DISSERTATIONES
INFORMATICAE
UNIVERSITATIS
TARTUENSIS
13

HUISHI YIN

Using a Kano-like Model to Facilitate Open Innovation in Requirements Engineering

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

13

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS
13

HUISHI YIN

Using a Kano-like Model to Facilitate Open Innovation in Requirements Engineering

UNIVERSITY OF TARTU
Press

Institute of Computer Science, Faculty of Science and Technology, University of Tartu, Estonia.

Dissertation has been accepted for the commencement of the degree of Doctor of Philosophy (PhD) in informatics on October 24, 2019 by the Council of the Institute of Computer Science, University of Tartu.

Supervisor

Prof.	Dietmar Alfred Paul Kurt Pfahl University of Tartu, Estonia

Opponents

Department Head	Andreas Jedlitschka Fraunhofer Institute for Experimental Software Engineering IESE, Germany
Assoc. Prof.	Richard Berntsson Svensson Chalmers and University of Gothenburg, Sweden

The public defense will take place on December 17, 2019 at 10:15 in J.Liivi 2-405.

The publication of this dissertation was financed by the Institute of Computer Science, University of Tartu.

ISSN 2613-5906
ISBN 978-9949-03-210-5 (print)
ISBN 978-9949-03-211-2 (PDF)

University of Tartu Press
http://www.tyk.ee/

To my family and friends

ABSTRACT

When Requirements Engineering (RE) is applied, requirements analysis is often used to determine which candidate requirements of a feature should be included in a software release. This plays a crucial role in the decisions made to increase the economic value of software. Nowadays, products evolve fast, and the process of requirements prioritization is becoming shorter as well. Companies benefit from receiving quick feedback from end users about what should be included in subsequent releases. One effective approach supporting requirements prioritization is the Kano model. The Kano model defines the relationship between user satisfaction and product features. It is a method used to classify user preferences according to their importance, and in doing so, supports requirements prioritization. However, implementing the Kano model is costly and time-consuming, and the application of the Kano model cannot be repeated quickly. Moreover, this is even more difficult for small companies because they might not have sufficient funds and resources to contact end users and conduct interviews. This puts small businesses, especially start-ups, at an unfair disadvantage in competing with big companies.

To address this problem and make the application of the Kano model simpler, faster, and cheaper, we propose evolving the Kano model in two aspects. First, free online text data should be used to replace responses collected from interviewees. Second, in order to handle the higher amount of data that can be collected from free online text data and in order to facilitate frequent analyses, the data analysis process should be automated.

The goal of this research is to propose methods for (semi-)automatically classifying user opinions collected from online open sources (e.g., from online reviews) to help decision-makers decide which software requirements to include in subsequent product versions. To achieve this research goal, we propose the Open Innovation in Requirements Engineering (OIRE) method to help software organizations gain a better understanding of user needs and satisfaction with existing products. A key element of the OIRE method is its Kano-like model. This Kano-like model mimics the traditional Kano model, except that it uses data from online reviews instead of interviews conducted with select focus groups. We use machine learning and sentiment analysis methods to deal with text lines corresponding to the input of the Kano-like model.

The purpose of the OIRE method is to help software organizations assess end users' level of appreciation for software products to be developed. In addition, a partly automated approach lowers software RE costs. We think that the OIRE method mostly offers benefits to small companies with small teams and a low marketing and customer research budget.

Another significant contribution of this research is that we present three typical use cases and a proof-of-concept of the OIRE method that demonstrate the applicability of the OIRE method using real-world data collected from the Internet. We

also discuss the suitability of potential input sources for the OIRE method.

A further contribution of this research is the design and implementation of a web-based prototypical system, the OIRE System (OIRE-S), and the evaluation of the OIRE method using OIRE-S. We conducted a case study with two Chinese companies and an interview study with two other stakeholders. The results of the case study and the interview study show that the OIRE method was perceived to be useful by all stakeholders, and that it was perceived to be most useful for decision-makers in small companies.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

A - Attractive quality
ACC - Analysis of complaints and compliments
AW - Adversative words
BST - Binary search tree
CIT - Critical incident technique
CrowdRE - Crowd-based requirements engineering
CS - Computer science
CV - Cumulative voting
DPV - Dysfunctional predictive value
FPV - Functional predictive value
I - Indifferent quality
IW - Intense words
M - Must-be quality
N - No vector
NEW - Negative emotional words
NW - Negative words
O - One-dimensional quality
OI - Open innovation
OIRE - Open innovation in requirements engineering
OIRE-S - Open innovation in requirements engineering rystem
OSS - Open source software
PD - Product developer
PEW - Positive emotional words
PG - Planning game
PLM - Product lifecycle management
PO - Product owner
PPS - Probability proportional to size
Q - Questionable
QFD - Quality function deployment
R - Reverse quality
RE - Requirements engineering
Qu - Question
SA - Sentiment association
SE - Software engineering
SVM - Support vector machine
VEPW - Very positive emotional words

VNEW - Very negative emotional words
VOP - Value oriented prioritization
Y - Yes vector

1. INTRODUCTION

The overarching goal of this thesis is to extend the body of knowledge regarding Open Innovation (OI) in software Requirements Engineering (RE). Since 2003, when Henry Chesbrough proposed the concept of OI [17], Closed Innovation gradually fell behind while more and more companies began to apply OI in their business. The increasing popularity of OI can be observed in many research fields, including computer science [106]. Nevertheless, according to a mapping study conducted by Wnuk et al. [104], there is still room for research on the potential of OI in Software Engineering (SE), one example being the field of software requirements engineering [110].

Due to time and effort limitations, we decided to focus on requirements analysis in this thesis, in particular on classifying and prioritizing requirements. When RE is applied, requirements prioritization is performed to rank requirements in their order of importance. Prioritization is often used to determine which candidate requirements of a feature should be included in a software release. This plays a crucial role in making decisions aimed at increasing the economic value of a piece of software [1]. However, simple one-dimensional prioritization of requirements is not necessarily clearly correlated with end user satisfaction and thus economic value. To capture the more complex relationship between the economic value of a piece of software and its attributes (or features), the Kano model was developed by Noriaki Kano in the 1980s [46].The Kano model classifies customer requirements (or potential features) into five categories. The different categories of customer requirements influence user satisfaction and dissatisfaction [46]. Beyond a simple prioritization of requirements, the Kano model also provides a mechanism for identifying (1) the set of requirements that must be implemented in order to succeed, and (2) the set of requirements that must not be implemented in order to prevent failing.

1.1. Research Approach

In our research, we have designed a Kano-like model according to the Kano model theory. The main goals of this study are: 1) to develop a new approach that could (semi-)automatically classify user reviews collected from online open sources to help decision-makers decide which software requirements to include in subsequent product versions; 2) to demonstrate the applicability and usefulness of the new approach. To achieve these research goals, we use an engineering approach comprising the following steps: a) an analysis of the state of the art; b) the design of a new method by combining existing approaches and ideas in a new way; c) the development of prototypical tool support; d) validation (proof-of-concept using several use cases); e) evaluation in industry. Steps a) to e) are covered by Chapters 3, 4, 5, 6, and 7, respectively. The steps of the research approach followed in this thesis are shown in Figure 1.

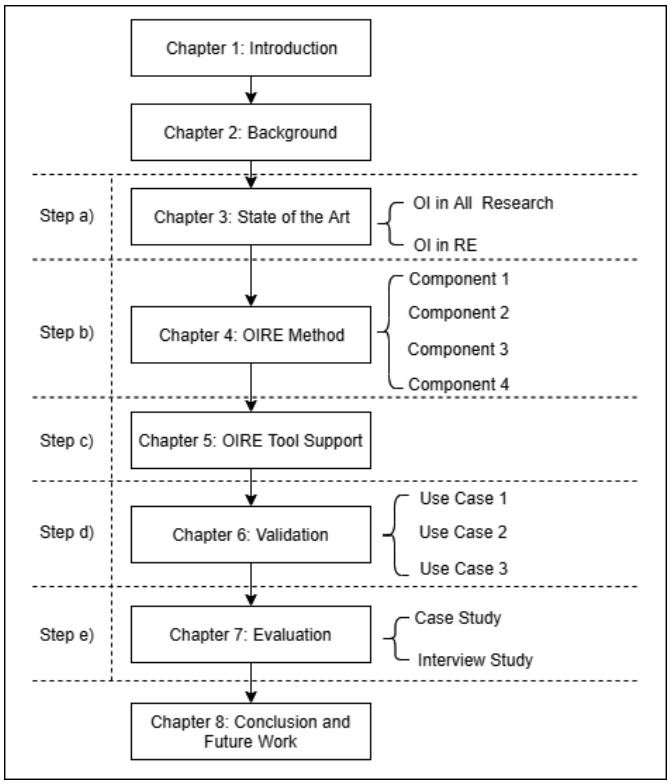


Figure 1. The steps of the research approach

We propose the Open Innovation in Requirements Engineering (OIRE) method to help software organizations gain a better understanding of user needs and satisfaction with existing products. The OIRE method is also designed to identify user needs that are not addressed well by competing products. We believe that implementing the OIRE method will help software organizations understand the impact of a software product to be developed on user appreciation. This is particularly an issue for small companies. Due to a lack of resources, social influence, user feedback, etc., small companies, particularly start-ups, rely almost exclusively upon their own expertise with regard to technological innovation [111]. Hence, we believe it could be beneficial for them to be able to use results from automated analyses of freely available online data, such as provided by the OIRE method. In addition, a partly automated approach could lower RE costs. Therefore, we think that the OIRE method will especially offer benefits for small companies with small teams and a low marketing and customer research budget.

1.2. OIRE Method

The OIRE method mimics the well-known Kano model, except that it uses data from online reviews instead of interviews conducted with select focus groups. The

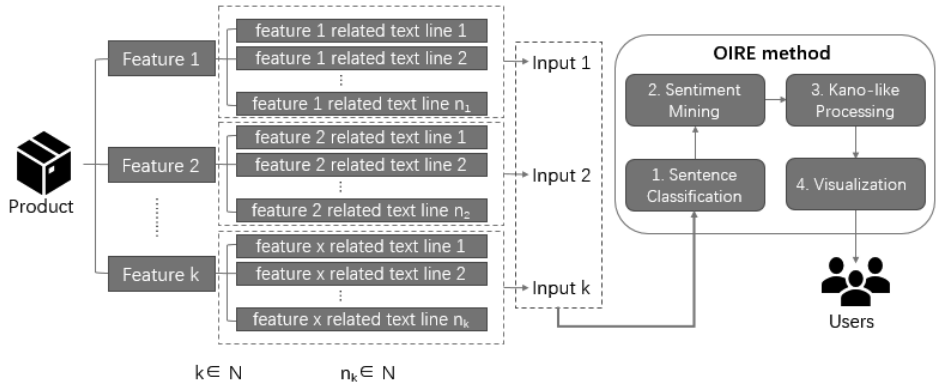


Figure 2. Expected input for the OIRE method and the components of the OIRE method

right half of Figure 2 shows the components of the OIRE method and the order in which they are typically used. The OIRE method comprises four components: Sentence Classification, Sentiment Mining, Kano-like Processing, and Visualization.

Suppose, for instance, that we have an input file with twelve text lines relating to Feature A of a software product. The process of the OIRE method, using all of its components from input to output, is shown in Figure 3. We will describe each component below. A more detailed introduction will be given in Chapter 4. The content of the input file is shown at the top of Figure 3.

Component 1 - Sentence Classification: We use machine learning to classify the text lines of the input into two classes, "functional" and "dysfunctional". This classification is inspired by the "functional question" and "dysfunctional question" of the traditional Kano model [46]. Text classified as "functional" corresponds to text lines stating the presence of a feature, while text classified as "dysfunctional" corresponds to text stating the absence of a feature [109]. The unit of analysis of the classifier is one line of text. Figure 3 shows the example input and the expected output of Component 1.

Component 2 - Sentiment Mining: We use a dictionary-based method [109] to calculate the sentiment score of each text line in each of the two classes (output from Component 1). Then we classify the polarity of the sentiment (from very negative to very positive) of each text line according to its sentiment score and translate it into the corresponding Kano score. For example, the sentiment "very negative" corresponds to a Kano score of "-2" and the sentiment "very positive" corresponds to a Kano score of "+2". Figure 3 shows the example input and the expected output of Component 2.

Component 3 - Kano-like Processing: We have designed a Kano-like model algorithm that is applied when the traditional Kano model cannot be used because the functional and dysfunctional input is unpaired or partially missing [108]. Figure 3 shows the example input and the expected output of Component 3. In the

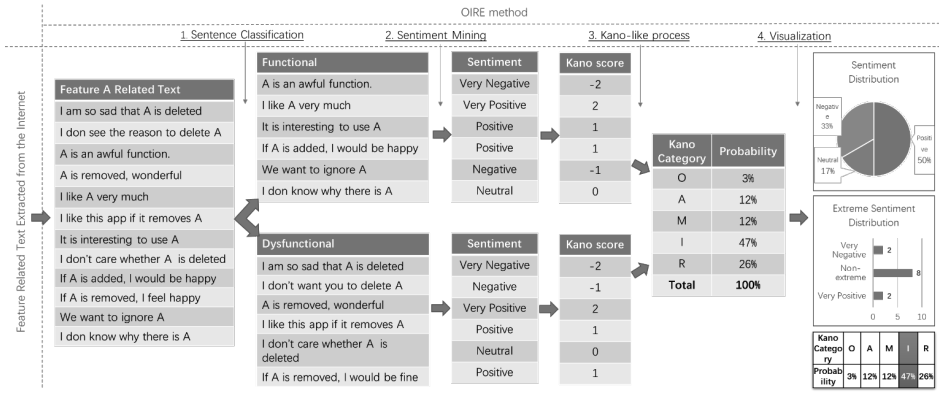


Figure 3. The process the OIRE method

OIRE method, we do not assign a final Kano category to a feature but rather a probability distribution over the Kano categories. This is different from the traditional Kano model, where the most frequently assigned Kano category is used as the last assigned category value.

Component 4 - Visualization: We visualize the outputs of Components 1 to 3 for the user. Figure 3 shows three examples of different output formats: pie chart, bar chart, and table.

The left half of Figure 2 also shows how the input for the OIRE method has to be formatted. We see from Figure 2 that a product may have several features. We put all text lines related to one feature into one file as the input. Thus, each product consists of a set of input files, with each file corresponding to one feature and containing all text lines related to exactly one feature. All the input text used in this thesis was taken from the Internet, in particular from online open sources.

1.3. Contributions

To summarize, our contributions in this work are as follows:

- We conducted a systematic mapping study to survey the state of the art of OI in the sub-fields of SE, especially in RE.
- We propose the OIRE method.
  - We provide a solution by applying machine learning to determine whether a text line extracted from an online open source potentially corresponds to an answer to the functional or dysfunctional question asked in the Kano model.
  - We designed a dictionary-based method to classify the sentiment found in text lines into five sentiment classifications: Very Negative, Negative, Neutral, Positive, and Very Positive.
  - We propose and validate the Kano-like model. This model follows the

Kano model theory and can be used as an approximation of the traditional Kano model in situations where the input to the Kano model is unpaired or partly missing.

- We designed a web-based prototypical system, the OIRE System (OIRE-S). OIRE-S is a tool supporting the OIRE method.
- We present three typical use cases and a proof-of-concept of the OIRE method, which demonstrate the applicability of the OIRE method using real-world data collected from the Internet.
- We interviewed industry people and conducted a case study and an interview study to evaluate the usefulness of the OIRE method.

1.4. Outline

After the introductory Chapter 1, Chapter 2 introduces the main concepts involved in this thesis, including open innovation, requirements engineering, the Kano model, and classification.

In Chapter 3, we introduce the literature relating to OI in different research fields. When analyzing peer-reviewed literature on OI, we observed that the field of Computer Science seems to have significantly less diversity than all other fields. To further understand the research status of OI in RE, we summarize the body of knowledge regarding the use of OI in the field of RE. More specifically, we analyze what uses of OI in the context of RE have been reported and how OI has contributed to individual steps of the RE process.

