	Towards multi-decision-maker requirements prioritisation via multi-objective optimisation	Workshop: CEUR Workshop Proceedings	2017
163	Seyff, N., Stade, M., Fotrousi, F., Glinz, M., Guzman, E., Kolpondinos-Huber, M., Arzapalo, D.M., Oriol, M.f, Schaniel, R.	End-user driven feedback prioritization	Workshop: CEUR Workshop Proceedings	2017
164	Laura Lehtola, Marjo Kauppinen, Sari Kujala	Requirements Prioritization Challenges in Practice	Conference: International Conference on Product Focused Software Process Improvement	2004
165	Laura Lehtola, Marjo Kauppinen	Empirical Evaluation of Two Requirements Prioritization Methods in Product Development Projects	Conference: European Conference on Software Process Improvement	2004
166	Patrik Berander, Anneliese Andrews	Requirements Prioritization	Chapter:Engineering and Managing Software Requirements	2005
167	Daniel Port, Tung Bui	Simulating mixed agile and plan-based requirements prioritization strategies: proof-of-concept and practical implications	Journal: European Journal of Information Systems	2009
168	Chuan Duan, Paula Laurent,Jane Cleland-Huang, Charles Kwiatkowski	Towards automated requirements prioritization and triage	Journal:Requirements Engineering	2009
169	Gaur Vibha, Soni Anuja	Identifying an Appropriate Requirements Prioritization Methodology Using Fuzzy Decision-Making	Chapter: Computer Networks and Intelligent Computing	2011
170	Dayvison Chaves Lima, FabrÃcio Freitas, Gutavo Campos, Jerffeson Souza	A Fuzzy Approach to Requirements Prioritization	Symposium: International Symposium on Search Based Software Engineering	2011
171	Zornitza Bakalova, Maya Daneva, Andrea Herrmann, Roel Wieringa	Agile Requirements Prioritization: What Happens in Practice and What Is Described in Literature	Conference: International Working Conference on Requirements Engineering: Foundation for Software Quality	2011

ID	Authors	Title	Type	Year
172	Persis Voola, A. Vinaya Babu	Interval Evidential Reasoning Algorithm for Requirements Prioritization	Conference: Proceedings of the International Conference on Information Systems Design and Intelligent Applications	2012
173	Philip Achimugu, Ali Selamat, Roliana Ibrahim	A Web-Based Multi-Criteria Decision Making Tool for Software Requirements Prioritization	Conference: International Conference on Computational Collective Intelligence	2014
174	Philip Achimugu, Ali Selamat, Roliana Ibrahim, Mohd Nazâri Mahrin	An Adaptive Fuzzy Decision Matrix Model for Software Requirements Prioritization	Chapter: Advanced Approaches to Intelligent Information and Database Systems	2014
175	Zhixiang Tong, Qiankun Zhuang,Qi Guo, Peijun	Research on Technologies of Software Requirements Prioritization	Conference: International Conference on Trustworthy Computing and Services	2014
176	Mohd Sadiq,S. K. Jain	Applying fuzzy preference relation for requirements prioritization in goal oriented requirements elicitation process	Journal: International Journal of System Assurance Engineering and Management	2014
177	Panagiota Chatzipetrou, Christos Karapiperis, Chrysa Palampouiki, Lefteris Angelis	Statistical Analysis of Requirements Prioritization for Transition to Web Technologies: A Case Study in an Electric Power Organization	Conference: International Conference on Software Quality	2014
178	Jenjira Jaimunk, Pradorn Sureephong	A Comparison Approach for Accuracy Feature of Requirements Prioritization Models	Chapter: Industrial Engineering, Management Science and Applications	2015
179	Norman Riegel,Joerg Doerr	A Systematic Literature Review of Requirements Prioritization Criteria	Conference:International Working Conference on Requirements Engineering: Foundation for Software Quality	2015
180	Mohamad Kassab, Nil Kilicay-Ergin	Applying analytical hierarchy process to system quality requirements prioritization	Journal: Innovations in Systems and Software Engineering	2015
181	McZara, J., Sarkani, S., Holzer	Software requirements prioritization and selection using linguistic tools and constraint solvers—a controlled experiment	Journal: Empirical Software Engineering	2015

ID	Authors	Title	Type	Year
182	Sita Devulapalli, Akhil Khare, O. R. S. Rao	Requirements Prioritization: Survey and Analysis	Conference: Proceedings of the International Congress on Information and Communication Technology	2016
183	Mohammad Dabbagh, Sai Peck Lee, Reza Meimandi Parizi	Functional and non-functional requirements prioritization: empirical evaluation of IPA, AHP-based, and HAM-based approaches	Chapter: Soft Computing	2016
184	Misaghian, N. & Motameni, H.	An approach for requirements prioritization based on tensor decomposition	Journal: Requirements Engineering	2016
185	Sita Devulapalli, O R S Rao, Akhil Khare	Comparison of ABC Framework with AHP, Wiegers Method, Cost-Value, Priority Groups for Requirements Prioritization	Conference: Proceedings of International Conference on Communication and Networks	2017
186	Estefanía Serral, Paolo Sernani, Aldo Franco Dragoni, Fabiano Dalpiaz	Contextual Requirements Prioritization and Its Application to Smart Homes	Journal: European Conference on Ambient Intelligence	2017
187	Luay Alawneh	Requirements Prioritization Using Hierarchical Dependencies	Chapter: Information Technology - New Generations	2017
188	erander, Patrik; Jonsson, Per	Hierarchical Cumulative Voting (HCV) prioritization of requirements in hierarchies	Journal :INTERNATIONAL JOURNAL OF SOFTWARE ENGINEERING AND KNOWLEDGE ENGINEERING	2006
189	Azar, J, Smith, RK, Cordes, D	Value-oriented requirements prioritization in a small development organization	Journal:IEEE software	2007
190	Daneva, Maya; Herrmann, Andrea	Requirements Prioritization Based on Benefit and Cost Prediction: A Method Classification Framework	Conference: PROCEEDINGS OF THE 34TH EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS	2008
191	Ilie, D ; Lindemann, U ; Kain, A	EVALUATION AND PRIORITIZATION OF CROSS-LINKED REQUIREMENTS	Conference: ASME INTERNATIONAL DESIGN ENGINEERING TECHNICAL CONFERENCES AND COMPUTERS AND	2009