In Chapter 4, we introduce the composition of the OIRE method as well as the design and verification of the algorithm of each component. Based on the OIRE method, we will develop a web-based prototypical system as tool support for the OIRE method: OIRE-S.

In Chapter 5, we introduce the design and implementation of OIRE-S.

In Chapter 6, we present three typical use cases and a proof-of-concept of the method, which demonstrate the applicability of the method using real-world data collected from the Internet.

In Chapter 7, we describe a case study and an interview study that we conducted to evaluate the OIRE method using OIRE-S.

We conclude this thesis and suggest directions for further study in Chapter 8.

2. BACKGROUND

In this chapter, we will introduce the main concepts involved in this research. Understanding these concepts will help the reader comprehend the subsequent parts of this thesis.

2.1. Open Innovation

In 2003, Henry Chesbrough proposed the term "Open Innovation", defining it as follows: "Open innovation is a paradigm that assumes that firms can and should use external ideas as well as internal ideas, and internal and external paths to market, as the firms look to advance their technology" [17, 107]. The concept has also been described as "Companies should traverse the firm's boundaries to absorb technology sources from other companies" [17], meaning that boundaries between a company and its environment should become permeable so that innovations can easily move from outside the company boundaries to the inside and vice versa. This also implies that companies and their partners start sharing risks and rewards.

In the past decade, OI has become a major element of companies' innovation progress in almost all industries. The influence of OI in the development and evolution of software products has become significant [16, 21, 87]. OI has been identified as a new strategy for software-developing organizations to benefit from the exchange of innovative ideas and the adoption of value-creating processes across and beyond company boundaries [38].

However, we found there is still room for research on the potential of OI in software engineering [111]. For example, there is a lack of empirical research regarding the use of OI in software engineering [104]. We also found that the OI journey of small enterprises has not been smooth because small enterprises do not have enough resources (e.g., financial and human resources, etc.) to take the initiative in terms of OI. At the same time, their ability to bear the risk of OI is weak.

In a mapping study, Hussan Munir et al. stated that "OI is not for free" [69]. According to their research, they found that "in order to gain the full and long term benefits from OI, companies must invest in the open communities, and since these are complex networks with a multitude of actors, these companies must have a clear resources investment plan, just as they need for closed innovations."

Yubing [114] classified OI models according to the flow of direction of technology and knowledge. Yubing believes that when a company is too small, especially a start-up, it is better for it to follow the "inside-out" model, which means that small companies should offer technology to the "outside world" and expect other organizations to commercialize this technology. The main reason for this is that small companies usually cannot afford the cost of buying technology from big organizations or lack the resources to cooperate with other organizations [8, 15]. Through a systematic literature review, Hossain et al. gave a similar conclusion

that small businesses publicize or sell innovative technology to large companies in order to obtain attention and benefit. Hossain et al. also found that it might be true that small enterprises rely more on OI than large businesses [40]. It indicates that when the open innovation model is driven by resources, innovation, and profitability largely in the hands of large companies, small businesses will lose their power of initiative. Our research could use OI to help small businesses save time and costs in the RE process in order to lower the development risk.

2.2. Requirements Engineering

We have been studying the combination of OI and RE to expand the possibilities for enterprises to obtain more resources and to reduce costs and risks. We believe that this will be conducive to small businesses, especially to start-ups.

Since the term "Requirements Engineering" was coined by Mack Alford in 1978, [2] the field has matured a lot and has become one of the most important fields of software engineering [12]. Nuseibeh and Easterbrook defined requirements engineering as follows: "It is the process of discovering that purpose, by identifying stakeholders and their needs and documenting these in a form that is amenable to analysis, communication, and subsequent implementation." [74]. The typical activities involved in RE are requirements elicitation, requirements analysis, requirements specification, requirements validation, requirements management, etc. [91]. Requirements prioritization is a sub-activity of requirements analysis and is also related to requirements management.

In the RE process, requirements prioritization is one of the key activities often used to determine which candidate requirements of a feature should be included in a software release. Requirements are also prioritized to minimize risk during development, meaning that the most important requirements or those with the lowest risk are implemented first [5, 55, 108]. Many requirements prioritization techniques have been proposed. By conducting a systematic literature review, Achimugu et al. identified and analyzed 49 existing prioritization techniques [1].The most frequently used and most prominent techniques include: Analytic Hierarchy Process (AHP) [48, 85], Quality Function Deployment (QFD) [30], Planning Game (PG) [78], Cumulative Voting (CV) [54], Cost-Value approach, sometimes called Value Oriented Prioritization (VOP) [47], and Binary Search Tree (BST) [48]. Other requirements prioritization techniques like MosCow, Bubble Sort, Minimal Spanning Tree, Priority Groups, Win-Win, Top Ten, Wiegers' Matrix Approach, and Binary Priority Listing are also used sometimes [1].

Crowd-based requirements engineering (CrowdRE) is the term for automated or semi-automated methods that contribute to the collection and analysis of "user feedback" from a crowd of people to obtain validated user requirements [35, 36]. According to Hosseini et al. [41], the "Four Pillars of CrowdRE" are crowdsourcer, crowd, crowdsourced task, and crowdsourcing platform. Groen et al. [35] proposed a tentative model for gathering "user feedback" based on the principle

of CrowdRE. This model combines tools that existed in the field at the time, such as "social collaborations, text mining, or data mining". In another study, Groen et al. [36] discussed the benefits, challenges, and lessons learned from several CrowdRE projects and experiments and assessed how to apply CrowdRE approaches and tools in an industrial setting. Their research presented the concepts and models of CorwdRE but did not explain how to implement their models automatically. In a recent mapping study, Wang et al. [98] provided an overview of the literature published on CrowdRE.

In our research, we are interested in a method that is related to but goes beyond simple prioritization of requirements and does not require the involvement of a crowd. To capture the more complex relationship between the economic value of a piece of software and its attributes (or features), we focus on using the well-known Kano model developed by Noriaki Kano in the 1980s [46].

2.3. Kano Model

The Kano model was originally developed by Noriaki Kano [46, 115] who studied Herzberg's Motivation-Hygiene theory [39]. The traditional Kano model defines the relationship between user satisfaction and product features.

Since the 1980s when the Kano model was first introduced, it has become a popular theory used by researchers and business practitioners across many industries. Many researchers use the Kano model to increase user satisfaction and to improve the product design process [10, 11, 75, 97]. After an extensive review of the literature on the Kano model, Josip and Darko summarized and evaluated five methods that classify quality features into the categories defined by the Kano model [65], however, in a different way than Noriaki Kano proposed. The methods analyzed were the original Kano model developed by Noriaki Kano [46], the "Penalty reward contrast analysis" originally proposed by Brandt [10], the "Importance grid" developed by IBM [97], the "Qualitative data methods" including CIT (critical incident technique) developed by Herzberg and ACC (Analysis of Complaints and Compliments) used by Cadotte [11], Oliver [75], Friman, and Edvardsson [32], and the "Direct classification" method proposed by Emery and Tian [27]. Among those five methods, only CIT and ACC are based on the same assumption, i.e., that "quality features can be categorized by comparing how frequently customers mention it in a positive context or a negative context" [65]. However, compared to the Kano model, the reliability of both the CIT and ACC method remains questionable when the frequency with which customers mention features is low. According to Josip and Darko's research, the Kano model and the direct-classification method are the only methods capable of classifying Kano features [65].

The traditional Kano model defines five categories of user needs that have different effects on user satisfaction. These categories are One-Dimensional Quality (O), Attractive Quality (A), Must-be Quality (M), Indifferent Quality (I), and Re-

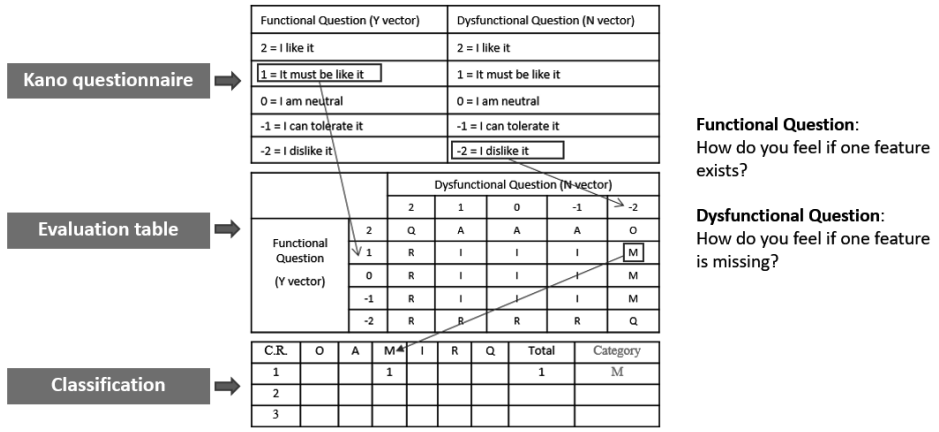


Figure 4. An example of the process of the traditional Kano model

verse Quality (R) [6, 43, 88, 105]. "O" implies that a user is the more satisfied the more they can get from this feature quantitatively. "A" indicates that the feature is unexpected and has a highly positive effect on the user. Such features might distinguish the product positively from other products. "M" implies that if such a feature is not provided or provided with bad quality, the user will be dissatisfied and not buy the product. "I" implies that the user does not care whether the feature is present or not. "I" features are not desirable because they cost money to implement without adding value to the product. "R" implies that a user is dissatisfied if that feature is present in the product [46, 112]. Since it is possible to receive contradictory responses from customers, the category Questionable (Q) is also an option. Use of the Kano model requires the Kano questionnaire. It is composed of a pair of questions, i.e., a functional and a dysfunctional question, that a group of users has to answer for every feature to be categorized. In the software engineering domain, it is a well-known method to classify user preferences according to their importance, and by doing so support requirements prioritization [108]. Figure 4 shows an example of the process of the traditional Kano model.

Since its introduction, many researchers have used the Kano model to improve user satisfaction and the product design process [9, 13, 25, 64]. Some researchers studied the potential use of the Kano model for classifying and prioritizing user needs. They conducted extended research based on the original Kano model. For example, using the Kano model, Zhang and Chen constructed a functional relationship between user satisfaction and product quality. They defined an adjustment coefficient "K" to obtain the importance of user needs [116]. This method relies on questionnaires. Fabijan and Olsson developed a model resembling the Kano model [29]. Their model focuses on software products with rapid customer feedback capabilities. The model defines four types of features named "duty", "wow", "checkbox" and "flow". The input for this model must be collected from

interviews with customers.

Lili et al. [57] established the "eXtreme Programming high-quality analysis module" based on the Kano model theory. This model divides user requirements and developer requirements into eight categories forming a requirements matrix. A demand module is set up to enhance the requirements quality and to reduce misunderstandings, barriers, and potential business risks. This model is an extension of the Kano model. The input to this model is based on discussions with developers and users.

The above-mentioned studies using the Kano model are based on interviews and questionnaires. The difference in our research is that we use online open source data as input to the Kano model. In addition, we automatically use the Kano model theory to classify user needs.

In Nascimento and Aguas's research [70], the data is collected from online sources. However, the role of the online data is not to provide the input data for the Kano model but to help developers locate the people to be interviewed.

There exist also studies focusing on the combination of the Kano model with QFD to improve user satisfaction and to reduce user dissatisfaction [22,33,60,95]. We found two articles that describe a combination of the Kano model and QFD while using online reviews to analyze user satisfaction and the importance of user demands. The goals of these two articles are similar to our research goal. They attempt to analyze online text data using an automated approach based on a combination of the principles of the Kano model and QFD. In Shugang Li and Yueming Li's paper [56], the authors used a new word alignment model for sentiment analysis on online text data. They first calculated a value called "sentiment association" (SA) and then combined this SA with the usage of the Kano model and QFD; finally, they classified all attributes into the Kano categories. Song and Chen [92] combined data mining with the Kano model and QFD to "obtain the real demand of multiple customers as well as the weight of demand". Although some contents of these two articles are close to our research, we found that both focus on improving user satisfaction rather than classifying user needs. At the same time, we noticed that the analysis process used in these two articles is not clear, and it is difficult to understand how the results of the analysis were derived.

Unlike the above research, in our research, we designed a Kano-like model that follows the Kano model principle, to replace the artificial interview or questionnaire step used when implementing the traditional Kano model with a mechanism that automatically classifies online text data. In order to obtain standardized input data that meets the requirements of the Kano-like model, we need to classify the features and related sentiments into two classes corresponding to answers of the Kano paired questions, i.e., functional questions and dysfunctional questions. Within each of these classes, each feature-related text must then be classified into one of five sentiment categories, which are Very Negative, Negative, Neutral, Positive, and Very Positive.

According to Reagan et al.'s study [81], sentiment detection methods can be

one of the following types: 1) dictionary-based methods [51]; 2) supervised learning methods [20]; 3) unsupervised / Deep learning methods [90]. Since, we need to classify text lines into five sentiment categories, multiclass instead of binary classification methods are needed. However, multiclass classification is more intricate than solving binary classification problems [4]. In other words, using supervised machine learning methods is costlier. Since it is easy to implement, we therefore designed a dictionary-based method to classify the polarity of sentiments [109].

There exists quite a lot of research related to sentiment analysis. Sentiment analysis, also known as opinion mining or emotion mining, is a field of study that analyzes texts containing opinions, comments, and evaluations. While research on sentiments and opinions using online texts (e.g., product comments, reviews) started in 2001 [19, 67, 94, 100], the terms "opinion mining" and "sentiment analysis" appeared for the first time in 2002 [71, 77]. Some researchers do sentiment classification at the document and sentence levels with regard to their emotional bias towards either the positive or negative side [96, 102, 113]. Unlike this research, we divide emotions into five categories instead of two (positive, negative) or three (positive, neutral, negative) categories.

In our research, we used supervised machine learning methods to classify lines of text in input files into two categories, functional and dysfunctional. We designed a dictionary-based sentiment analysis method to determine the sentiment polarity in each line of text, and lines of text are classified into five categories: Very Positive, Positive, Neutral, Negative, and Very Negative.

3. STATE OF THE ART

In this chapter, we will introduce the state of the art of OI in RE. This chapter is structured as follows. In Section 3.1, we will present a preliminary study that we conducted to introduce and summarize the research status of OI, specifically, the research distribution of OI in different research fields. In Section 3.2, we will present a mapping study we performed to show the research status of OI in RE.

3.1. OI in All Research Fields

In Section 3.1, we will describe our study results regarding research on OI in peer-reviewed literature in all research fields. The special focus is on the field of computer science as compared to other fields using the categorization scheme of Thomson Reuters' Web of Science Core Collection (ISI Web of Science[1]).

3.1.1. Introduction

In order to get a baseline for the year 2015, the start of our research, we searched for the number of publications with the term "Open Innovation" in their title in two popular repositories, i.e., Google Scholar and the ISI Web of Science. The difference between the two repositories is that Google Scholar does not distinguish predefined categories of publications. Also, Google Scholar automatically collects all kinds of publications (including gray literature) from the Internet, whereas the ISI Web of Science mainly indexes peer-reviewed scientific literature from internationally recognized sources with high scientific standards. As shown in Figure 5, the number of publications on OI considerably increased over the course of eleven years (2003 to 2014). We retrieved a total of 2463 publications on the subject of OI from Google Scholar (all types of publications) and 579 from the ISI Web of Science (peer-reviewed publications).

We see from Figure 5 that the line for Google Scholar increases more sharply than the line for the ISI Web of Science. Starting in 2007, the number of all types of publications grew, especially from 2007 to 2010. In 2011, the number of publications grew once more, but the growth rate was lower. According to the data, over 79% of the total publications (all types) were published during a five-year period (2010 to 2014).

The line of the ISI Web of Science shows that the trend of peer-reviewed publications is more gradual. There is only one big increase shown in 2009, but two significant drops in 2010 and 2014. Nevertheless, over 76% of the total publications were published between 2010 and 2014.