ID	Authors	Title	Type	Year
		IN THE AUTOMOTIVE DEVELOPMENT PROCESS	Conference: INFORMATION IN ENGINEERING	
192	Ramzan, Muhammad; Jaffar, M. Arfan; Shahid, Arshad Ali	VALUE BASED INTELLIGENT REQUIREMENT PRIORITIZATION (VIRP): EXPERT DRIVEN FUZZY LOGIC BASED PRIORITIZATION TECHNIQUE	Journal: INTERNATIONAL JOURNAL OF INNOVATIVE COMPUTING INFORMATION AND CONTROL	2011
193	Payyavula, SS, Jahagirdar, SS, Kumar, M	Application of Value Based Requirement Prioritization in a Banking Product implementation	Conference: 3rd International Conference on Services in Emerging Markets	2012
194	Azzolini, Martin; Isabel Passoni, Lucia	Prioritization of Software Requirements: a Cognitive Approach	Workshop:4th International Workshop on Knowledge Discovery, Knowledge Management and Decision Support	2013
195	Thakurta, R	A framework for prioritization of quality requirements for inclusion in a software project.	Journal: SOFTWARE QUALITY JOURNAL	2013
196	Dabbagh, Mohammad; Lee, Sai Peck	An Approach for Integrating the Prioritization of Functional and Nonfunctional Requirements	Journal: SCIENTIFIC WORLD JOURNAL	2014
197	Elsood, Mukhtar A. Abo; Hefny, Hesham A.; Nasr, Eman S.	A Goal-Based Technique for Requirements Prioritization	Conference: 9th International Conference on Informatics and Systems	2014
198	Fadzir, Syarifah Fazlin Seyed; Ibrahim, Suhaimi; Mahrin, Mohd Naz'ri	Requirement prioritization approaches and evaluation strategies: A systematic literature review	Conference: INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS	2015
199	McZara, Jason; Sarkani, Shahryar; Holzer, Thomas; Eveleigh, Timothy	Software requirements prioritization and selection using linguistic tools and constraint solvers-a controlled experiment	Journal : EMPIRICAL SOFTWARE ENGINEERING	2015
200	Franceschini, Fiorenzo; Maisano, Domenico	Prioritization of QFD Customer Requirements Based on the Law of Comparative Judgments	Journal:QUALITY ENGINEERING	2015

ID	Authors	Title	Type	Year
201	Franceschini, Fiorenzo; Maisano, Domenico; Mastrogiacomo, Luca	Customer requirement prioritization on QFD: a new proposal based on the generalized Yager's algorithm	Journal: RESEARCH IN ENGINEERING DESIGN	2015
202	Achimugu, Philip; Selamat, Ali; Ibrahim, Roliana	USING THE FUZZY MULTI-CRITERIA DECISION MAKING APPROACH FOR SOFTWARE REQUIREMENTS PRIORITIZATION	Journal: JURNAL TEKNOLOGI	2015
203	Salado, A, Nilchiani, R	Adaptive Requirements Prioritization (ARP): Improving Decisions between Conflicting Requirements	Journal:SYSTEMS ENGINEERING	2015
204	Choochotkaew, S, Yamaguchi, H, Higashino, T, Shibuya, M	Requirement-Based Prioritization system in Multi-user IoT	World Forum:2015 IEEE 2nd World Forum on Internet of Things	2015
205	Asghar, Aneesa Rida; Bhatti, Shahid Nazir; Tabassum, Atika; Sultan, Zainab; Abbas, Rabiya	Role of Requirements Elicitation & Prioritization to Optimize Quality in Scrum Agile Development	Journal: INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS	2016
206	J Khan, JA; Izaz-ur-Rehman; Khan, SP; Afzal, W; Qasim, I; Khan, YH	An Evaluation of Requirement Prioritization Techniques with ANP	Journal: INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND APPLICATIONS	2016
207	J dos Santos, JRF; Albuquerque, AB; Pinheiro, PR	Requirements Prioritization in Market-Driven Software A survey based on large numbers of stakeholders and requirements	Conference: PROCEEDINGS 2016 10TH INTERNATIONAL CONFERENCE ON THE QUALITY OF INFORMATION AND COMMUNICATIONS TECHNOLOGY	2016
208	Abou-Elseoud, MA; Nasr, ES; Hefny, HA	Enhancing Requirements Prioritization Based on Hybrid Technique	Conference : PROCEEDINGS OF 2016 11TH INTERNATIONAL CONFERENCE ON COMPUTER ENGINEERING & SYSTEMS	2016
209	Atukorala, Nimanthi L.; Chang, Carl K.; Oyama, Katsunori	Situation-Oriented Evaluation and Prioritization of Requirements	Symposium : 3rd Asia-Pacific Requirements Engineering Symposium	2016
210	Ahuja, H; Sujata; Purohit, GN	Understanding Requirement	Conference : IEEE International Conference on	2016

ID	Authors	Title	Type	Year
		Prioritization Techniques	Computing, Communication and Automation	
211	Kravchenko, TK and Sergey B	Prioritization of requirements for effective support of the communication process with customers of a commercial bank	Journal:BIZNES INFORMATIKA-BUSINESS INFORMATICS	2017

B. Web crawler to extract app reviews from Google Play Store

```
'''
Python Script to extract reviews of an app hosted on Google Play Store
'''

#load webdriver function from selenium
from selenium import webdriver
import bs4
import pandas as pd
from selenium.webdriver.common.keys import Keys
from webdriver_manager.chrome import ChromeDriverManager
import time

x = 100


link="https://play.google.com/store/apps/details?id=nz.co.zenergy.loyaltycard.android&showAllRevi
ews=true"

driver = webdriver.Chrome(ChromeDriverManager().install())
driver.get(link + '&showAllReviews=true')

num_clicks = 0
num_scrolls = 0

while num_clicks <= x and num_scrolls <= x*10:
   try:
      show_more=driver.find_element_by_xpath('//*[@id="fcxH9b"]/div[4]/c-
wiz/div/div[2]/div/div[1]/div/div/div[1]/div[2]/div[2]/div/span/span')
      # Change accordingly if GooglePlay is updated
      show_more.click()
      num_clicks += 1

      print ("num_clicks =", num_clicks)
   except:
      html = driver.find_element_by_tag_name('html')
      html.send_keys(Keys.END)
      num_scrolls +=1
      time.sleep(3)
      print("num_scrolls =", num_scrolls)
```

```
## File mode
# with open('F:/final.html', encoding='utf-8') as source:
#     source_content = source.read()
#     try:
#         soup = bs4.BeautifulSoup(source_content, 'html.parser')
#         h2 = soup.find_all('h2')
#     except Exception as e:
#         print(e)
#         raise e

## Live mode
try:
    soup = bs4.BeautifulSoup(driver.page_source.encode('utf-8'), 'html.parser')
    h2 = soup.find_all('h2')
except Exception as e:
    print(e)
    raise e

results_df = pd.DataFrame()

blocks = soup.findAll('div', {'class':'zc7KVe'})
# print('blocks :', len(blocks))
for one_block in blocks: # Change accordingly if GooglePlay is updated
    name = one_block.find('span', {'class':'X43Kjb'})
    rate = one_block.find('div', {'class':'pf5lIe'})
    try:
        rate = len(rate.findAll('div', {'class':'vQHuPe'}))
    except AttributeError:
        rate = ''

    date = one_block.find('span', {'class':'p2TkOb'})
    review = one_block.find('div', {'class':'UD7Dzf'})
    try:
        temp_df = pd.DataFrame([[date.text, rate, name.text, review.text]], columns =
['Date','Rating','User','Review'])
        results_df = results_df.append(temp_df)
    except:
        continue

results_df = results_df.reset_index(drop=True)
results_df.to_csv('F:/ZappNZ.csv', index=False)

driver.close()
```

C. University of Otago Human Ethics Committee application

Reporting Sheet for use ONLY for proposals considered at departmental level

---

Form Updated: December 2017

UNIVERSITY OF OTAGO HUMAN ETHICS COMMITTEE APPLICATION FORM: CATEGORY B

(Departmental Approval)

Please ensure you are using the latest application form available from: http://www.otago.ac.nz/council/committees/committees/HumanEthicsCommittees.html

1. University of Otago staff member responsible for project:

Licorish Sherlock Title (Dr)

2. Department/School:

Department of Information Science

3. Contact details of staff member responsible *(always include your email address)*:

Sherlock Licorish
Information Science Department
University of Otago
Email: sherlock.licorish@otago.ac.nz
Phone: (03) 479 8319

4. Title of project:

Evaluating the outcome of an automated requirements prioritisation solution

5. Indicate type of project and names of other investigators and students:

Staff Research	☐	**Names**	Sherlock Licorish Bastin Tony Roy Savarimuthu
Student Research	X	**Names**	Saurabh Malgaonkar
Level of Study (e.g. PhD, Masters, Hons)			PhD
	☐		

External Research/ Collaboration **Names**

Institute/Company

6. When will recruitment and data collection commence?

3 February 2019

When will data collection be completed?

3 March 2019

7. Brief description in lay terms of the aim of the project, and outline of the research questions that will be answered (approx. 200 words)**:**

Identification of prioritised app requirements assists valuably in launching essential app updates with end-users solicited functionalities in the app market. Therefore, it is crucial that the priorities of the requirements match closely with end-users' preferences. The PhD study under consideration addressed this issue, by automating the requirements prioritisation process, which is now to be evaluated. The outcome reflects a set of prioritised reviews reflecting end-user requirements belonging to four apps. These prioritised reviews have been assigned priority labels High, Medium or Low by the solution.

During the evaluation process, participants will assign the appropriate priority labels (High-Medium-Low) to the same set of non-prioritised reviews based on a predefined criteria (see attached). Later, the accuracy of the outcome will be computed by comparing the priority labels assigned by the solution (i.e., actual outcome) against those assigned by the participants (i.e., expected outcome). The participants of this study are expected to have knowledge about various apps and their usage; thus, they eventually exhibit the role of apps' end-users in this study.

8. Brief description of the method. Include a description of who the participants are, how the participants will be recruited, and what they will be asked to do and how the data will be used and stored *(Note: if this research involves **patient data or health information** obtained from the Ministry of Health, DHBs etc please refer to the UOHEC(H) Minimal Risk Health Research - Audit and Audit related studies):-*

This research will employ a direct one on one interaction to elicit participant's evaluation outcome. The participants will comprise graduates from the Information Science department. We intend to solicit evaluation outcomes from a total of 10 participants.

After consenting to participate in the study, the participants will be briefed upon task that needs to be performed to evaluate the accuracy of the outcome generated by the solution as stated in the task description (refer to Task Description sheet). This briefing period is expected to be of approximately 5 minutes.

Upon the completion of the briefing, the participant will complete an evaluation task consisting of 29 reviews to assign the necessary priority labels to each review. This period is expected to last for 30 minutes.

9. Disclose and discuss any potential problems and how they will be managed: (For example: medical/legal problems, issues with disclosure, conflict of interest, safety of the researcher, etc)

1. Respondents will be assured of confidentiality and anonymity. Only the investigators mentioned in the application will view the data.
2. The response will only be reported in an aggregate form.
3. To ensure anonymity any personal information collected that identifies the respondents will be deleted at the end of the research.
4. An information sheet with a brief introduction to the research will be presented to respondents at the time of recruiting (see attached).

***Applicant's Signature:** .....

Name (please print): Sherlock Licorish

Date: 29/01/2019

**The signatory should be the staff member detailed at Question 1.*

ACTION TAKEN

☐ Approved by HOD ☐ Approved by Departmental Ethics Committee

☐ Referred to UO Human Ethics Committee

Signature of **Head of Department: .....

Name of HOD (please print): ……..

Date: .....

**Where the Head of Department is also the Applicant, then an appropriate senior staff member must sign on behalf of the Department or School.

Departmental approval: *I have read this application and believe it to be valid research and ethically sound. I approve the research design. The research proposed in this application is compatible with the University of Otago policies and I give my approval and consent for the application to be forwarded to the University of Otago Human Ethics Committee (to be reported to the next meeting).*

IMPORTANT NOTE: As soon as this proposal has been considered and approved at departmental level, the completed form, together with copies of any Information Sheet, Consent Form, recruitment advertisement for participants, and survey or questionnaire should be forwarded to the Manager, Academic Committees or the Academic Committees Administrator, Academic Committees, Rooms G22, or G26, Ground Floor,

Clocktower Building, or scanned and emailed to either gary.witte@otago.ac.nz. or jane.hinkley@otago.ac.nz

D. External evaluation details

Reporting Sheet for use ONLY for proposals considered at departmental level

---

EVALUATING THE OUTCOME OF AN AUTOMATED REQUIREMENTS PRIORITISATION SOLUTION

INFORMATION SHEET FOR PARTICIPANTS

Thank you for showing an interest in this project. Please read this information sheet carefully before deciding whether or not to participate. If you decide to participate we thank you. If you decide not to take part there will be no disadvantage to you and we thank you for considering our request.

What is the aim of the study?

This project is being undertaken as part of the requirements for Saurabh Malgaonkar's study for the award of Doctor of Philosophy (PhD). The aim of the study is to evaluate the accuracy of the outcomes that are generated by a developed automated requirements prioritisation solution. The outcome reflects a set of prioritised reviews reflecting end-user requirements belonging to four apps. The prioritised reviews have been assigned priority labels *High*, *Medium* or *Low* by the solution. The participant of this study will assign the appropriate priority labels (High-Medium-Low) to the same set of non-prioritised reviews based on predefined criteria. Later, the accuracy of the outcome will be computed by comparing the priority labels assigned by the solution (i.e., actual outcome) against those assigned by the participant (i.e., expected outcome).

Identification of prioritised app requirements assists valuably in launching essential app updates with end-users solicited functionalities in the app market. Hence, it is crucial that the priorities of the requirements match closely or completely with the requirements preferences of the end-users. Based on this, the participant of this study is exhibiting the role of an app's end-user.

What types of participants are being sought?