We can recognize an interesting trend in Figure 5, namely that the growth rate of all types of publications (Google Scholar) is much higher than the rate of peer-reviewed publications (ISI Web of Science). This may be because the interest

[1] http://apps.webofknowledge.com

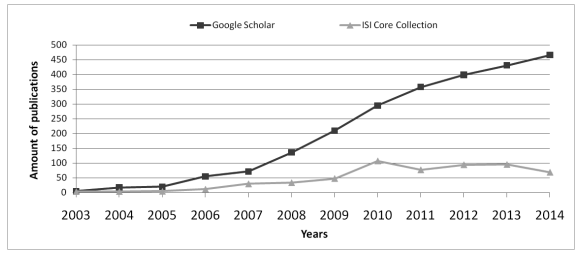


Figure 5. Number of publications on OI (all research fields)

in OI is generally increasing, but thorough research about OI has not yet been conducted – or at least has not been published yet.

The fact that the growth rate of all types of publications (Google Scholar) is much higher than the rate of peer-reviewed publications (ISI Web of Science) seems to suggest that general interest in OI is growing at a higher rate than the number of strictly quality-controlled research on OI. In Section 3.1, we aim to characterize the body of peer-reviewed research literature on OI and find out whether there are differences between research areas.

3.1.2. Goal and Data Collection

In this section, we will try to understand and describe the literature on OI indexed in the ISI Web of Science. To achieve this goal, we aim to answer two questions (Qu 3.1.1 and Qu 3.1.2):

Qu 3.1.1: What topics are discussed in peer-reviewed publications about OI?

Qu 3.1.2: Are there differences in peer-reviewed publications about OI between research areas?

The ISI Web of Science provides us with a convenient interface that accesses several academic publication sources. Using "Open Innovation" as a search term in publication titles, we retrieved a total of 579 publications. The time span was set to the period from 1980 to 2014, but the first hit occurred in the year 2003. Since we were only interested in primary sources, we excluded publications of the type "Review" and similar, and restricted our results set to publications of the types "Article", "Proceedings Paper", "Book", and "Book Chapter". This filtering reduced the total number of publications to 477 (82% of 579), as shown in Table 1. In the remainder of Section 3.1, all analyses will be based on the final results set of 477 publications.

Table 1. Included and excluded publications on OI

	Count	Percentage
Included Types	477	82%
Excluded Types	102	18%
Total	579	100%

After retrieval and filtering of peer-reviewed publications on OI, we used text

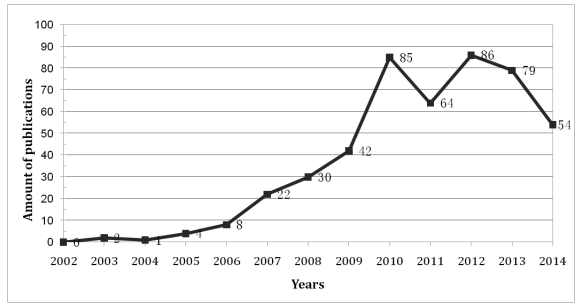


Figure 6. Trend of peer reviewed publications on OI

mining methods for further analysis. The trend of peer-reviewed publications on OI from 2003 to 2014 is shown in Figure 6. We see that the number of publications continues to grow until the year 2010. After 2010, this trend does not continue; rather, the number of publications seems to have reached a ceiling with high variability. In 2011, the number of publications decreased from 85 to 64, but in the following year, it reached an all-time peak of 86 publications. After that, the number of publications decreased again, down to only 54 publications in 2014.

The classification scheme provided by the ISI Web of Science distinguishes 40 different research areas. The overall distribution of the 477 peer-reviewed publications distributed over these 40 research areas is shown in Figure 7. Each point without a label represents a peer-reviewed publication and each labeled point represents a research area. The size of a labeled point is proportional to the number of publications classified under this research area and corresponds to the number of connections to unlabeled points. In our data set, 15 research areas contained only one publication, 16 research areas contained two to six publications, and nine research areas had more than ten publications.

In Figure 7, most publications have been classified in the research area "Business Economics", indicated by the size of the point labeled "Business Economics", which is the largest. The second largest research area is "Engineering", followed by the research areas "Operation Research / Management Science" and "Computer Science". There are also connections between different research areas because some publications are related to more than one research area. The distance between labeled points indicates the degree of connectedness between research areas. For example, there are quite a lot of connections between the research areas "Operation Research / Management Science" and "Engineering" because the distance between the corresponding labeled points is closer than that between most of the other labeled points.

3.1.3. Results and Analysis

To answer Qu 3.1.1, we applied text mining techniques on titles and abstracts of the 477 selected publications from the years 2003 to 2014. The pre-processing of the available data included removal of stop words and removal of non-informative

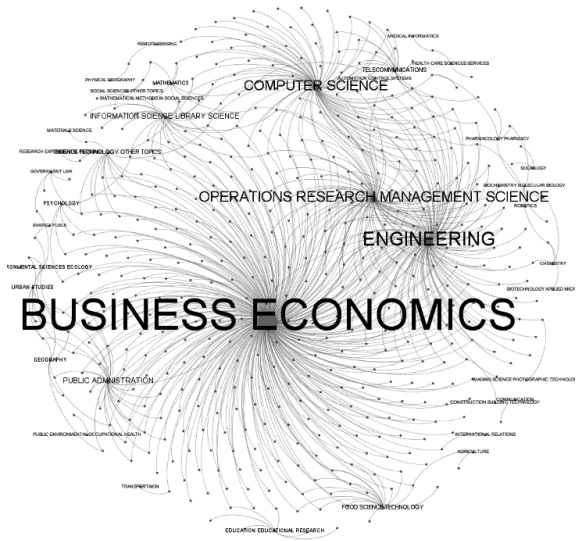


Figure 7. Distribution of publications across research areas using the ISI Web of Science classification scheme

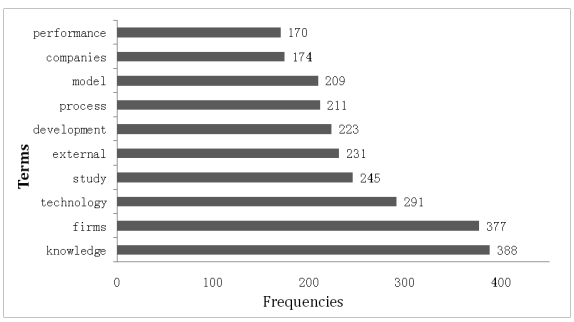


Figure 8. Frequencies of top 10 unique terms

words such as the search term "Open Innovation".

Text mining yielded 4913 unique terms (in the following, simply called "terms") from the 477 peer-reviewed publications. The top ten most frequent terms with their frequencies are shown in Figure 8. The most popular term is "knowledge" (appearing 388 times). We interpret the ten terms shown in Figure 8 as the descriptors of the concept of OI.

The growth rates of (a) publications and (b) terms from 2004 to 2014 are shown in Figure 9. Before 2011, the growth rates of publications and terms appear to be synchronized. From 2011 to 2014, the rate of publications continues to show a pattern of changing growth rates, while the change rates of terms seem to be more stable (-0.132, 0.021, -0.05, -0.047) and close to zero.

The observation that the change rate of terms became out of synch with the

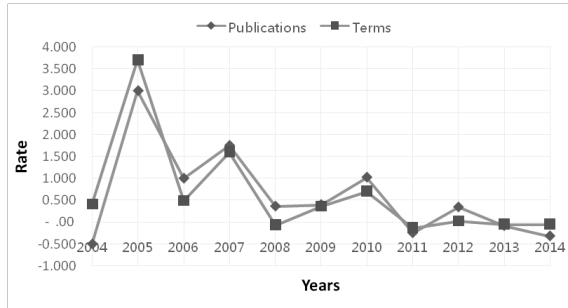


Figure 9. Growth rates of publications and terms

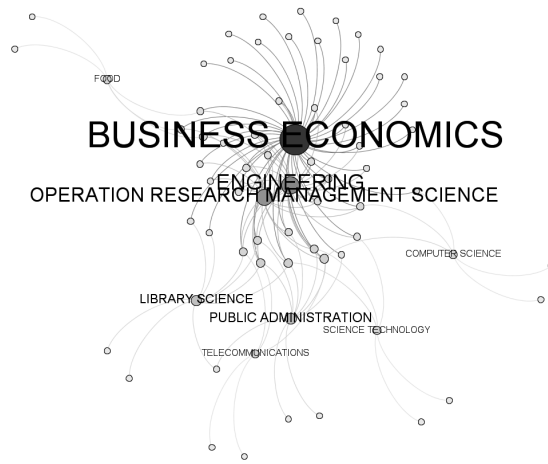


Figure 10. Connections between nine research areas based on text mining results

change rate of publications is quantified in Table 2, which shows the correlations[2] between the number of publications and the number of terms. The correlation coefficients shown in Table 2 are calculated based on the three most recent pairs of data. For example, the coefficient 0.92 in the row corresponding to the year 2005 uses the data on publications and termsfrom the years 2003 to 2005. It can be seen that the correlation is high (above 0.8) for all years except the first two years after 2011, where it drops to 0.57 and 0.22, respectively. Currently, we do not have an explanation for the singular behavior in the year 2011, where the number of publications dropped at a much higher rate than that of the number of terms.

We were also interested in how the research areas are connected based on terms. For that purpose, we chose the top 1% of the most frequent terms for each research area and drew the graph shown in Figure 10. Similar to the semantics in the graph of Figure 7, labeled points represent research areas, while unlabeled

[2] $Correl(X,Y) = \dfrac{\sum (x-\bar{x})(y-\bar{y})}{\sqrt{\sum (x-\bar{x})^2 \sum (y-\bar{y})^2}}$

points represent terms (and not publications, as in Figure 7). The graph in Figure 10 structurally resembles the one shown in Figure 7. This indicates that terms and publications structure the research areas in a similar way. The natural explanation for this observation is that when research areas share publications, they share the terms used in those publications to a similar degree.

To answer Qu 3.1.2, we focused on the nine research areas that had more than ten peer-reviewed publications. We again applied text mining to extract the five most frequent terms for each research area. The results are shown in Table 3.

Table 2. Correlation between number of publications and number of terms

Years	Number of Publications	Number of Terms	Correlation Coefficient
2003	2	37	-
2004	1	52	-
2005	4	245	0.92177105
2006	8	365	0.976732365
2007	22	950	0.998570958
2008	30	892	0.897151567
2009	42	1211	0.836385146
2010	85	2068	0.998411235
2011	64	1794	0.981660336
2012	86	1832	0.574248289
2013	79	1741	0.21996331
2014	54	1659	0.941399378

The gray cells contain terms that are also included in the top ten most frequent terms across all 40 research areas (cf. Figure 8). Three research areas, i.e., Computer Science, Information Science / Library Science, and Food, seem to have only one term among their top five most frequent terms that is also contained in the top ten most frequent terms across all research areas.

Table 3. Top five frequent terms in nine research areas

	Top1	Top2	Top3	Top4	Top5
Business Economics	firms	knowledge	technology	external	development
Engineering	technology	firms	external	knowledge	performance
Operations Research / Management Science	firms	technology	external	knowledge	performance
Computer Science	model	platform	based	web	framework
Public Ad-ministration	knowledge	firms	model	regional	development
Information Science / Library Science	knowledge	information	ideas	creation	source
Food	food	industry	case	companies	value
Telecom	model	firms	mobile	external	project
Science / Technology	interme-diaries	technology	process	public	analysis

Because the distinction of research fields based on the data presented in Table 3 is somewhat coarse-grained, we applied a more detailed quantitative analysis of the distribution of terms in the sets of publications of each research area. The

results of this analysis are shown in Table 4[3].

Table 4. Distributions of the results of the terms analysis

Research Area	n	S	S/n	σ_{Fi}	$\sum_{i=1}^{S} Fi$	AP
Business Economics	328	4314	13.15	12.9	22074	0.0156
Engineering	123	2435	19.80	7.19	8716	0.0291
Operations Research / Management Science	86	2108	24.51	6.11	6744	0.0372
Computer Science	81	444	5.48	0.95	644	0.0179
Public Ad-mini stration	30	1143	38.10	2.58	2479	0.0723
Information Science / Library Science	23	851	37.00	2.35	1652	0.0844
Food	17	453	26.65	1.41	735	0.0954
Telecom	16	492	30.75	1.24	803	0.102
Science / Technology	14	620	44.29	1.51	1026	0.1182

In Table 4, n denotes the number of publications in a research area, S denotes the number of terms in a research area, and S/n equals the average number of terms per publication in a research area. Sum(Fi) denotes the sum of term frequencies for a research area and σ_{Fi} denotes the standard deviation of term frequencies in a research area. Finally, AP denotes the average probability of a term to appear in a specific publication of a research field. The formula for AP is as follows:

$$AP = \frac{\sum_{i=1}^{s} Fi}{S * n}$$

One can see from Table 4 that there is a regular pattern: A higher number of publications goes hand in hand with a higher number of terms, a higher standard deviation of term frequencies, a lower number of (unique) terms per publication on average, and a lower probability for a term to appear in a specific publication. Only the research area "Computer Science" breaks this pattern. While Computer Science has the fourth-highest number of publications, it has the lowest values for S/n, σ_{Fi} and Sum(Fi), and almost the lowest AP value.

3.1.4. Discussion and Conclusion

In this section, we tried to answer two questions.

Regarding Qu 3.1.1, based on data retrieved from the ISI Web of Science, we found that there was a steady growth in the number of publications on OI across all research areas up to the year 2010. Beginning with 2011, a ceiling for the number of annual publications appears to have been reached, with strong

[3] n=Count of publications; S=Count of terms; Fi =Term frequencies per research area; σ =Standard Deviation; $\sum_{i=1}^{S} Fi = Sum(Fi)$

variations in subsequent years. Business Economics is the research area with the highest number of publications. We found that research areas were connected both via multi-classified publications and the top most frequent terms used in publications classified in different research areas.

Regarding Qu 3.1.2, based on a comparison of terms, term frequencies, and term distributions between research areas, we found that Computer Science has characteristics that make this area stand out from the other research areas, with more than ten publications on OI retrieved. The small number of unique terms (for the number of publications in the field) and the small frequencies of these terms seem to indicate that publications on OI in Computer Science are less diverse than in other fields. This could signal that the body of literature does not yet represent the potential that OI appears to offer in other research areas, or at least that the full potential has not yet been investigated and/or reported.

Based on the preliminary results regarding Qu 3.1.2, we believe that more research on OI in directions not yet addressed by the literature is possible. This motivated us to perform a more progressive study on OI in the field of Computer Science, specifically, on OI in RE.

3.2. OI in Software Requirements Engineering

In Section 3.2, we aim to summarize the body of knowledge about the use of OI in the field of RE. More specifically, we will analyze what uses of OI in the context of RE have been reported and how OI has contributed to individual steps of the RE process. We will also report on a mapping study we conducted on the literature provided in four scientific databases (ISI Web of Science, IEEE Xplore, ACM Digital Library, and Science Direct).

3.2.1. Introduction

The research presented above indicates that the use of OI in Computer Science is less diverse than in other fields. Figure 11 shows the percentage of peer-reviewed publications on OI in software engineering (SE) from 2003 to 2014 in contrast to the number of publications on OI in computer science (CS), based on data and classifications from the ISI Web of Science. We see from Figure 11 that the first article on OI both in SE and in CS was published in 2007. The percentage of OI publications in SE compared to those in CS is stable from 2007 to 2014 (11% to 20%), with the exception of 2008. In 2008, there was no OI publication in SE. These numbers clearly show that there is room for research on the use of OI in SE. In addition, based on these findings, we assume that RE as a subfield of SE must have even less research on OI. Therefore, we decided to survey existing research on how OI as a strategy is used in RE to see how OI contributes to RE. We found very few survey studies dealing with OI in the context of software engineering. Munir et al. [69] conducted a very comprehensive mapping study on OI in SE. In a small study, Lorenzi and Rossi [61] discussed the innovation

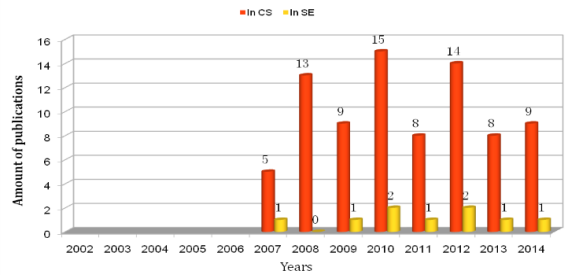


Figure 11. Peer reviewed publications on OI in CS and in SE

potential of open source software in the Italian software industry. However, none of these studies provides details about applying OI in subfields of SE such as testing, design, architecture, or RE.