Postgraduate Information Science students or staff members who are familiar with various apps and have experience with apps usage. The participants (e.g., Masters or PhD students, faculty members) from Information Science will be recruited through the departmental administrators and staff members (course coordinators of various papers).

What will participants be asked to do?

Should a participant agree to take part in this study, they will be asked to attend a face-to-face interaction with the mediating researcher (Saurabh Malgaonkar) in person. Once the study begins, each participant will be briefed on the evaluation task mentioned in the Task Instruction

document. The participant will be provided with the Task Instruction document for reference purposes.

Each participant will then be asked to complete a single task which comprises of 29 reviews reflecting app requirements belonging to four apps. The participant will assign a priority label to each review. The estimated time for completing the study will be about 35 minutes for the participants with no additional constraint being imposed.

What data will be collected and what use will be made of it?

Only anonymous data will be collected from the participants. The data collected from participants would reflect the priorities of the apps' requirements from the end-user perspective. This data would act as the ground truth to evaluate the accuracy of the outcome generated by the requirements prioritisation solution. The manual prioritisation results from the data collected from each participant would be aggregated to determine the overall accuracy of the solution. The data collected will be securely stored in such a way that only the student and supervisors will be able to gain access to it. Data obtained as a result of the research will be retained for at least 5 years before deletion.

Can participants change their mind and withdraw from the project?

Participants may withdraw from participation in the project at any time and without any disadvantage to themselves.

What if participants have any questions?

If you have any questions about our project, either now or in the future, please feel free to contact:

Dr. Sherlock Licorish
Department of Information Science
University Telephone Number: (03) 479 8319
Email Address: sherlock.licorish@otago.ac.nz

This study has been approved by the Department stated above. However, if you have any concerns about the ethical conduct of the research you may contact the University of Otago Human Ethics Committee through the Human Ethics Committee Administrator (ph +643 479 8256 or email gary.witte@otago.ac.nz). Any issues you raise will be treated in confidence and investigated and you will be informed of the outcome.

EVALUATING THE OUTCOME OF AN AUTOMATED REQUIREMENTS PRIORITISATION SOLUTION

CONSENT FORM FOR PARTICIPANTS

I have read the Information Sheet concerning this project and understand what it is about. All my questions have been answered to my satisfaction. I understand that I am free to request further information at any stage.

I know that:-

1. My participation in the project is entirely voluntary;

2. I am free to withdraw from the project at any time without any disadvantage;

3. Any raw data on which the results of the project depend will be retained in secure storage for at least five years;

4. This project involves an outcome evaluation task. The general line of evaluation includes how the participant evaluates the outcome of the requirements prioritisation solution;

5. The results of the project may be published and will be available in the University of Otago Library (Dunedin, New Zealand) but every attempt will be made to preserve my anonymity.

I agree to take part in this project.

....
(Signature of participant) (Date)

.....
(Printed Name)

EVALUATING THE OUTCOME OF AN AUTOMATED REQUIREMENTS PRIORITISATION SOLUTION

TASK DESCRIPTION SHEET FOR PARTICIPANTS

As a regular apps end-user, your task is to assign priority labels to a set of 29 reviews belonging to four apps (anonymously). These apps are available on Google Play, and were installed by end-users on their android devices. Having used the apps, reviews were logged. As a participant of the project, you are to assign a priority label (low, medium or high) for these reviews based on the criteria below:

Priority label assignment guideline

Priority label	**Description**	**Example Reviews**
Low	Reviews that reflect issues, suggestions or requests pertaining to an app that seem **optional (not obligatory)** towards the functionalities or performance of the app.	"Would love it more if we can add other players and visit their sanctuary but still this game is very entertaining." "Like to have a save my favorites option." "Love the material design. Dark mode and Chromebook optimization would be awesome." "I like it when you can get free add-ons sometimes on your Vodafone app but make it regular please."
Medium	Reviews that reflect issues, suggestions or requests pertaining to an app that seem **mandatory (imperative)** towards the app's functionalities or performance.	"The first few times I turned it on the graphics were great but now the butterflies are just coloured squares along with the writing is messed up." "Can't even watch a full episode of anything, pauses most of the time or restarts the episode. No problem viewing the too many ads within the episodes though. fix this.." "It's okay, good for basic use but some options are not available on the app so sometimes I need to use the full website on a computer." "App isn't storing credit card info properly any more, cuts off last number so topups don't go through. Also keep getting "unable to retrieve balance" and sometimes shows inaccurate balance. Sort it out guys."
High	Reviews that reflect issues, suggestions or	"I lost my whole game I have enjoyed this game for 2 years and now it no longer works after the last update it erased my whole game like I

	requests pertaining to an app that seem **critical (severe)** towards the app's functionalities or performance.	never played and now it won't open." "The only streaming app on my Samsung note that won't work. Crashes frequently. Always gives 'unexpectedly stopped working' notice." "Clock error. All of a sudden it tells my clock is wrong. Tried reboot reinstall let network decide my time etc. but still cannot get the app to work." "I can't seem to download the app due to "Error: 941" and it says "My Vodafone can't be downloaded". Please fix this!"

The study takes about 30 minutes to assign the priority labels to the 29 reviews. Thank you for your valuable participation in this study.

This study has been approved by the Department stated above. However, if you have any concerns about the ethical conduct of the research you may contact the University of Otago Human Ethics Committee through the Human Ethics Committee Administrator (ph +643 479 8256 or email gary.witte@otago.ac.nz). Any issues you raise will be treated in confidence and investigated and you will be informed of the outcome

Participant Evaluation Sheet

Date: 04/02/2020

Participant #: 1

App Reviews

In the following document there are 29 reviews to label. The reviews are followed by 3 labels scale. You are supposed to label each review by circling only one appropriate response on the scale below. Please label each review to the best of your ability and to your own interpretation.

App 1

1.1) Boring, hard to reach goals This game gets boring after a while. It is also impossible to reach certain goals if you don't have friends who play. There needs to be an option to make friends within the game, with other players.

Low Medium High

1.2) Odd problem Marshmallow update makes the butterflies look boxy and funny. I have to restart my S6 Edge for it to work right. Is there an update coming soon for that?

Low Medium High

1.3) It's a great way to pass the time, the graphics are nice and cute but the trouble you go through just to unlock one little part of the garden is ridiculous. 10/13/19 Removed a star because of the recurring issue with the stupid calendar. You guys really should get it together.

Low Medium High

1.4) Enjoying the game quite a bit and like that during the wisp events you get all the butterflies....though completing them all by the end is a challenge.i have one problem though, my butterflies, frog, bird and even the caterpillars have gone blocky while the blossoms went huge. This hasn't been an issue for me before and isn't occurring on starlight...

Low Medium High

1.5) Messed up Ever since the update everything is enlarged and squished together. Have to reinstall just to see it right. What's the point if I have to do this every time I open the game. Please fix.