Finally, Edison et al. conducted a study in which they compared the state of the art with the state of the practice of innovation measurement in the software industry [24]. Their study presents different types of definitions of innovation for the software industry but focused neither focus on OI nor on sub-fields of SE.

3.2.2. Systematic Mapping Study Process

We will now present the systematic mapping study process, including the formulation of the research questions, the approach to search string construction and data source selection, the steps taken to identify primary studies, and the method used to extract content that will help answer the research questions. We followed the guidelines on conducting mapping studies by Petersen et al. [80] and Kitchenham et al. [49].

Questions. Section 3.2 has three main goals. First, we want to get an overview of the body of published literature on OI specifically focusing on RE. Second, the RE process typically contains activities such as requirements elicitation, requirements specification, requirements analysis, requirements prioritization, requirements validation, and requirements management. We are interested in understanding whether certain OI-related concepts are used more or less often in the various requirements activities. Third, whenever OI ideas and concepts are used in a specific RE activity, we are interested in understanding what tool support exists and to what degree OI has been automated for that specific activity.

To achieve the research goals, we defined questions relevant to each goal.

Qu 3.2.1: How many and what types of studies on "OI in RE" have been published?

Qu 3.2.2: How is OI used in the RE process?

Qu 3.2.3: What is the degree of automation of proposed solutions on "OI in RE"?

Search Strings and Selected Database. We first identified control studies reporting on the application of OI strategies to RE. We then used the keywords

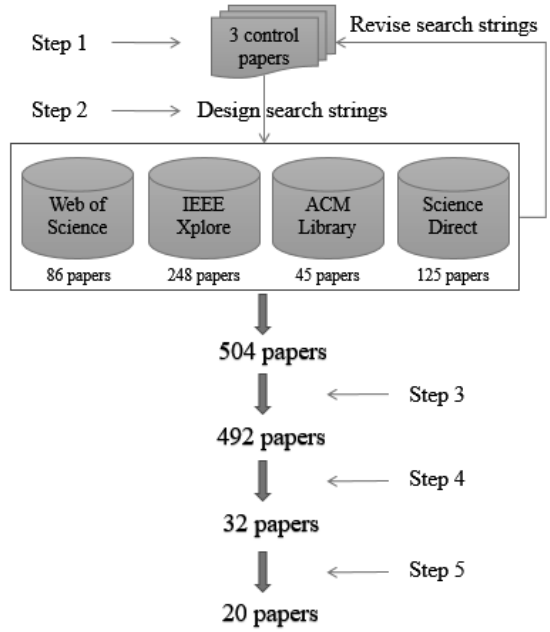


Figure 12. Identification steps

highlighted in the control studies to design the search strings. Next, we searched frequently used digital databases for relevant literature.

The principles for designing the search strings were:

a) The title of the publication should contain keywords related to OI.

b) Either the title, abstract, or author keywords should contain words related to RE.

Since the concept of OI was proposed in 2003, we only searched the time period from 2003 to 2016.

Identification of Studies. We performed five steps to identify relevant studies, as shown in Figure 12. To reduce the risk of excluding relevant studies on the one hand and including irrelevant studies on the other hand, we defined three groups of inclusion/exclusion criteria for each of the selection steps (step 3 to step 5), as suggested by Petersen and Ali [79]. The inclusion/exclusion criteria are shown in Table 5.

Step 1: We identified three control papers [52, 58, 63], and used the keywords highlighted in these control papers to design our search strings. The keywords derived from the three control papers are shown in Table 6.

Step 2: We followed the string design principle described in the previous section to design the initial search strings. Then we applied these search strings to four frequently used digital databases: ISI Web of Science, IEEE Xplore, ACM Digital Library, and Science Direct. Because the formats of the search string used by each database differ, we had to adjust the search strings to get reasonable results. After several rounds of revisions, we confirmed the search string for each

Table 5. Inclusion and exclusion criteria

		Criteria
For Step 3	Inclusion	Written in English Peer-reviewed papers Studies from 2003 to 2016 The study must be accessible in full text.
	Exclusion	Non-English papers Duplicate studies
For Step 4	Inclusion	Relates to Innovation in RE domain The studies pertaining to the scope of open source software used as OI examples
	Exclusion	Study of Innovation in non-software domain Research on OI, but does not relate to RE Studies about RE, but does not relate to OI
For Step 5	Inclusion	Studies with (expected) contribution for OI in RE
	Exclusion	Studies without (expected) contribution for OI in RE

Table 6. Keywords per related field

Related fields	Keywords
OI	Open innovation; OI; Open-innovation; Innovation; open source
RE	Requirement(s); Requirement(s) Engineering

database. The confirmed search strings are shown in Table 7. Next, we executed those strings on the digital databases and extracted 504 papers in total.

Table 7. Search strings

Database	Search strings
Web of Science	TI=("Open Innovation" OR open-innovation OR innovation OR OI OR "open source") AND (TI=("requirement*" OR "requirement* engineer*") OR TS="requirement* engineer*") Indexes=SCI-EXPANDED, SSCI, A&HCI, CPCI-S, CPCI-SSH Timespan=2003-2016
IEEE Xplore	((("Document Title":open innovation) OR (p_Title:OI) OR (p_Title:open-innovation) OR (p_Title:innovation) OR (p_Title:openness)) AND ((p_Abstract:requirements engineering) OR (p_Author_Terms:requirements engineering) OR (p_Abstract:requirement*) OR (p_Author_Terms:requirement*)))
ACM Digital Library	{ acmdlTitle:(+open +innovation OI innovation requirements engineering requirements requirement) AND recordAbstract:(requirements engineering requirement) AND keywords.author.keyword:(requirements engineering requirement) }
Science Direct	("open innovation" OR "OI" OR "open-innovation" OR innovation OR "openness") AND ("requirements engineering" OR "requirement engineering")

Step 3: We applied the inclusion/exclusion criteria set for step 3 to filter the 504 papers found in step 2. We found 12 duplicate papers, which were excluded.

Step 4: From the remaining 492 papers, we removed 460 papers that were found to be not relevant after checking their title, abstract, and keywords and applying the inclusion/exclusion criteria set for step 4. We removed those papers as being not relevant to OI in RE even though OI and RE were keywords found in the title or abstract. For example, one paper has OI in its title but presents "water requirements" in the abstract.

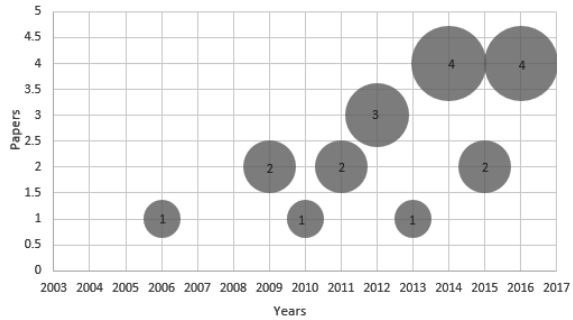


Figure 13. Time distribution of studies

Step 5: Then we used the inclusion/exclusion criteria for step 5 to analyze the exclusion criteria for step 5 to analyze the content of each of the remaining 32 papers, and identified the papers pertaining to the scope of our study. At the end, 20 papers were found to be relevant, including the three control papers.

Data Extraction. To answer questions Qu 3.2.1 to Qu 3.2.3, we extracted the relevant data from the set of identified papers. The properties of the extracted data and the relationship between the data and the Qus are shown in Table 8.

Table 8. Data extraction properties mapped to research question

Category	Properties	Qus
General information	Title, Year of paper, Abstract	Qu 3.2.1, Qu 3.2.2
Study type	Proposal solution, Evaluation, Validation, etc.	Qu 3.2.1
Research methods	Case study, Survey, etc.	Qu 3.2.1
Research problem	Subject, Research questions	Qu 3.2.1, Qu 3.2.2, Qu 3.2.3
Outcomes	Affected steps of the RE process	Qu 3.2.2, Qu 3.2.3

3.2.3. Results and Analysis

In this section, the results of our mapping study will be reported in three parts according to the Qus. First, we will report the overview information we extracted regarding time distribution, research method, and type of study. Then we will focus on the content of the papers to sum up the existing contributions of OI in RE.

Overview Information. To answer Qu 3.2.1, we analyzed the general information contained in the 20 identified papers, i.e., the year of publication, the research method used, and the type of study. Figure 13 shows the time distribution of the identified papers based on their publication years. We see from Figure 13 that during the period from 2003 to 2005, there was no identified paper. During the period from 2006 to 2016, only one paper each was published in 2006, 2010, and 2013, two papers each were published in 2009, 2011, and 2015, three papers were published in 2012, and four papers each appeared in 2014 and in 2016.

Next, we classified the selected papers according to the type of study and the research method used. According to Wieringa et al., the main study types are "a new solution, evaluation research, validation research, and opinion research."

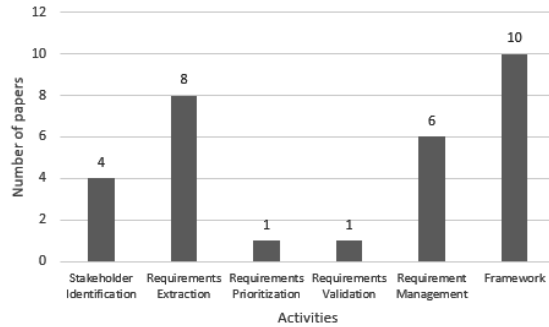


Figure 14. Classified activities of RE that OI may affect

[101]. To classify the research methods used in the selected papers, we used the categories established by Creswell in 2003 [18]: qualitative approach (i.e., case study, phenomenological research, etc.), and mixed approach (i.e., transformative research, framework). The classification is shown in Table 9. We see from Table 9 that there are nine evaluation studies among the selected papers. The most frequently used research method is transformative research, with seven papers in this category.

Table 9. Classification based on the type of study and the research methods used

		Type of Study				
		Solution	Evaluation	Validation	Opinion	Total
Research Methods	Experiments	R_14 [76]	0	R_8 [45]	0	2
	Survey	0	R_6 [53]; R_12 [50]; R_13 [7]	R_16 [59]	0	4
	Case study	R_19 [86]	R_9 [68]; R_20 [44]	R_5 [103]; R_18 [72]	R_4 [66]	6
	Phenomenological	0	R_15 [52]	0	0	1
	Transformative	R_1 [31]; R_2 [63]; R_10 [82]	R_7 [26]; R_11 [58]; R_17 [34]	R_3 [83]	0	7
	Total	5	9	5	1	

OI Used in the RE Process. This section answers Qu 3.2.2. The RE process is organized as a set of activities aimed at transforming input into output [14, 62]. Among the selected papers, we found that the following RE activities were addressed: stakeholder identification, requirements extraction, requirements prioritization, requirements validation, and requirements management. In addition, some papers addressed the role of OI in the RE framework as a whole, including all activities of the RE process.

Table 10 shows to which RE activities OI was applied in the set of selected papers. The frequency with which RE activities or the RE framework are addressed is shown in Figure 14. The numbers add up to more than 20 because some papers describe the use of OI applied to several activities. Papers R_8, R_10, R_11, R_16, and R_19 discuss how OI is used in the RE framework as a whole as well as in select RE activities. In 50% of the papers, OI was applied to the RE framework as a whole. The most frequent activities to which OI was applied are requirements extraction (8 occurrences), followed by RE management (6 occurrences) and stakeholder identification (4 occurrences). The application of

OI in the context of requirements prioritization and requirements validation was only mentioned once for each activity.

Table 10. Classification based on OI used in the RE process

Reference No.	Stakeholder Identification	Requirements Extraction	Requirements Prioritization	Requirements Validation	Requirements Management	Framework
R_1	x					
R_2						x
R_3						x
R_4	x					
R_5					x	
R_6					x	
R_7		x				
R_8		x				x
R_9						x
R_10		x				x
R_11	x		x		x	x
R_12					x	
R_13		x				
R_14	x					
R_15						x
R_16		x			x	x
R_17						x
R_18		x				
R_19		x		x	x	x
R_20		x				

Stakeholder Identification. There are four papers related to stakeholders. Two papers (R_1, R_4) propose new approaches for identifying and discovering new stakeholders. One paper (R_11) presents a research agenda and a framework "along with framing model to help researchers frame and break down their research questions", including identifying stakeholders. By proposing a web-based process, another paper (R_14) attempts to enhance stakeholders' contributions in the software engineering process.

Requirements Extraction. Eight papers connect OI and requirements extraction. Four papers (R_7, R_8, R_18, R_20) are related to extraction techniques. Paper R_7 verifies a method for obtaining new ideas based on semantic recognition technology to support innovation in RE. Paper R_8 provides a framework that can "support pre-clustering and evolvement of open innovation input before transfer it into the company". Paper R_18 presents an approach for an automated requirements elicitation process using the requirements and ideas from end users discovered in open source software (OSS) communities. Paper R_20 proposes a process based on customer requirements collected from previous products and problem analyses to extract new software requirements. The four remaining papers (R_10, R_13, R_16, R_19) focus on the impact of an OI strategy on requirements extraction. In Paper R_10, a framework is designed for assessing the innovation capabilities of development teams in the early stages of RE, especially with regard to a new features-design process. In Paper R_13, a research plan is designed to verify whether the identification and acquisition of requirements

meet the "Twin Peaks" model in an open source development environment. Paper R_16 proposes a model (RAMBO) focusing on the interaction and overlap between the internal RE process of the focal firm and its connected OSS ecosystem, to better manage the challenges implied by OI. Paper R_19 validates the impact of "informalist" requirements extraction, as part of a RE framework, on software development in an open source software development environment.

Requirements Prioritization. There is only one paper (R_11) that has relevant content regarding the contribution of OI to the activity of requirements prioritization. This paper provides a research agenda that guides researchers on how to frame and break down research questions related to requirements prioritization, considering the different angles implied by the OI model.

Requirements Validation. There is one paper (R_19) that mentions the validation of requirements with the help of OI as part of an overall RE framework.

Requirements Management. Two papers (R_5, R_12) refer to the challenges and risks of sharing information, including sharing requirements with partners in the context of OI. Paper R_5 highlights that "managing requirements in an open innovation context is challenging as requirements are freely available for several potentially contributing companies." Paper R_12 analyzes what possible challenges the RE process would face when using OI. For instance, the authors found that managing context and mapping requirements to actors are two highly interconnected and challenging RE tasks.

Based on the interoperability between innovation and requirements management, paper R_6 proposes the L model. The L model improves the quality of software development by improving the quality and speed of innovation within complex systems.

RE Framework. Ten papers relate OI to RE frameworks (R_2, R_3, R_8, R_9, R_10, R_11, R_15, R_16, R_17, and R_19). There are two papers (R_15, R_17) that discuss the differences in RE practice and the RE process between the development environment of OSS and that of closed source software. Four papers (R_2, R_3, R_9, and R_19) introduce the new RE frameworks or processes that support innovation.

One paper (R_2) demonstrates an RE approach to support the optimization of innovation (TI) assessment. Another paper (R_3) emphasizes the need to consider the business analytics role of RE in Product Lifecycle Management (PLM) when high PLM data is to be turned into a successful market-oriented innovation management strategy. The paper (R_9) proposes a process that integrates RE with innovation. The paper (R_19) validates the impact of "informalist" software RE in an open source software development environment, including requirements extraction, validation, and management.

Automation of Proposed Solutions on "OI in RE". This section answers Qu 3.2.3. We found only one paper (R_18) that presents an automated approach.

The authors of paper R_18 present an approach for the identification of an automated requirements elicitation process in OSS communities. The goal of this

paper is to address the relationship between the role of end users and the influence of the development processes in OSS. This paper can be seen as a preliminary guideline for the organization of community-oriented software development.

3.2.4. Discussion

Regarding our three research questions, our findings can be summarized as follows.