Low Medium High

1.6) These updates are killing me! Before everything was enlarged and overlapping and I was able to temporarily fix the issue by restarting my phone. Now I'm having the issue of extremely pixelated graphics. These recent updates are horrible and truly ruining an enjoyable gaming experience.

Low Medium High

1.7) Though the game is amazing Events are impossible to complete. And this game is all about events. You can barely complete a set of butterflies and never perfect it in one go. And the same set never repeats. Aka you have to spend a lot.. really a lot just to complete 1 set of butterflies. And spoiler alert there are way to many..

Low Medium High

App 2

2.1) For some reason the developers have decided to screen Jono and Ben as a series of a few of the skits from each episode, meaning that there is no way to watch a full episode, only some fragments of it. And most of these glitch out and black out the entire video apart from the sound and a small line of moving colour at the bottom of the screen. Please fix, this app is a disgrace. TVNZ Ondemand is way better, mostly because it actually works.

Low Medium High

2.2) works ok after a few tries but now just plays ads and plays less shows. Tvnz on demand might have allot of ads but at least its reliable!!

Low Medium High

2.3) To much ads because sometimes I accidentally exit the program and then I have to watch another set of ads which is really annoying

Low Medium High

2.4) Worst This seriously needs an overhaul. Can't even watch a full episode of anything stop starts all the time pauses for ages. No problem viewing the adds though.BT not the point fix it

Low Medium High

2.5) This app use to work on my s9+ but now crashes every time i even attempt to click on the app and open it. Have restarted my phone, deleted and redownloaded the app numerous times. This has been my 3rd time and still it does not work. Unsure of how this happened as it worked fine the first time i downloaded it however it did freeze my screen a few times and gave me panic attacks that I couldnt turn my screen on amd was stuck with a black screen.

Low Medium High

2.6) Its so basic. You can't save fav shows, you can't see which episodes you've watched. It doesn't store stuff you recently watched. You can't even see basic info like the date the show aired. I use it on my smart tv and it's just pathetic

Low Medium High

2.7) Worst app for reliability. Wifi strength is good, Netflix working fine, TVNZ working fine but Three.... stuttering and crashing, sound only as image freezes, jump back 30secs and ends up taking 5mins of your life before you give up and quit. These issues have been going on now for a year or more which is really dissapointing! Hate that whenever you stop playing an episode due to the issues listed above and go back in your are forced to watch adverts again... btw the adverts work perfectly fine everytime, go figure! Grr

Low Medium High

App 3

3.1) I can't use half of trade me features on the app. I got smart phones and tablets so that I didn't need my laptop anymore..... If I want to edit a product, offer an item fixed price, see how many watchers/views my auctions have at a glance I have to use a pc why can't you just make the app better.....

Low Medium High

3.2) Seller info The app works smoothly and layout is cool but the seller info is not shown which makes it difficult for verification and I have to sign in using the browser.

Low Medium High

3.3) Really good. You should be able to have the option of not logging in, so you can just browse... other than that god app

Low Medium High

3.4) It's ok... thou still have to use desktop version to do simple things like update payment details and payment instructions which is annoying! Also it doesn't have notifications for questions on your listings if you are selling have to rely on emails...

Low Medium High

3.5) Stupid update Why do we now have a notifications section on our trade me? I'm sick of clearing them all the time. Please let us choose whether we want these or not.

Low Medium High

3.6) I have searching for apartment to rent but the search results are a bit confusing! Please make the listings visible just for a limited time. I can see listings placed for than a year.

Low Medium High

3.7) 1st time using the listing function. Kept getting told I need to input a figure less than 1000 but I never had any figures over 300. Ended up using desktop to complete the action. The tired to answer some questions on my auction and got told I couldn't do that either coz trade me had requested more info but I couldn't find any requests... frustrated the hell out of me the 1st 20minutes ever using this app

Low Medium High

3.8) Great app but needs all options available to pc users Desperately needs view sellers other listings opton. Also need to be able to see exactly when auctions are closing not just a rough idea in the closing hours

Low Medium High

App 4

4.1) Good but impractical app A good app but updates all the time and with each update it requires more space on your device. I've deleted just about everything off of my phone and there still Ian't enough space for it. It won't let you stick with a previous version either.

Low Medium High

4.2) Fussy, overly animated, frequently broken and has a layout that looks like it was designed by two year old. Update: it's still useless but at least it tells you how many mb you have left.

Low Medium High

4.3) I can't seem to download the app due to "Error:941" and it says "My Vodafone can't be downloaded".

Low Medium High

4.4) Great but hate updates Hate updating the app!!! I have very limited space on my phone so sometimes have to delete apps to upgrade or I can't use the app. So stupid!!

Low Medium High

4.5) Spyware Latest version requires access to a list of all running applications. No justification for this given what the app is intended to do.

Low Medium High

4.6) The summary screen no longer shows remaining minutes. Remaining data still shows along with Fantastic Fridays and Text, but no minutes. Remaining minutes have always been there until now. Please fix this in the next update.

Low Medium High

4.7) Not able to fully view when my prepay credit expires, as the text displayed fails to fit my mobile screen. Also charges are not detailed to specify what the charge is for, as Vodafone cannot be trusted to get things right! New app is unacceptably slow and crashes all the time! I absolutely HATE the new sign in! I also absolutely HATE how you force customers to change their preferred way to sign in to the app! Because I HATE the new app and you did not ask my permission to change the way I use the app, I have uninstalled it, and will be seeking to take my business elsewhere! I also HATE contacting Vodafone and talking to non-New Zealanders outside New Zealand, plus Immigrants in my homeland who disrespect my Kiwi culture, and have poor English!!!

Low Medium High

E. List of requirements prioritisation methods

ID.	Method	Number of studies
1	Analytical Hierarchical Process	42
2	Cumulative Voting	13
3	Quality Function Deployment	12
4	Numerical Assignment	11
5	Planning Game	10
6	Hierarchical Cumulative Voting	9
7	Cost Value	8
8	Fuzzy Analytical Hierarchical Process	7
9	Priority Groups	7
10	Ranking	7
11	Binary Search	6
12	Case Base Ranking	6
13	Cost and Benefit Prediction	6
14	EVOLVE	6
15	Hierarchy Analytical Hierarchical Process	6
16	Pairwise Comparison	6
17	Top 10	6
18	Value Oriented Prioritization	6
19	B – Tree (Binary Tree)	5
20	Cognitive Approach	5
21	FUZZY Logic	5
22	Minimal Spanning Tree	5
23	MosCoW	5
24	Bubble Sort	4
25	Fuzzy Multi Attribute/Criteria Decision Making	4
26	Kano Model	4
27	SERUM(Software Engineering Risk: Understanding and Management)	4
28	Value Based Requirements Prioritization	4
29	AGORA(Attribute Goal Oriented Requirements Analysis)	3
30	Binary Priority List	3
31	Conceptual Model	3
32	Interactive Genetic Algorithm	3
33	Minimal Marketable Features	3
34	Multi Criteria Decision	3
35	Value Based Intelligent Requirements Prioritization	3
36	Weiger's Method	3
37	Win Win	3
38	ABC Framework	2
39	Automated Requirements Triage	2
40	Dot Voting	2
41	Eclipse Process Framework	2
42	Extensive Numerical Assignment	2
43	Group Recommendation Heuristics	2
44	Hybrid Assessment Method (HAM)	2
45	Internal Evident Reasoning	2
46	Lanchester Theory	2

ID.	Method	Number of studies
47	Larman	2
48	Linear Programming-GW-Analytical Hierarchy Process	2
49	Linear Regression	2
50	Mathematical Programming Technique	2
51	Multi Criteria Preference Analysis Requirements Negotiation	2
52	Multi Objective Next Release Problem	2
53	Multi Voting System	2
54	Other (Plan Based + Agile)	2
55	PHandler	2
56	Ping Pong Balls	2
57	Quality Function Deployment – Linear Programming	2
58	Ranking based on product definition	2
59	Relative Weighting	2
60	Requirements uncertainty prioritization approach	2
61	SNIPR	2
62	Theme Screening/Scoring	2
63	Theory W	2
64	TOPSIS	2
65	Weiger's Matrix Method	2
66	Weighted Sum Method	2
67	100 Points	1
68	100$ Method	1
69	Adaptive Fuzzy Decision Matrix Model	1
70	Adaptive Fuzzy Hierarchy Cumulative Voting	1
71	Adaptive Requirements Prioritization	1
72	Adhoc Prioritization	1
73	AHP-GORE-PSR	1
74	Alpha – Beta – Gamma Framework	1
75	Analytic Network Process (ANP)	1
76	Apriori Technique	1
77	Architecture Driven	1
78	Binary Inputs	1
79	ConTexter	1
80	Contextual Requirements Prioritization	1
81	Correlation Based Assessment Framework	1
82	Cost of Delay	1
83	Decision Weighted Matrix	1
84	DRank	1
85	Dynamic Reprioritization of requirements in Agile Development	1
86	ELECTRE - I	1
87	Enhanced Genetic Algorithm	1
88	Evolutionary Algorithms	1
89	Fuzzy Hierarchy Cumulative Voting	1
90	Fuzzy Quality Function Deployment	1
91	Fuzzy TOPSIS	1
92	Game Theory	1
93	Genetic Algorithm	1

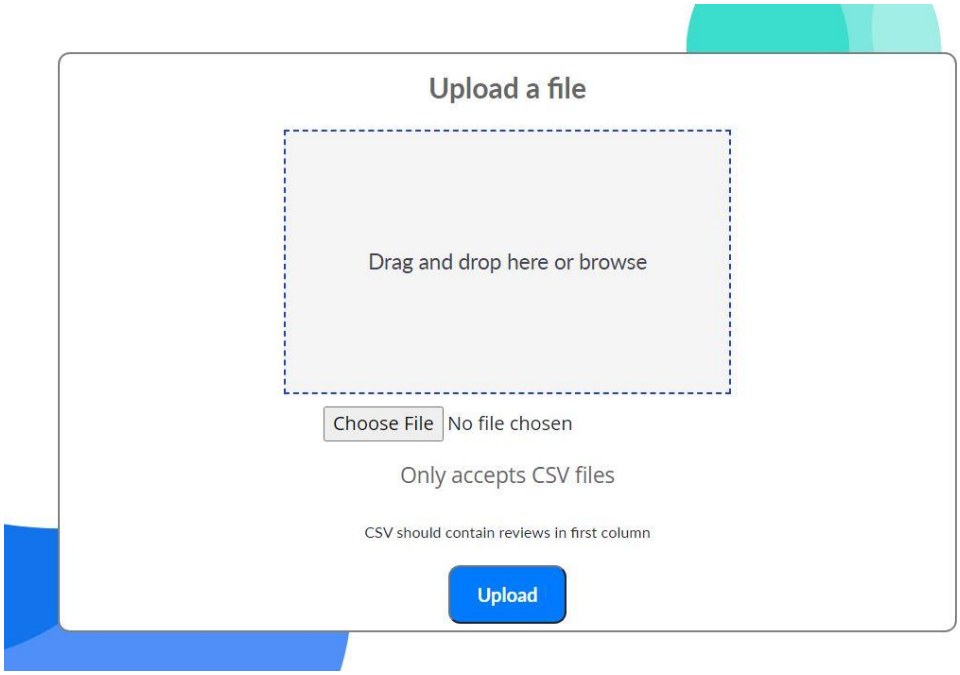
ID.	Method	Number of studies
94	Goal Based Technique	1
95	Goal Skill Preferences	1
96	GOASREP (Goal Oriented Software Requirements Elicitation & Prioritization)	1
97	Gradient Descent Ranking	1
98	Grey Relational Analysis	1
99	GW- Analytical Hierarchy Process	1
100	Hierarchical Dependencies	1
101	Importance Performance Analysis	1
102	Incomplete Analytical Hierarchy Process	1
103	Incremental Funded Methodology	1
104	Individual Attribute Based Ranking	1
105	Integrated Prioritization Approach (IPA)	1
106	K-Means	2
107	Laplace Evidential Reasoning	1
108	Maintainability Based	1
109	Majority Voting Goal Based	1
110	Meta Networks Based	1
111	MPRAN	1
112	Multi Attribute Utility Theory	1
113	NAcAHP (Numerical Assignment + Analytical Hierarchy Process)	1
114	Natural Language Processing	1
115	New Lanchester Theory	1
116	Other (Cumulative Voting + Decision Weighted Matrix)	1
117	Other (Data Mining + Machine Learning)	1
118	Other (Lagrange Function + Group Decision Making)	1
119	Other (Multi Voting + Binary Search)	1
120	Other (Quality function Deployment + Yager's Algorithm)	1
121	Other(Data Mining + Recommender System)	1
122	Other(Satisfactory Modulo Theory + Pairwise Comparison)	1
123	Others(Multi Criteria + Automated Reasoning)	1
124	Outranking	1
125	Partial Order Assimilation	1
126	PGcAHP (Planning Game + Analytical Hierarchy Process)	1
127	Planning Poker	1
128	Preference Weights	1
129	Prioritization of Stakeholder Values using Metric	1
130	Priority ranking using topological potential	1
131	PROMETHEE	1
132	Psychotherapy For System Requirements	1
133	Purpose Alignment Model	1
134	Quantitative Framework	1
135	REMBRANDT (Multi Criteria Decision Analysis based)	1
136	RepiZer	1