Regarding Qu 3.2.1, we found only 20 papers that connect OI strategy and RE. Among these 20 papers, OI strategy in RE was limited to the context of OSS development (papers R_5, R_13, R_14, R_15, R_16, R_17, R_19). More than half of the selected primary studies contained evaluation or validation. Only one paper (R_4) was purely an opinion paper. This indicates that published research on OI in the context of RE is of practical value for industrial software development. More than half of the selected primary studies include an evaluation or validation. Only one paper (R_4) is purely an opinion paper. This indicates that published research on OI in the context of RE is of practical value for industrial software development.

Regarding Qu 3.2.2 and Qu 3.2.3, we found that there appears to be a lack of research on the use of OI for requirements prioritization and requirements validation, as there was only one paper each dealing with these topics, i.e., papers R_11 and R_19, respectively. In addition, we found only one paper (paper R_18, dealing with OI in the context of requirements extraction) that presented a solution approach with tool support. This indicates that there is little automation support mentioned in the literature on the use of OI strategies in RE.

We are aware that regarding the use of OI strategies in the context of RE, much more research is ongoing than what has been published in research papers. For example, the field of OSS development has been studied well and different business models combining open and closed development have been developed. Similarly, crowdsourcing is emerging as another way to open up traditionally closed work settings. However, our study, as limited as it may be, provides some indication about facets of OI in RE that have been researched to a greater or lesser extent. For example, we can say that, similar to many other methods applied in SE (and specifically in RE), OI could be used in two flavors, either in the form of an intrusive approach or in the form of a non-intrusive approach. Our study results suggest that currently, the way that companies implement OI is normally intrusive. For example, the way a company reaches out to the OSS community in order to inject innovation into their RE processes is usually to participate in OSS projects with their own human resources. Alternatively, companies may use surveys and interviews as instruments to extract innovative impulses from their user base. Both approaches are highly intrusive and effort-intensive, either on the company's or the users' side (or both). The upside of using this approach is that engineers establish a relationship with their user community and thus may

get much first-hand, high-quality input. The downside of intrusiveness is that end users might feel disturbed. Thus, not only is the intrusive approach costly, but it might also induce self-selection bias and researcher bias, as well as other negative effects.

3.2.5. Conclusion

The application of OI strategies in RE has not been researched much to date, especially with regard to using OI in the prioritization and validation steps of the RE process.

Summarizing, we found that 20 primary studies found in the period 2003-2016 report on results regarding the application of OI in RE. Half of these studies report on the application of OI in RE as a whole. Only one paper each was found to be related to requirements prioritization and validation. None of the primary studies present proprietary tool support for OI in RE. Only one study presents a method for automatic requirements extraction in OSS projects that can be implemented using standard machine learning tools.

Acknowledging the lack of published research on the use of OI strategies in specific RE activities, i.e., prioritization and validation, as well as the lack of reported tool support, we see new opportunities for research on automated and low-cost methods for applying OI strategies in RE.

3.2.6. Threats to Validity

Like all empirical studies, our mapping study has limitations. In the following, we will describe the threats to construct, internal, and external validity that we faced.

Construct Validity. In this mapping study, the greatest possible threat to construct validity is the co-existence of multiple definitions of OI. In order to control this threat, we decided to use the concept of OI as proposed and defined by Chesbrough in 2004 [17].

Internal Validity. We faced several threats to internal validity caused by the subjectivity imposed during several steps of the paper selection and analysis process.

First, the choice of the data sources from which to extract the primary studies is subjective. We tried to control this threat by choosing a standard set of digital databases typically used in software engineering survey studies.

Second, the construction of the search strings might have been inappropriate and might have created either too many false positives, imposing an inordinately high workload on the involved researchers for filtering these out in later steps of the selection process, or simply missed relevant publications. We tried to control this threat by identifying three control papers that were clearly within the scope of our study. We used the keywords of these papers to construct our search string. We also checked whether the control papers were retrieved when we applied the search string and whether they were contained in the final set of primary studies

after all filtering steps had been performed.

Third, all papers selected in the first step of the search, when the search string was applied, had to be filtered using the step-by-step approach described in Section 3.2.2. Each of these steps had to be conducted manually and thus was subject to the threat of subjective bias. We tried to control this threat by using previously defined inclusion/exclusion criteria in each selection step.

External Validity. The 20 papers identified in the mapping study are related to the field of software engineering, and thus the findings are restricted to that field. Since we retrieved the papers from data sources that typically archive research publications, we might have missed much of what is going on in the software industry with regard to OI and RE simply because it is never published. On the other hand, since many of the selected papers report on industry case studies, we believe that our findings are not completely irrelevant for software industry but are indeed beneficial for both industry and academic research.

3.3. Summary

Our analysis of the state of the art was conducted on two levels of abstraction. On the first level, we first compared the number of non-professional articles (searched by Google Scholar) with that of peer-reviewed articles (searched by the ISI Web of Science). We found that the number of non-professional articles was far higher than that of peer- reviewed articles in research databases. We assumed that this means that people are interested in OI, but that relevant research is not abundant. Next, we studied the distribution of OI research articles in the ISI Web of Science database. We found that in the field of computer science, especially in the SE field, the number of OI research publications were lower than that in other fields. When analyzing peer-reviewed literature on OI, we observed that the field of computer science seems to have significantly less diversity than all other fields with more than ten publications indexed in the ISI Core Collection. Our preliminary interpretation of this observation is that the topic of 'Open Innovation' has not yet been researched and discussed in a depth and breadth comparable to other fields, in particular Business Economics, Engineering, and Operations Research Management Science.

On the second level, we conducted further research about OI in RE. We chose four commonly used research databases as data sources and completed a mapping study. We found 20 primary studies from the period 2003-2016 that report on results regarding the application of OI in RE. Half of the studies report on the application of OI in RE as a whole. Only one paper each is related to requirements prioritization and validation. None of the primary studies present proprietary tool support for OI in RE. Only one study presents a method for automatic requirements extraction in OSS projects that can be implemented using standard machine learning tools. In summary, the application of OI strategies in RE has not been studied much, especially with regard to using OI in the prioritization and

validation steps of the RE process. In addition, there is a lack of reported tool support.

Based on this analysis, we saw an opportunity to develop automated and low-cost methods for applying OI strategies in RE. As discussed above, implementing OI by using online open sources (data and/or people) has been used for the purpose of requirements elicitation and prioritization. However, OI in RE has not yet been used for implementing more complex ideas about the relationship between requirements and the value of a software product, like those suggested by Noriaki Kano. This motivated us to design an automatic classification method based on the traditional Kano model [108] theory, the OIRE method. In the next chapter, we will present the OIRE method in detail.

4. OIRE METHOD

In this chapter, we will introduce the components of the OIRE method and how we designed each component.

The components of the OIRE method and the order in which they are typically used are shown in Figure 15. A user provides input data collected and preprocessed from the Internet, e.g., from online reviews. The input data must correspond to self-contained English sentences (lines of text) that relate to product features. As shown in Figure 15, the OIRE method comprises four components, i.e., Sentence Classification, Sentiment Mining, Kano-like Processing, and Visualization. In the following, we will use examples to illustrate the functioning of each component of the OIRE method.

Figure 16 shows how the input data must be organized. A product may have several features. All text lines relating to the same feature are stored in one input file. The opinions of reviewers about features of a product are captured by a set of input files, each file corresponding to one feature and containing all text lines related to this feature.

This chapter is structured as follows. In Section 4.1, we will outline the algorithm design and validation of Component 1 and in Section 4.2, that of Component 2. In Section 4.3, we will introduce the algorithm design and validation of Component 3. In Section 4.4, we will show the outputs of Component 4. We will use application examples to verify the applicability of the algorithms designed for Component 1 and Component 2. For Component 3, we will use simulations to verify the availability of the algorithms. In Section 4.5, we will show the work process of implementing the OIRE method.

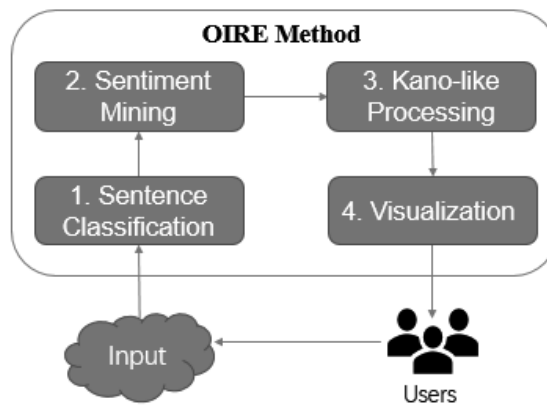


Figure 15. The composition of the OIRE method

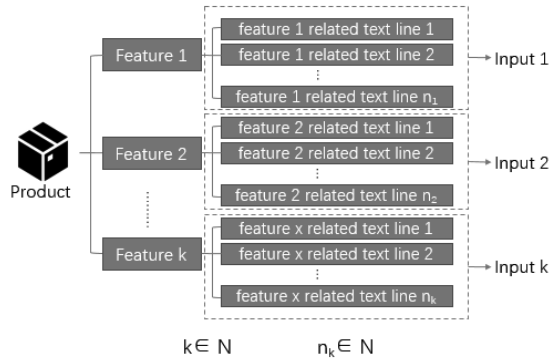


Figure 16. Expected inputs for the OIRE method

Input	Output
Feature A Related Text	**Functional**
I am so sad that A is deleted	A is an awful function.
I don see the reason to delete A	I like A very much
A is an awful function.	It is interesting to use A
A is removed, wonderful	If A is added, I would be happy
I like A very much	We want to ignore A
I like this app if it removes A	I don know why there is A
It is interesting to use A	**Dysfunctional**
I don't care whether A is deleted	I am so sad that A is deleted
If A is added, I would be happy	I don't want you to delete A
If A is removed, I feel happy	A is removed, wonderful
We want to ignore A	I like this app if it removes A
I don know why there is A	I don't care whether A is deleted
	If A is removed, I would be fine

Figure 17. Example input and expected output of Component 1

4.1. Component 1 - Sentence Classification

Since the input data of this step already contains only feature-related text, in this process we use machine learning methods to classify the text lines of the input into two classes, i.e., "functional" and "dysfunctional". The unit of analysis is one line of text. Text classified as functional corresponds to text lines stating the presence of a feature, while text classified as dysfunctional corresponds to text stating the absence of a feature [109]. Figure 17 shows the example input and the expected output of Component 1. Assume that we have an input file that has twelve text lines related to Feature A of a product. We realize that users of the OIRE method might not be familiar with the various machine learning methods. Therefore, in Component 1, we provide ready-to-use trained models.

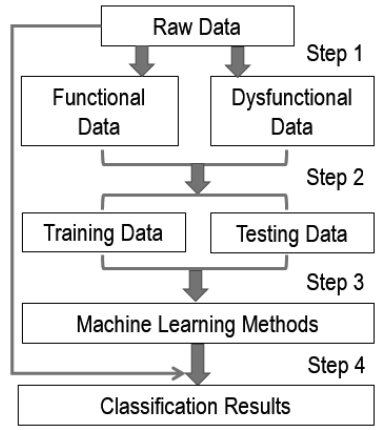


Figure 18. The process of Component 1

4.1.1. Algorithm Design

The process of classifying feature-related text lines into the functional and dysfunctional dimensions of the traditional Kano model is shown in Figure 18. We apply supervised machine learning methods in this step. The process consists of four steps.

Step 1: To be able to select a suitable machine learning method for our classification task, we must analyze the performance of candidate machine learning methods. To do so, we need to label training and test data sets.

Our input is unlabeled, which means that if we cannot find suitable existing labeled data sets for our purpose, we must create such data sets manually. We can do this, for example, by taking a subset of the input data set and analyzing each text line relating to a feature as follows: If a text line contains words of affirmation or complaint about using/having a feature, it means the feature exists or is imagined as being present, and thus we label the related text line as belonging to the category “functional”. If a text line contains words that express affirmation or complaint about the lack of a feature, it means the related feature does not exist, and thus we label the related text line as belonging to the category “dysfunctional”. However, when the subjunctive mood is used, we classify the text line according to the imagined or desired part of the text line. For example, when someone says, "I would be happy if you could add this feature", even though this means the feature is still missing, the text line expresses the feeling a person has when they imagine that the feature exists. Hence, we classify this text line as belonging to the category “functional”.

The total size of the labeled data set should be large enough to facilitate proper training and small enough not to be time-consuming. We suggest a data set size of 100-250 labeled text lines. Table 11 shows some example text lines that we labeled as either functional or dysfunctional.

Step 2: We split the set of labeled text lines into a test and a training data set.

Table 11. Examples of labeled text lines

	Examples
Functional	The plugin function is good.
	I have found that this method works.
	I use "checkstyle" to analyze my code.
	I cannot figure out why people want this button here.
	I would be happy if you could add this feature.
Dysfunctional	I miss the hierarchical (frame-based) view.
	The link is not available now.
	Basically, I'd like to avoid to do this in a js file.
	So far, I've had no luck to use the Bespin one in Pydev.
	I would be sad if this feature was deleted.

The ratio of the lines of text of the training data set to those of the test data set is 80:20.

Step 3: We used the training data set to train the classification models and the test data set to check the accuracy of the output of the models. We selected several supervised machine learning methods for comparison, e.g., Naïve Bayes, MaxEnt, Decision Trees, Support Vector Machines (SVM), and others.

We use a confusion matrix [93] to calculate the performance of each method. Table 12 shows the confusion matrix for our classification problem. Based on the predicted results of each model, we calculated the accuracy [1] of each trained classification model as well as FPV[2] (Functional predictive value), which shows the proportion of the functional text lines that are predicted correctly, and DPV [3] (Dysfunctional predictive value), which shows the proportion of the dysfunctional text lines predicted correctly.

Step 4: We selected the best-performing classification model for our original input and classified each text line as either functional or dysfunctional.

Table 12. The confusion matrix used to assess the performance of supervised machine learning methods

		Predicted Condition	
		Functional	Dysfunctional
True Condition	Functional	True Functional (TF)	False Functional (FF)
	Dysfunctional	False Dysfunctional (FD)	True Dysfunctional (TD)

4.1.2. Application Example for Component 1

Next, we will demonstrate the applicability of the algorithm of Component 1 with an example. We applied machine learning methods to determine whether a text

[1] $Accuracy = (TF + TD) / (TF + FF + FD + TD)$
[2] $FPV = TF / (TF + FF)$
[3] $DPV = TD / (FD + TD)$

line extracted from an online open source corresponds to an answer of the functional or dysfunctional question asked in the traditional Kano model.

Input: We used a data set containing 1,493 lines of text from an app store as input to our method. This data set was derived from an original set of 92,217 reviews, which was cleaned in order to make it appropriate for further processing. In addition, we removed stop words (e.g., "a", "an", "the", etc.) as well as punctuation and strange symbols. We also removed text lines containing less than 20 words because we think that long text lines may be able to comprehensively express a reviewer's real thoughts.

Approach: In order to classify the input text lines and transform them into input for Kano-like models, we first created a "functional-dysfunctional" corpus for use as the training and test data sets for selecting machine learning methods. We followed the rules described in Section 4.1.1. We needed approximately four person-hours of effort to manually label 250 lines of text. The split between text lines labeled as functional and dysfunctional was 50:50.

We compared five frequently used machine learning methods, i.e., Naïve Bayes, MaxEnt, Decision Trees, Random Forest, and SVM. We used the confusion matrix (cf. Table 12) to calculate the performance of each method. To check whether the proposed ratio of 80:20 between the training and the test data set really gets the best results, we varied the size of the training data set in the range from 50 to 200 with a fixed test data set of size 50. Thus, we checked for the ratios 50:50 (training data set size = 50), 66:34 (training data set size = 100), and 80:20 (training data set size = 200). Table 13 shows the results of the experiment.