ID.	Method	Number of studies
137	RepoTizer	1
138	Requirements Interdependencies Technique	1
139	Round the group prioritization	1
140	Sample Selection	1
141	SELRank	1
142	Simple Additive Weighting Rating Technique	1
143	Single Multi Criteria Rating Technique	1
144	Situation Oriented Evaluation	1
145	Stratified Analytical Hierarchy Process	1
146	Technique for ordering from similarity to ideal solution	1
147	Technique of bucketing requirements	1
148	Tensor Decomposition	1
149	Thurston's Law of Competitive Judgement	1
150	Value Based Fuzzy Requirements Prioritization	1
151	Value Oriented Framework	1
152	Value Oriented Hierarchical Cumulative Voting	1
153	Verbal Decision Analysis	1
154	Visualization Technique	1
155	Weighted Critical Analysis	1
156	Meta Model Based Requirements Prioritization	1
157	Market Driven	1

F. Screenshots based walkthrough of operational demonstration

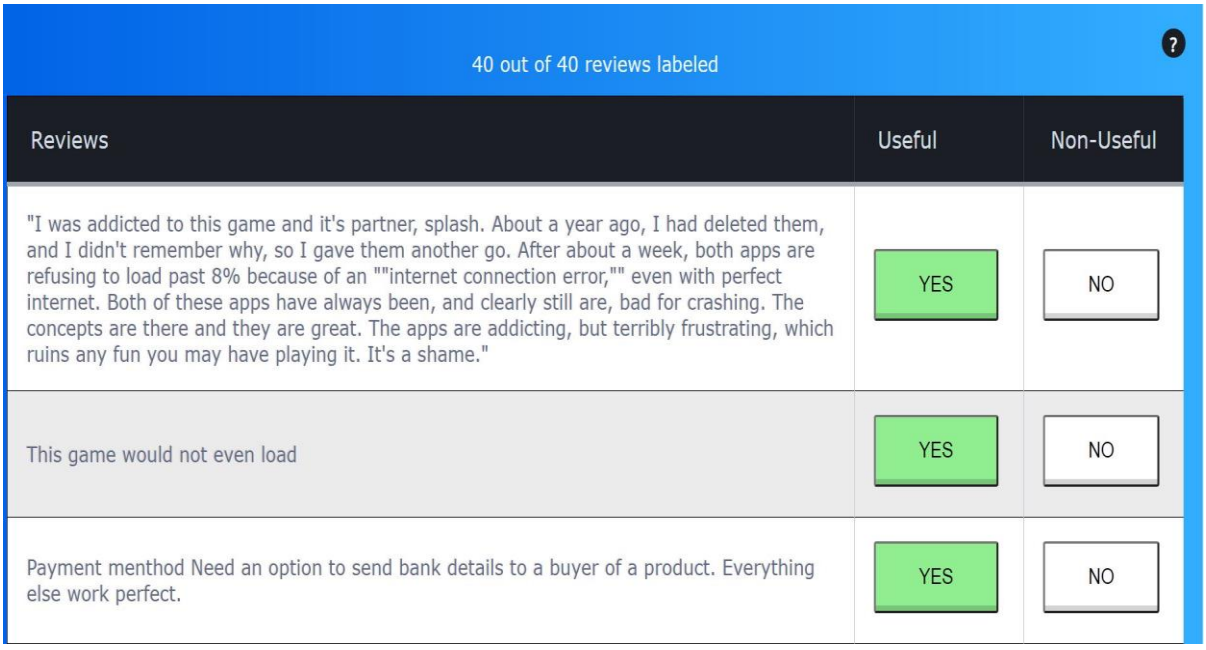
F.1 Reviews upload page

After accessing the link to RECP (RECP - Reviews Elicitation Classification Prioritisation) web tool via a web browser and successfully completing the registration process, the end-user can login and select the 'TRY IT' option to visit the reviews upload page. This page provides the necessary options for the end-user to upload the CSV containing only reviews in its first column. The end-user needs to select 'Upload' button after providing the CSV file to the web tool.



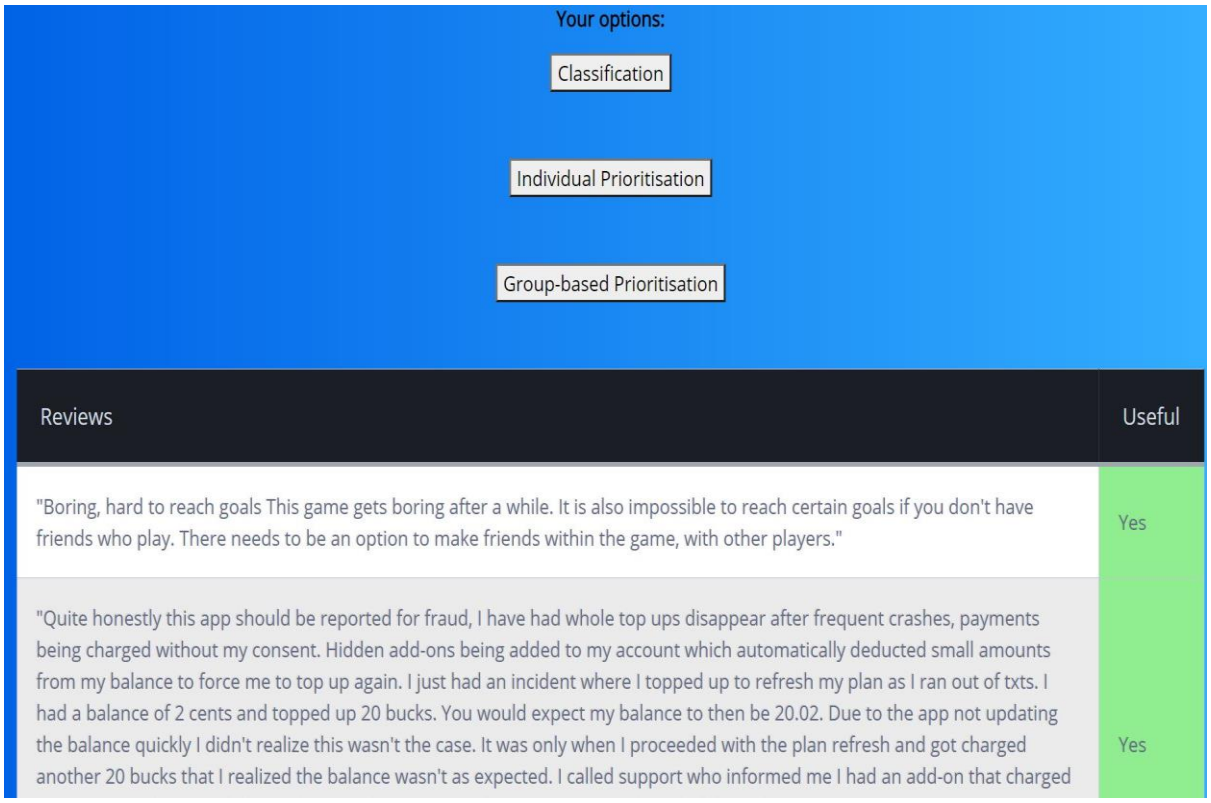
F.2 Manually tagging 50% reviews for filtering

After the CSV file containing the reviews has been successfully uploaded, the end-user will be directed to a page where the end-user needs to tag 50% of the reviews as 'Useful' or 'Non-Useful'. The end-user needs to click the appropriate buttons as mentioned below to tag a particular review as 'Useful' or 'Non-Useful'. Once the end-user has tagged 50% of the reviews, the end-user will be presented with a 'Continue' button to proceed with the filtering of remaining useful reviews. The end-user needs to click this button to proceed with the filtering task. This filtering task depicts the phase 2 of the undertaken research work.