We see from Table 13 that the Naive Bayes method achieved the highest average FPV (90%), while the SVM method achieved the highest average DPV (76.7%) and the highest average overall accuracy value (64.2%). For the training data set containing 200 text lines, the methods MaxEnt and SVM had the highest accuracy (65%) and the closest FPV and DPV values. In addition, the standard deviations of the accuracy values of the MaxEnt and SVM methods (0.006 and 0.014, respectively) from the three tests are very small. This suggests that the performance of the MaxEnt and SVM methods is stable and no further improvement can be expected for larger training data sets. However, although the accuracy values are the highest in Table 14 when the MaxEnt and SVM methods were used, the absolute values (65% for both methods) are not very high. Thus, to further improve accuracy, we decided to use the training data set of size 200, then implement both the MaxEnt and the SVM methods together, and then only keep those cases where the predictions of the two methods were consistent. According to the experiment results, we found that the two methods yielded the same classifications for 44 out of 50 text lines in the test data sets. A further analysis showed that 15 out of 22 text lines were accurately classified as functional (FPV = 68%) and 17 out of 22 text lines were accurately predicted as dysfunctional (DPV = 77%). This means that the overall accuracy increased from 68% to 73%. Hence, we decided to use MaxEnt together with SVM in our application example and only keep those

cases where the predictions of the two methods were consistent.

Table 13. Experiment results

Methods	Indicator	Size of Training Set (number of text lines)			Average Value	Standard Deviation (σ)
		50	100	200		
Naive Bayes	FPV	100%	100%	70%	90%	0.173
	DPV	0	0	45%	15%	0.260
	Accuracy	50%	50%	57.5%	52.5%	0.043
MaxEnt	FPV	45%	40%	60%	48.3%	0.161
	DPV	75%	80%	70%	73.3%	0.076
	Accuracy	60%	60%	65%	61%	0.006
Decision Trees	FPV	50%	40%	40%	43.3%	0.058
	DPV	55%	80%	80%	71.7%	0.144
	Accuracy	52.5%	60%	60%	57.5%	0.043
Random Forest	FPV	60%	70%	60%	63.3%	0.058
	DPV	35%	50%	45%	43.3%	0.076
	Accuracy	47.5%	60%	52.5%	53.3%	0.063
SVM	FPV	45%	45%	65%	51.7%	0.115
	DPV	80%	85%	65%	76.7%	0.104
	Accuracy	62.5%	65%	65%	64.2%	0.014

Table 14. Prediction accuracy of Component 1

	Lines of Texts	Proportion	Samples	Correct Classification
Functional	628	55%	126	116
Dysfunctional	523	45%	105	61
Total	1151	100%	231	173
FPV	92%			
DPV	58%			
Accuracy	75%			

Result: After implementing the classification method in Component 1, we found that 1,151 out of 1,493 lines of text were classified into the same categories when using both SVM and MaxEnt. 628 lines of text were classified as functional and 523 were classified as dysfunctional.

To estimate the actual classification accuracy of Component 1, we used the Probability Proportional to Size (PPS) method [89].We randomly chose 20% of the total number of classified text lines and then manually checked the correctness of the classification of the text lines contained in this sample. The results of this performance check are shown in Table 14. We see that the overall accuracy of Component 1 is 75%. The accuracy of classifying text lines into the category "functional" is very high (FPV = 92%), while the accuracy of classifying text

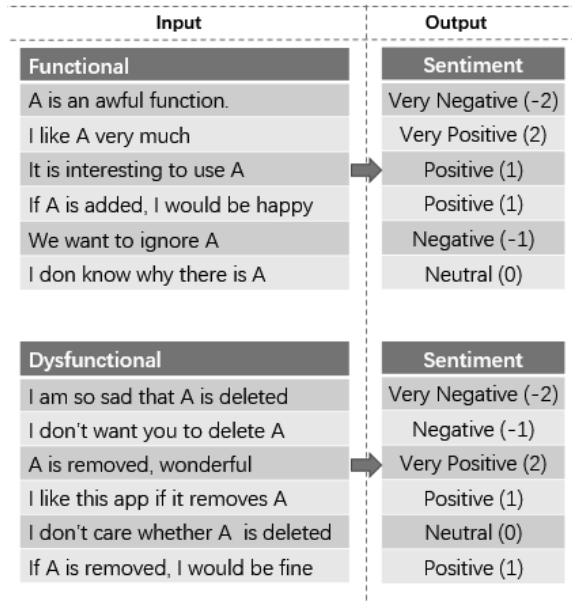


Figure 19. Example input and expected output of Component 2

lines into the category “dysfunctional” (DPV = 58%) is relatively low.

4.2. Component 2 - Sentiment Mining

In this section, we will first present the dictionary-based method [108] we designed to calculate the sentiment score of each text line in each of the two classes, functional and dysfunctional (output from Component 1). Next, we will classify the polarity of the sentiment (from Very Negative to Very Positive) of each text line according to its sentiment score and translate it into the corresponding Kano score. For example, the sentiment "Very Negative" corresponds to a Kano score of "-2" and the sentiment "Very Positive" corresponds to a Kano score of "+2". Figure 19 shows the example input and the expected output of Component 2.

4.2.1. Algorithm Design

The purpose of this process is to conduct opinion mining to attach sentiment scores to each of the labeled text lines. According to Reagan et al.’s study [81], sentiment detection methods can be one of the following types:

- Dictionary-based methods [51],
- Supervised learning methods [20],
- Unsupervised / Deep learning methods [90].

We want to classify text lines into five categories, i.e., Very Positive, Positive, Neutral, Negative, and Very Negative. To do so, multiclass instead of binary classification methods are needed. However, multiclass classification is more intricate

than solving binary classification problems [4]. In other words, using supervised machine learning methods is costlier. Since it is easy to implement, we decided to design a dictionary-based method to classify the polarity of sentiments contained in each of the labeled text lines received from Component 1 of the OIRE method.

Step 1: We first created a special emotion dictionary consisting of seven corpuses, i.e., containing "Positive Emotional Words" (PEW), "Very Positive Emotional Words" (VEPW), "Negative Emotional Words" (NEW), "Very Negative Emotional Words" (VNEW), "Adversative Words" (AW), "Negative Words" (NW), and "Intense Words" (IW), respectively.

The PEW and NEW corpuses of our emotion dictionary were created based on a sentiment dictionary consisting of two files provided by Minqing Hu and Bing Liu [42]. The two files contain 2,041 positive words (file: positive-words.txt) and 4,818 negative words (file: negative-words.txt), respectively. The IW corpus refers to the intense corpus file of the HowNet sentiment dictionary collected by Qiang Dong and Zhendong Dong [23]. The IW includes 71 words. To create VPEW, VNEW, AW, and NW, we used the world's largest and most trusted free online synonyms dictionary, Thesaurus.com. We first identified a keyword, such as the word "amazing", as a keyword of VPEW, or the word "awful" as a keyword of VNEW. Then we searched for the synonyms of this keyword. Next, we manually checked all the synonyms suggested by the synonym dictionary. This is how we finally got our word lists for the VPEW, VNEW, AW, and NW corpuses. We also removed those very positive words listed in VPEW from "positive-words.txt" (PEW) and those very negative emotional words listed in VNEW from "negative-words.txt" (NEW). Table 15 shows the total numbers of words and example words of each corpus of the emotion dictionary.

Table 15. Emotion dictionary with example words

	Corpus	No. of Words	Examples
Emotion Dictionary	PEW	2013	like, good, well, accept
	NEW	4794	bad, sad, cannot, delete
	VPEW	28	amazing, love, brilliant
	VNEW	24	awful, worst, terrible
	IW	71	very, much, extremely
	AW	9	but, however
	NW	18	no, not, never, aren't

Step 2: We split each labeled text line into words and searched for each word in the emotion dictionary in order to identify its sentiment polarity. When one word was confirmed as being included in one corpus of our emotion dictionary, we assigned a sentiment score to this word according to the different corpuses to which it belongs. For example, if a word was included in VPEW or in VNEW, the sentiment score of this word was 100 or -100. If this word was included in PEW or in NEW, the sentiment score of this word was 1 or -1.

```
set SentimentScore = 0;
for each sentence in text:
        for each word in sentence:
                if word in PEW:
                    SentimentScore = (SentimentScore + 1)
                if word in VPEW:
                    SentimentScore = (SentimentScore + 100)
                if word in NEW:
                    SentimentScore = (SentimentScore – 1)
                if word in VNEW:
                    SentimentScore = (SentimentScore – 100
                if word in AW:
                    SentimentScore = SentimentScore * 0
                if word in NW:
                    SentimentScore = SentimentScore * (– 1)
                If word in IW:
                    SentimentScore = SentimentScore * 100;
        end for
end for
```

Figure 20. The schematic process of calculating the sentiment score in the dictionary-based method

Step 3: We calculated the total sentiment score of each text line and transferred it to the Kano score (ranging from -2 to 2, i.e., from Very Negative to Very Positive), which constitutes the input needed for Kano-like models. In Figure 20, a simplified algorithm is used to show the schematic process for automatically calculating the sentiment score when using our dictionary-based method. The real script is implemented in the R language with over 100 lines of code.

Instead of classifying sentiments into three categories, i.e., positive, negative, and neutral, like other researchers have done, we need to classify functional and dysfunctional text lines into five categories, i.e., Very Positive (sentiment score >= 100), Positive (0 < sentiment score < 100), Neutral (sentiment score = 0), Negative (-100 < sentiment score < 0), and Very Negative (sentiment score < = -100).

Table 16[4] shows examples of how to calculate the sentiment score and the Kano score of text lines using the algorithm presented in Figure 20. The words in different colors indicate the reference to different emotion corpuses as shown in Table 15. For example, words in red refer to the PEW corpus.

Table 16. Examples of calculating the sentiment and Kano scores for text lines

Text Line	Sentiment Score	Polarity	Kano Score
i like (a) this function very much (b)	10000	Very positive	2
why cannot (c) you delete (c) this function	-2	Negative	-1
i hate (d) this feature	-100	Very Negative	-2
this feature is not (e) bad (c)	1	Positive	1
the software allows user to open files automatically	0	Neural	0

4.2.2. Application Example for Component 2

Next, we will demonstrate the applicability of the algorithm of Component 2 with an example.

[4] a = words from PEW, b = words from IW, c = words from NEW, d = words from VNEW, e = words from NW.

Input: The output of the application example of Component 1 (cf. Section 4.1.2), which contained two categorized files. One file has 628 lines of functional text, and the other file has 523 lines of dysfunctional text. These two files were used as the input of Component 2. In the remainder of this section, we will use "functional input" and "dysfunctional input" when referring to these two files.

Approach: We ran the dictionary-based method as described in Section 4.2.1 to calculate the sentiment and Kano scores of each line of text in the functional and dysfunctional inputs separately. Then we used the PPS method again to check the performance for 20% of the text lines contained in each input. To be able to manually check the emotions expressed in the sampled text lines and thus verify the accuracy of the classifications, we used the guidelines presented in Table 17.

Result: After implementing Component 2, we classified the text lines into five sentiment categories. The classification details as well as the corresponding degree of accuracy for the samples drawn from each class are presented in Table 18. We see from Table 18 that the highest accuracy value (93%) was achieved for the category Very Positive. 118 out of 127 text lines were predicted accurately. The lowest accuracy value (58%) was achieved for the category Positive, closely followed by the category Very Negative (59%). Nonetheless, due to the larger number of text lines classified as Very Positive (55% of 231) and a very high degree of accuracy for this category, the overall accuracy of Component 2 of our method reached 81%.

Table 17. Evaluation criteria for manually checking sentiment classifications

Sentiment Classification	Evaluation Criterion
Very Positive	When the content shows a very happy or excited mood or high satisfaction.
Positive	When the content shows a happy or excited mood or satisfaction without a very strong expression.
Neutral	When the content does not clearly show positive or negative emotions or the content has contradictory expressions.
Negative	When the content shows an unhappy or disappointed mood without a very strong expression.
Very Negative	When the content shows a very unhappy or disappointed mood.

4.2.3. Overall Performance of Component 1 and Component 2

To see the combined accuracy of both components 1 and 2, we consider those lines of text that were classified into the correct categories both in Components 1 and 2 as the final correct classifications. As described in Sections 4.1 and 4.2, the analysis of classification accuracy was done manually based on a sample of 231 text lines (out of a total of 1,151 classified text lines). The results of this analysis are shown in Tables 19 and 20, which present the results for the text lines classified

Table 18. Prediction accuracy of Component 2

	Lines of Text	Proportion	Samples	Correct Classification	Accuracy
Very Positive	629	55%	127	118	93%
Positive	154	13%	31	18	58%
Neutral	135	12%	27	20	74%
Negative	146	13%	29	23	79%
Very Negative	87	7%	17	10	59%
Total	1151	100%	231	188	81%

in Component 1 as functional and dysfunctional, respectively. When comparing the results shown in Tables 19 and 20, we observe that the overall accuracy of text lines classified as functional (81%, i.e., 102 out of 126 text lines in the sample) is much higher than the accuracy of text lines classified as dysfunctional (46%, i.e., 48 out of 105 text lines in the sample). The overall weighted average of the accuracy of all 231 text lines of the sample is 65%.

If we take a closer look at the details of Tables 19 and 20, we can see that classification accuracy varies a lot. For example, the highest overall accuracy is 92% for text lines expressing very positive emotions about something that exists (functional). On the other hand, the overall accuracy of dysfunctional text lines is very low, especially when expressing very positive (32%), positive (40%), and neutral (47%) emotions. While the accuracy of text lines classified as functional is generally better than that of text lines classified as dysfunctional, those text lines classified as dysfunctional that express negative and very negative emotions have higher accuracy (74%, resp. 60%) than the corresponding text lines classified as functional (60%, resp. 57%). We also observe that the main cause for low overall accuracy can be traced to both components of the method depending on the sentiment classification. For example, the low overall accuracy of 32% for text lines classified as dysfunctional and expressing very positive emotion is mostly due to the low accuracy in Component 1 (36%). On the other hand, the relatively low overall accuracy of 56% for text lines classified as functional and expressing positive emotion is mostly due to low accuracy in Component 2 (56%).

When comparing the accuracy of the method applied in Components 1 and 2, we observe that the lowest accuracy value for Component 1 is 36%, which is the only value lower than 65%, while the lowest accuracy values for Component 2 are 56%, 57%, 60%, and 60%, respectively. The low accuracy values in Component 2 are related to text lines classified as Positive and Very Negative for both the functional and dysfunctional categories.

4.3. Component 3 - Kano-like Processing

In this section, we will describe our design of two Kano-like models, i.e., a Half-Kano model and a Deformed-Kano model, that can be applied when the traditional

Table 19. Accuracy of Components 1 and 2 for text lines classified as functional in Component 1

	Lines of Text	Proportion	Analyzed Examples	Component 1		Component 2		Overall	
				Correct Classification	Accuracy	Correct Classification	Accuracy	Correct Classification	Accuracy
Very Positive	413	66%	83	77	93%	77	93%	76	92%
Positive	78	12%	16	14	88%	9	56%	9	56%
Neutral	52	8%	10	7	70%	9	90%	7	70%
Negative	50	8%	10	8	80%	7	70%	6	60%
Very Negative	35	6%	7	6	86%	4	57%	4	57%
Total	628	100%	126	116	92%	107	85%	102	81%

Table 20. Accuracy of Components 1 and 2 for text lines classified as dysfunctional in Component 1

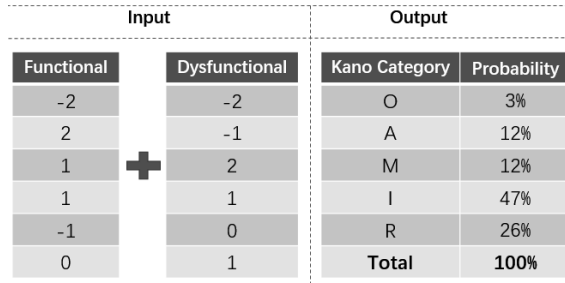
	Lines of Text	Proportion	Analyzed Examples	Component 1		Component 2		Overall	
				Correct Classification	Accuracy	Correct Classification	Accuracy	Correct Classification	Accuracy
Very Positive	216	41%	44	16	36%	41	93%	14	32%
Positive	76	15%	15	10	67%	9	60%	6	40%
Neutral	83	16%	17	11	65%	11	65%	8	47%
Negative	96	18%	19	16	84%	15	79%	14	74%
Very Negative	52	10%	10	7	70%	6	60%	6	60%
Total	523	100%	105	61	58%	85	81%	48	46%

Kano model cannot be used because the functional and dysfunctional input is unpaired or partially missing. Here, we prioritize the formatted data following the principle of the Kano model [108]. Figure 21[5] shows the example input and the example output of Component 3. Compared with the traditional Kano model, we use the probability of each category that one feature is categorized into instead of the one category to which the feature is most frequently categorized.

To apply the traditional Kano model to data extracted from the Internet, we assume that we have already filtered out the sentiment information expressing a person's feeling from online reviews, comments, or questions, and that we have translated this sentiment information into a data set similar to the format of the traditional Kano model. For example, the statement "I dislike X very much!" represents a very negative answer to a functional question regarding feature X, while "I would be very happy if there were no function X." represents a very positive answer to a dysfunctional question regarding feature X. We put all answers to the functional questions in a "Yes" (Y) vector, and all answers to the dysfunctional questions in a "No" (N) vector.

One of the biggest problems we are facing is how to pair the input required for the traditional Kano model without conducting interviews with real people to get answers to both functional and dysfunctional questions. Reviews and comments from online sources are usually unpaired, so we cannot process the data following the traditional Kano model. Because of this, we designed Kano-like model algorithms for processing unpaired data.

[5] O = One-dimensional Quality, A = Attractive Quality, M = Must-be Quality, I = Indifferent Quality, R = Reverse Quality.



Input			Output	
Functional		Dysfunctional	Kano Category	Probability
-2		-2	O	3%
2		-1	A	12%
1	✚	2	M	12%
1		1	I	47%
-1		0	R	26%
0		1	**Total**	**100%**

Figure 21. Example input and example output of Component 3

The two Kano-like models we propose differ in the way they interpret the unpaired answers derived from online open sources. The assumption of the Half-Kano model is that we only have either answers to a functional question or answers to a dysfunctional question. The assumption of the Deformed-Kano model is that answers to the functional and the dysfunctional questions are from the same group of people, even though we lost the links between answers. One output of the traditional Kano model only contains one specific category into which a feature is classified. However, to enable comparison with the output of the Kano-like models in our study, we use the probability of each category that one feature is categorized into instead of the one category to which the feature is most frequently categorized to. For example, if we get five paired answers about one feature, and each paired answer leads to one category, then we have a list of five categories (e.g., "M", "M", "A", "M", "O"). The traditional Kano model output is that this feature is classified into category "M". However, in our study, we say that the output is that there is a 60% probability that this feature is classified into category "M", a 20% probability that it is classified into category "A", and a 20% probability that it is classified into category "O".

4.3.1. Half-Kano Model

To implement the Kano-like models on unpaired data, we assumed the following extreme case: Each time we have an interview with our interviewees, we ask only functional questions or only dysfunctional questions relating to one software feature; hence we only get two groups of responses for functional, resp. dysfunctional, questions from the different interviewees. In such a case, we cannot use two responses from different interviewees to classify a person's satisfaction and then, based on that, derive the Kano category to which the software feature belongs. However, we can implement an algorithm that calculates the probability with which a software feature would be classified based on the responses for the functional and the dysfunctional questions. Since the data in the Y and the N vectors are not matched, the Half-Kano model is not a traditional Kano model. Nevertheless, we calculate the probabilities following the traditional Kano model. The difference is that in this method, we use each signal value from vectors Y and

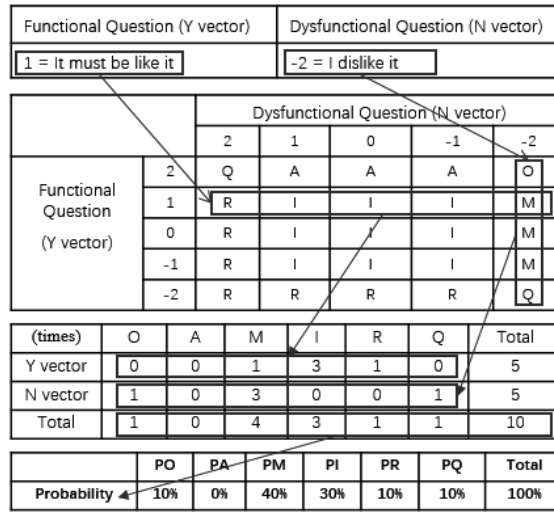


Figure 22. Example of the process of the Half-Kano model

N to derive a Kano category. Figure 22 shows an example of how the Half-Kano model processes the Y and N vectors when the unpaired input is "1" for vector Y and "-2" for vector N. In this case, we can say that this feature should be classified into the “M” category with a probability of 40%, into the "I" category with a probability of 30%, and into the "R", "O", and "Q" categories with a probability of 10% each.

The algorithm of probability (P) that vector Y (functional) and vector N (dysfunctional) will be categorized into the same category (X) can be written as

$$P\left(cat\left(Y\right)=cat(N)=X\right)=\frac{\sum_{i=1}^{m}F_{x}cat\left(Y(i)\right)+\sum_{j=1}^{n}F_{x}cat(N(j))}{(m+n)*5}$$

and

$$F_{O}cat(Y(i))=\begin{cases}1 & \text{if } Y(i)=2\\ 0 & \text{if } Y(i)=\{-2,-1,0,1\}.\end{cases}$$

$$F_{A}cat(Y(i))=\begin{cases}3 & \text{if } Y(i)=2\\ 0 & \text{if } Y(i)=\{-2,-1,0,1\}.\end{cases}$$

$$F_{M}cat(Y(i))=\begin{cases}1 & \text{if } Y(i)=\{-1,0,1\}.\\ 0 & \text{if } Y(i)=\{-2,2\}.\end{cases}$$

$$F_{I}cat(Y(i))=\begin{cases}3 & \text{if } Y(i)=\{-1,0,1\}.\\ 0 & \text{if } Y(i)=\{-2,2\}.\end{cases}$$

$$F_{R}cat(Y(i))=\begin{cases}4 & \text{if } x=-2\\ 1 & \text{if } x=\{-1,0,1\}\\ 0 & \text{if } x=2\end{cases}$$

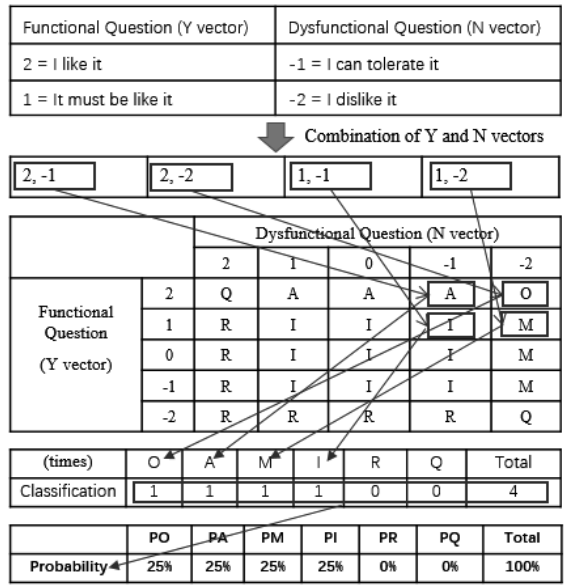


Figure 23. Example of the process of the Deformed-Kano model

$$F_Q cat(Y(i)) = \begin{cases} 1 & \text{if } Y(i) = \{-2, 2\}. \\ 0 & \text{if } Y(i) = \{-1, 0, 1\}. \end{cases}$$

and F_x is a function that maps the statement X to the set $\{0,1,3,4\}$

$$F_x : X \to \{0,1,3,4\}$$

where

$$i \in \{1,2,3...m\}$$
$$j \in \{1,2,3...n\}$$
$$X \in \{O,A,M,I,R,Q\}$$

m is the number of values of Y vector

n is the number of values of N vector

4.3.2. Deformed-Kano Model

In the Deformed-Kano model, we assume that the responses are from the same group of people, but we lost the links between the answers to the functional and dysfunctional questions.

We sequentially pick a number of vector Y and combine it with each number of vector N to derive the Kano categories, resulting in a list of Kano categories. After each value of vector Y has been combined with all values of vector N, we calculate the overall proportion of the appearance of each category. Figure 23 shows an example of the process of the Deformed-Kano model when the unpaired input is "2, 1" for vector Y, and "-1, -2" for vector N. The output is that the probability that this feature is classified into category "M", "A", "O", and "I" is 25% for each category.

The algorithm of the probability (P) that vector Y (functional) and vector N (dysfunctional) are categorized into the same category (X) can be written as

$$P(cat(Y) = cat(N) = X) = \frac{\sum_{i=1}^{m}(\sum_{j=1}^{n}(cat(Y(i)) * cat((N(j))))}{m * n}$$

and

$$cat(Y(i)) * cat(N(j)) = \begin{cases} O & \text{if } Y(i) = 2 and N(j) = -2 \\ A & \text{if } Y(i) = 2 and N(j) \in \{-1,0,1\} \\ M & \text{if } Y(i) \in \{-1,0,1\} and N(j) = -2 \\ I & \text{if } Y(i) \in \{-1,0,1\} and N(j) \in \{-1,0,1\} \\ R & \text{if } Y(i) \in \{-2,-1,0,1\} and N(j) = 2 || Y(i) = -2 and N(j) \in \{-1,0,1\} \\ Q & \text{if } Y(i) = 2 and N(j) = 2 || Y(i) = -2 and N(j) = -2 \end{cases}$$

and x_x is a function that maps the statement X to the set $\{0,1,3,4\}$

$$x_x : X \to \{0,1\}$$

where

$$i \in \{1,2,3...m\}$$
$$j \in \{1,2,3...n\}$$
$$X \in \{O,A,M,I,R,Q\}$$

m is the number of values of Y vector
n is the number of values of N vector

4.3.3. Simulation of Kano-like Models

To see which type of Kano-like model performs better, we ran simulations with artificial inputs. According to the simulation results, we chose the model that performed better as the method for Component 3.

Simulation Input: There are 31 possible value sets in both vector Y and vector N. For example, value set ID No. 1 indicates that vectors Y and N only contain elements with the value "-2". Value set ID No. 31 indicates that both vectors contain all possible values, i.e., "-2, -1, 0, 1, 2".

Simulation Approach: We used the R language to execute the simulation algorithms we proposed in sections 4.3.1 and 4.3.2. We first set the length of vectors Y and N to 20, and then picked these 20 numbers from each possible value set to simulate the responses of one feature. We combined all 31 possible value sets of vectors Y and N. The total number of possible combinations of the value sets of vectors Y and N is 31 * 31 = 961. In each simulation round, for each combination of value sets of Y and N, we sampled the data randomly, following a predefined distribution, e.g., uniform distribution. Then we ran the traditional Kano model and the Kano-like models five times each. Next, we calculated the average value of those with the same value set ID of vectors Y and N and combined them, which finally yielded a table containing 961 rows and 20 columns (value set ID of vectors Y and N plus PO, PA, PM, PI, PR, and PQ for the traditional Kano, Half-Kano, and Deformed-Kano model, respectively).

Simulation Hypothesis 1: The output generated by the Deformed-Kano model is more similar to the traditional Kano model than that of the Half-Kano model.

We picked the data from one of the 961 rows to show an example of how to calculate the difference between the traditional Kano model and the Kano-like models. Table 21 shows how to calculate the difference between the traditional Kano model and the Kano-like models, and also shows the calculation results. For calculating the absolute value of the difference between the two sets of data (traditional Kano model and Half-Kano model, or traditional Kano model and Deformed-Kano model), the range was 0 to 200%. Hence, we divided the absolute value by 2 to get a result in the range of 0 to 100%.

Table 21. An example of the difference between the traditional Kano model and the Kano-like models

	PO	PA	PM	PI	PR	PQ
Traditional (%)	0	20	0	45	25	10
Half (%)	3	15.5	7	40.5	27.5	6.5
Deformed (%)	0	19.5	0	45.5	24.5	10.5
Difference (%)	Traditional – Half = $(\|0-3\|+\|20-15.5\|+\|0-7\|+\|45-40.5\|+\|25-27.5\|+\|10-6.5\|)/2$=12.5					
	Traditional – Deformed = $(\|0-0\|+\|20-19.5\|+\|0-0\|+\|45-45.5\|+\|25-24.5\|+\|10-10.5\|)/2$=1					

The lower value of the difference represents output that is closer to the traditional Kano model. In the case shown in Table 21, we see that the Deformed-Kano model's output is closer to the output of the traditional Kano model (difference = 1%) than the output of the Half-Kano model (difference = 12.5%).

The ranges and means of the differences between the outputs of the traditional Kano model and the Kano-like models are shown in Table 22. We see from Table 22 that the range of differences between the traditional Kano model and the Half-Kano model varies from 10.5% to 80%, which is much higher than the range of differences between the traditional Kano model and the Deformed-Kano model, which is 0% to 18.74%. The means show the same trend: 25.99% between the traditional Kano model and the Half-Kano model, and 4.28% between the traditional Kano model and the Deformed-Kano model.

Table 22. The range and means of the differences between the outputs of the traditional Kano model and the Kano-like models

	Traditional-Half (%)	Traditional-Deformed (%)
Ranges	[10.5, 80]	[0, 18.74]
Means	25.99	4.28

To see the distribution of the differences of the outputs between the traditional Kano model and the Kano-like models more clearly, we drew figures. Figures 24 and 25 show that the Deformed-Kano model shows outputs that have lower differences to the outputs of the traditional Kano model than the outputs of the Half-Kano model. This means that Simulation Hypothesis 1 has been confirmed as true.

Simulation Hypothesis 2: The Deformed-Kano model yields similar outputs as the traditional Kano model.

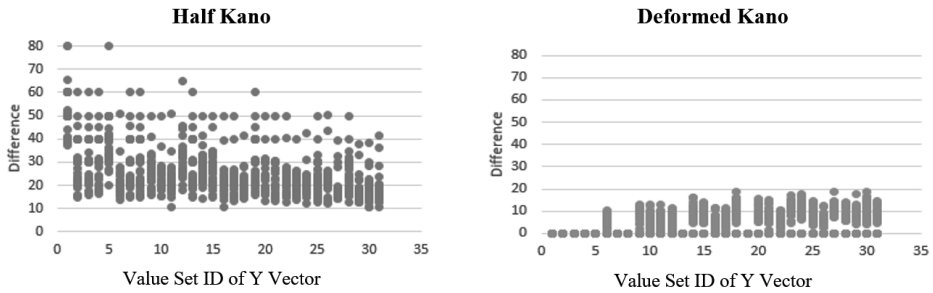


Figure 24. Projection of the distribution of the differences between the traditional Kano model and the Kano-like models on the Y vector plane

According to the simulation results, we found that when the input value of the vector Y or the vector N belongs to the set $\{-2\}$, $\{-1\}$, $\{0\}$, $\{1\}$, $\{2\}$, $\{-2,0\}$, $\{-2,1\}$, $\{0,1\}$, $\{-2,0,1\}$, the difference always equals zero, which means 477 out of 961 (49.6%) output combinations of the Deformed-Kano and the traditional Kano model show no difference.

When the input value of vector Y or the vector N does not belong to the set $\{-2\}$, $\{-1\}$, $\{0\}$, $\{1\}$, $\{2\}$, $\{-2,0\}$, $\{-2,1\}$, $\{0,1\}$, $\{-2,0,1\}$, the difference will always be more than zero. 484 out of 961 (50.4%) combinations show differences ranging from 1% to 18%. In addition, the average values are less than 11%.

Simulation Result: The results of our simulation experiments revealed that the results of using the Deformed-Kano model were always close to the results of the traditional Kano model. Because of that, we consider the Deformed-Kano model to be a good approximation of the traditional Kano model. Moreover, the Deformed-Kano model can be used even when the input is unbalanced or partly missing. We believe that the low cost of using the Deformed-Kano model combined with the possibility to use unbalanced data compensates for the potential lack of paired data compared to the traditional Kano model. Therefore, we chose the Deformed-Kano model as the default method for Component 3 (Kano-like Processing).

4.4. Component 4 - Visualization

In this section, we will visualize the outputs of Components 1 to 3. Figure 26 shows three examples of different output format, i.e., pie chart, bar chart, and table.