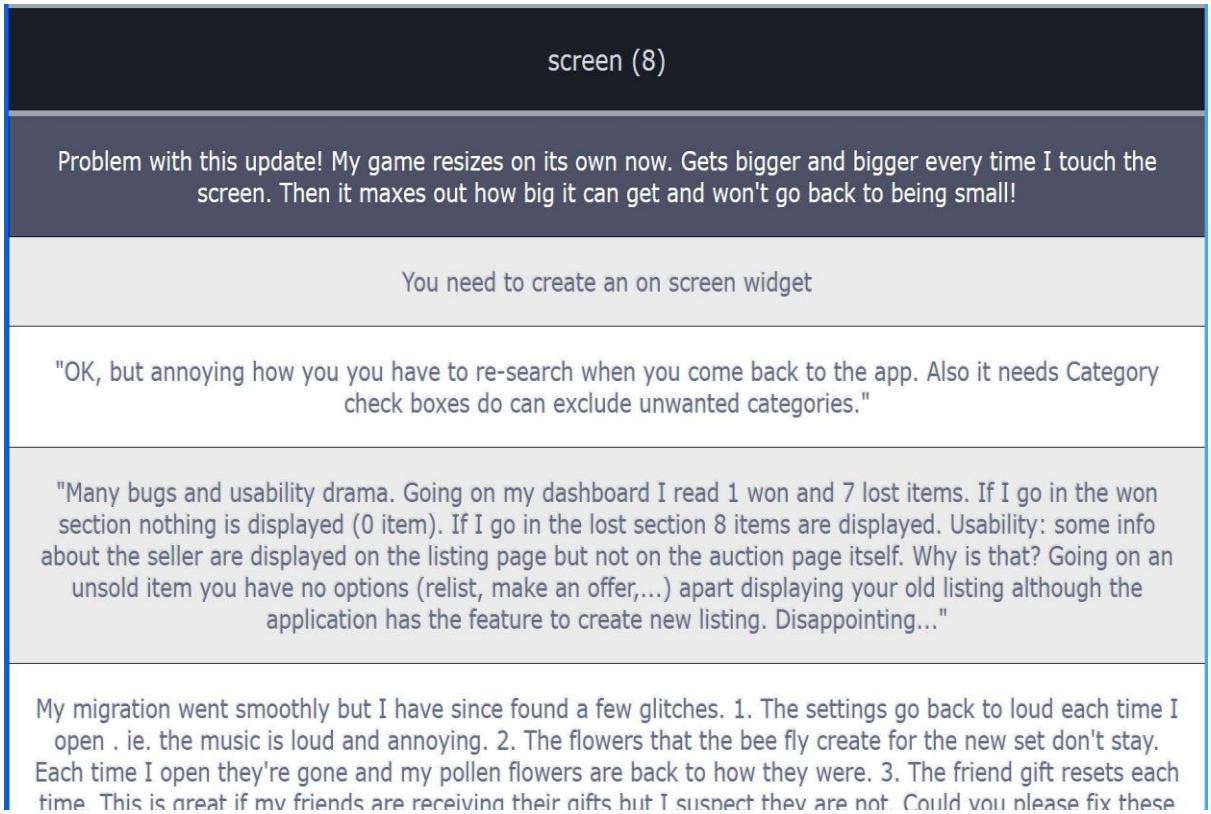
F.3 Filtered useful reviews

Once the filtering process is completed, the end-user is directed to a page where the end-user can view the classified 'Useful' and 'Non-Useful' reviews. Along with this, the end-user is provided with three options - Classification, Individual Prioritisation and Group-based prioritisation to select. These options (i.e., Classification, Individual Prioritisation and Group-based Prioritisation) reflect phase 3 and phase 4 of the undertaken research respectively.



F.4 Classified useful reviews

When the end-user selects the 'Classification' option, the end-user is directed to a page that displays the results of classification method. Initially, the end-user is presented with a list of groups. In the display list, the name of the group is followed by the number of useful reviews it holds. When the end-user clicks a particular group, a list of useful reviews the group holds is displayed as shown below.



F.5 Individual prioritisation

When the end-user selects the 'Individual Prioritisation' option, the end-user is directed to a page that displays the results of individual prioritisation method. The end-user is presented with a list of prioritised useful reviews. In the display list, the particular useful review is accompanied by its associated priority as shown below.

Reviews	Priority
With SOOO many people saying the same thing youd think tv3 would fix This app. And it's not that there are ads it's ALL the places they pop up like every time the app freezes... which is a lot!!!! But now this app is frustrating as it keeps putting up a playback error tap here sign then when you tap the sign stays on the screen and the loading signal shows while the movie starts playing behind it. And it stays like this unless you go out of the movie and suffer ANOTHER ad coming in again. And what's with all the ads every time you pause the movie or have to go back out and come back on to movie. Aren't the ads through out the movie enough???? Why do we have to watch all these other ads too. It's more punishing than watching normal tv :-(But sometimes I miss something and I want to watch it :-(	0.92
"Quite honestly this app should be reported for fraud, I have had whole top ups disappear after frequent crashes, payments being charged without my consent. Hidden add-ons being added to my account which automatically deducted small amounts from my balance to force me to top up again. I just had an incident where I topped up to refresh my plan as I ran out of txts. I had a balance of 2 cents and topped up 20 bucks. You would expect my balance to then be 20.02. Due to the app not updating the balance quickly I didn't realize this wasn't the case. It was only when I proceeded with the plan refresh and got charged another 20 bucks that I realized the balance wasn't as expected. I called support who informed me I had an add-on that charged $1 when I used 25mb of data.. whatever that means. Either way I've never heard of that add-on nor is it listed against my account or sold on the store. How sneaky. I managed to get the money refunded but be warned. This sort of thing has happened way to often."	0.81

F.6 Group-based prioritisation

When the end-user selects the 'Group-based Prioritisation' option, the end-user is directed to a page that displays the results of group-based prioritisation method. Initially, the end-user is presented with a list of groups along with their generated priorities. In the display list, the name of the group is followed by the number of useful reviews it holds along with its associated priority. When the end-user clicks a particular group, a list of useful reviews the group holds is displayed along with the priorities of those useful reviews as shown below.