4.5. Work Process

The work process of implementing the OIRE method is shown in Figure 27. The OIRE method provides users with two work models. The first is the sentiment

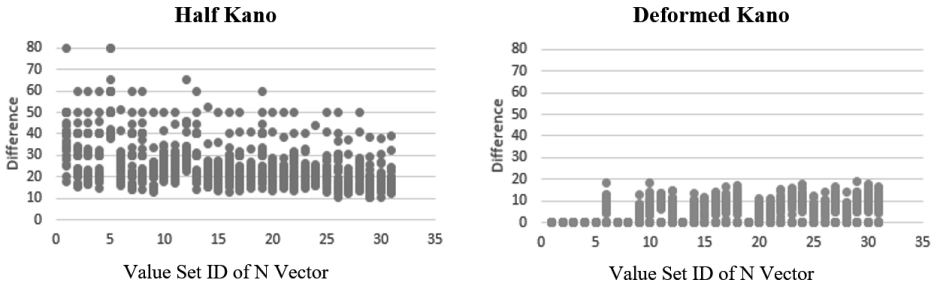
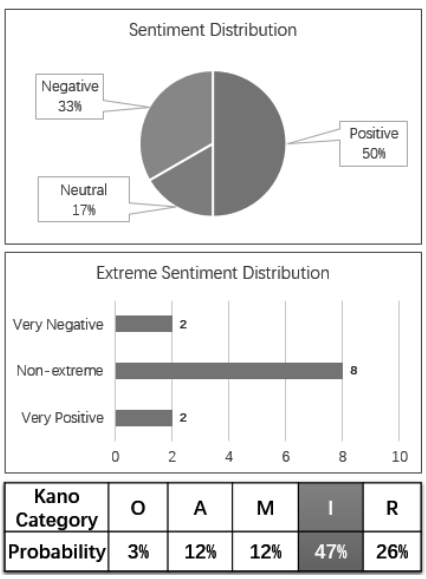


Figure 25. Projection of the distribution of the differences between the traditional Kano model and the Kano-like models on the N vector plane



Kano Category	O	A	M	I	R
Probability	3%	12%	12%	47%	26%

Figure 26. Example visualizations produced by Component 4

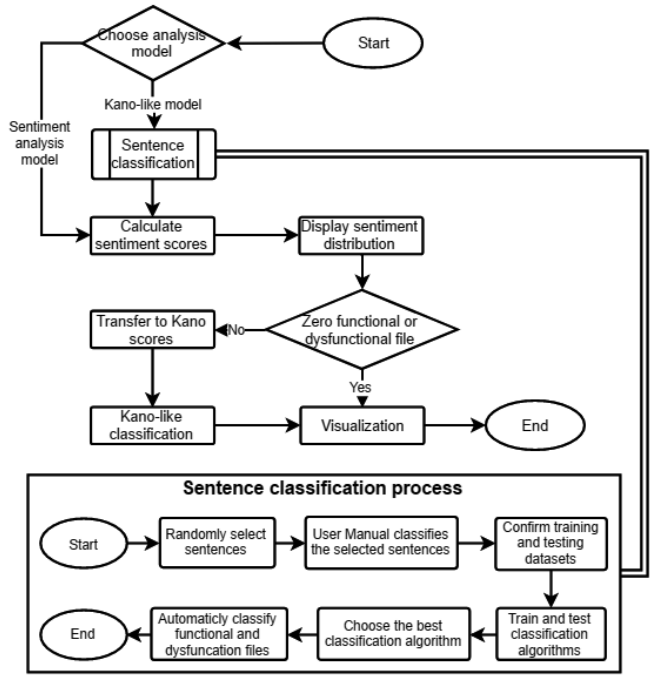


Figure 27. Flowchart of implementing the OIRE method

analysis model (Component 2) and the second is the Kano-like analysis model (Components 1+2+3).

The advantage of the sentiment analysis model is that it is simple and saves time. The disadvantage is that the output is not comprehensive. For example, when a user says: "If you could remove this feature, I would be very happy", then the result is "Very Positive" if we use the sentiment analysis model. However, we understand that the user intended to express that they are very dissatisfied with this feature; hence, we got a "correct", but misunderstood result. The advantage of the Kano-like model is that the result is more comprehensive, while the disadvantage is that it is time-consuming. Therefore, we provide these two models at the same time so that users can choose which model is more suitable for their needs.

We see from Figure 27 that when a user chooses to use the Kano-like model to do an analysis, the input data is first processed by the "Sentiment Classification" process. The "Sentiment Classification" process has six sub-processes, which are shown in Figure 27 as well. After that, the processed data enters the processes "Calculate sentiment score" and "Transfer to Kano score" and then undergoes the "Kano-like classification" process. Finally, the "Visualization" process presents users a variety of charts and a diversity of information.

4.6. Summary

In this chapter, we introduced the four compositions of the OIRE method and how each algorithm was designed. We used experimental methods to verify the availability of the algorithms of Component 1 and Component 2, and we found that the accuracy of the algorithm was more than 60%. We used simulations to compare the performance of two Kano-like models (Half-Kano and Deformed-Kano) with that of the traditional Kano model. We found that the Deformed-Kano model produced outputs that were more similar to those of the traditional Kano model than those produced by the Half-Kano model. Therefore, we chose the Deformed-Kano model as the default method for Component 3. We also showed an example of Component 4 and the work process of implementing the OIRE method. In the next chapter, we will introduce tool support for the OIRE method, OIRE-S.

5. OIRE TOOL SUPPORT

To support the application of the four components of the OIRE method, we developed OIRE-S, a prototypical web application. OIRE-S was implemented using R, PHP, Apache, and Wampserver. The OIRE-S prototype supports only basic file operations.

This chapter is structured as follows. Section 5.1 describes the structure of OIRE-S, while Section 5.2 shows how to implement OIRE-S.

5.1. Structure

OIRE-S offers three functions: "Upload file", "Sentiment analysis", and "Kano-like analysis". In addition, OIRE-S comprises four basic modules corresponding to the components of the OIRE method. The basic modules are grouped in different ways to support OIRE-S's two main functions "Sentiment analysis" and "Kano-like analysis". The "Sentiment analysis" function uses the "Sentiment mining" and "Visualization" modules. The "Kano-like analysis" function uses the "Sentence classification", "Sentiment mining", "Kano-like processing", and "Visualization" modules. Figure 28 shows the structure of OIRE-S. In the following, we will briefly describe each of the functions.

"Upload file" function: The user selects the input files that need to be analyzed and uploads them to the system. The format of the file must be CSV (Comma-Separated Values file). Figure 29 shows the process flow of the "Upload file" function. If there are several features to be analyzed, several files can be uploaded.

"Sentiment analysis" function: From among the uploaded files, the user selects those files for which a sentiment analysis is to be done. OIRE-S will perform the sentiment analysis and display the results. Figure 30 shows the process flow of the "Sentiment analysis" function.

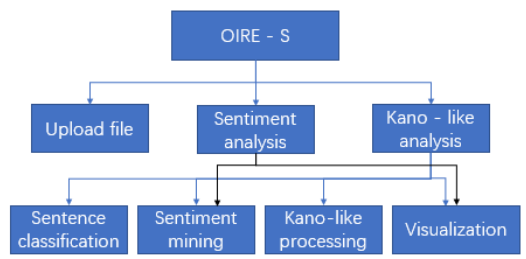


Figure 28. Structure of OIRE-S

Figure 29. Flowchart of the function "Upload file"

Figure 30. Flowchart of the function "Sentiment analysis"

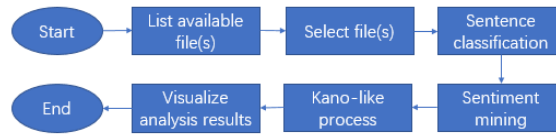


Figure 31. Flowchart of the function "Kano-like analysis"

"Kano-like analysis" function: From among the uploaded files, the user selects those files for which a Kano-like analysis is to be done. OIRE-S will automatically analyze the file contents and display the results of the analysis. Figure 31 shows the process flow of the "Kano-like analysis" function.

5.2. Implementation

We designed OIRE-S for Windows 10. We used R-3.5.0, PHP-5.6.19, Apache-2.4.18, Jpgraph-4.2.1 and Wampserver 64 to implement the system. We made the code open source [1] so that users can modify OIRE-S according to their actual

[1] https://figshare.com/s/9bc19c086449be76ed90

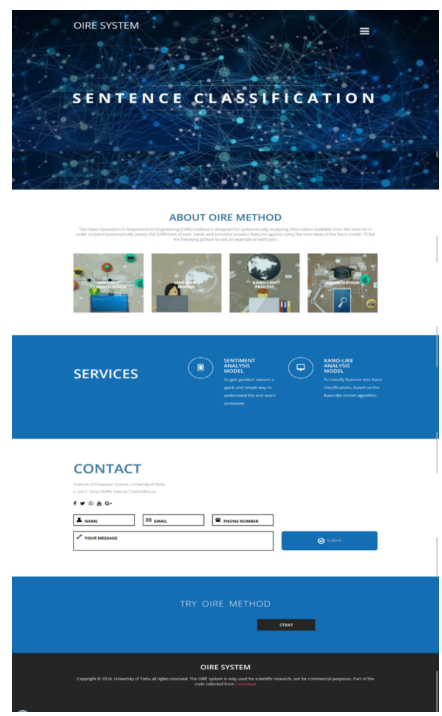


Figure 32. Main page of OIRE-S

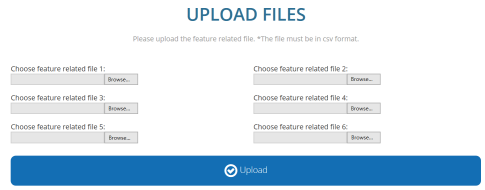


Figure 33. "Upload file" function

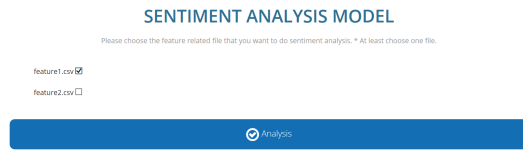


Figure 34. "Sentiment analysis" function - selecting files

requirements. For example, users can add new machine learning methods to the "Sentence classification" module or modify the visual style of the analysis results. Figure 32 shows an image of the homepage of the OIRE-S web application.

5.2.1. Function "Upload file"

Figure 33 shows the interface (input side) of the "Upload file" function. Users can choose the file(s) to be uploaded and then click on the "Upload" button.

5.2.2. Function "Sentiment analysis"

Figure 34 and Figure 35 show the user interface (input and output sides) of the "Sentiment analysis" function. First, as shown in Figure 34, a user chooses the file(s) to be analyzed by the "Sentiment analysis" function. Then they click on the "Analysis" button, and the file(s) is/are analyzed automatically. Figure 35 shows the visualization of the output of the sentiment analysis. The "Sentiment analysis" function of the OIRE method classifies the sentiment polarity of the input data into five categories: Very Negative, Negative, Neutral, Positive, and Very Positive. For visualization purposes, the sentiment of the input data is mapped to three categories in two different ways. The pie chart on the left of Figure 35 shows the sentiment distribution of the input data mapped to the categories Negative (merging Very Negative and Negative sentiments), Neutral, and Positive (merging Positive and Very Positive sentiments). The bar chart on the right of Figure 35 shows the extreme sentiment distribution, where the sentiment distribution of the input data is mapped to Very Negative, Non-Extreme (merging Negative, Neutral, and Positive), and Very Positive.

5.2.3. Function "Kano-like analysis"

Figures 36 to 38 show the user interface (input side) of the "Kano-like analysis" function. First, as shown in Figure 36, a user chooses the file(s) to be analyzed.

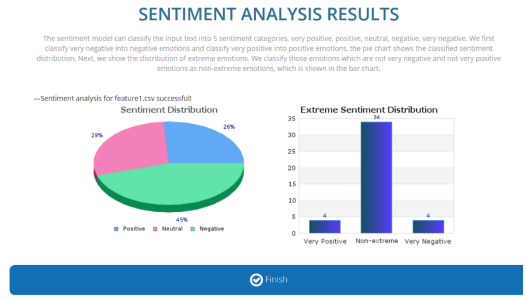


Figure 35. "Sentiment analysis" function - visualizing the analysis output

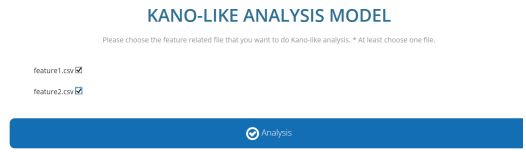


Figure 36. "Kano-like analysis" function - selecting files

Next, they click on the "Analysis" button, and the file(s) is/are analyzed automatically. Step 1 classifies the text lines in each file into "functional" and "dysfunctional". Then the user clicks on the "Proceed to Step 2" button. In Step 2, the sentiment analysis is performed. Then the user clicks on the "Proceed to Step 3" button, and the output file(s) of Step 2 is/are analyzed automatically in Step 3. In Step 3, the "Kano-like processing" is conducted, where for each feature for which an input file has been submitted, the Kano category distribution is calculated. The results of Step 3 are presented using the "Visualization" module.

Figure 39 shows the visualization of the "Kano-like analysis" output. The visualization consists of three parts: sentiment distribution of functional sentences, sentiment distribution of dysfunctional sentences, and Kano-like classification. The sentiment distribution of functional sentences embodies the users' feeling about the existence or good performance of a feature. The sentiment distribution of dysfunctional sentences embodies users' feeling about the lack or poor performance of a feature. The Kano-like classification table shows the probability distribution over Kano categories for each feature.

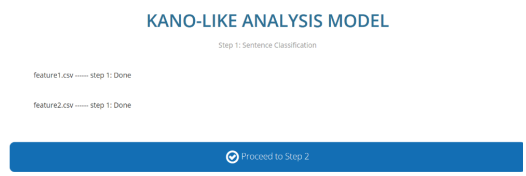
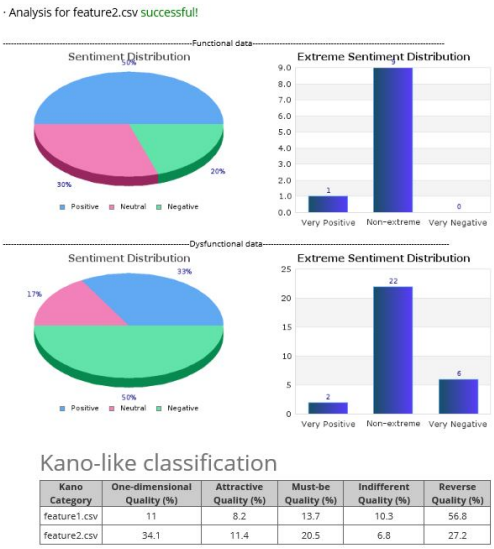


Figure 37. "Kano-like analysis" function - step 1: Sentence classification

Figure 38. "Kano-like analysis" function - step 2: Sentiment analysis



Kano-like classification

Kano Category	One-dimensional Quality (%)	Attractive Quality (%)	Must-be Quality (%)	Indifferent Quality (%)	Reverse Quality (%)
feature1.csv	11	8.2	13.7	10.3	56.8
feature2.csv	34.1	11.4	20.5	6.8	27.2

Figure 39. "Kano-like analysis" function - visualizing the analysis output

5.3. Summary

In this chapter, we introduced our tool support for the OIRE method, OIRE-S. OIRE-S has three main functions: "Upload files", "Sentiment analysis", and "Kano-like analysis". We introduced the function structure of OIRE-S as well as the operation flow of each function and showed screenshots of OIRE-S.

6. VALIDATION

In the previous chapter, we described a web-based prototypical system, OIRE-S, for implementing the OIRE method. In this chapter, we will describe the validation with a more detailed demonstration of how to use the OIRE method. We will present three typical use cases and a proof-of-concept of the method to validate the applicability of our method using real-world data collected from the Internet. We semi-automatically ran the process of the OIRE method to explain what it can do and how it provides useful information to help software engineers and managers make better-informed decisions.

This chapter is structured as follows. In Section 6.1, we will describe the selection of the input sources of the OIRE method. In Section 6.2, we will outline the design of the use cases and in Section 6.3, the application of the use cases. In Section 6.4, we will summarize the threats to validity. In Section 6.5, we will discuss our findings, and Section 6.6 will conclude the validation study.

6.1. Task-Adequacy of Input Source

There are different types of online open sources, and we believe that different data sources can be selected for different analytical purposes. In order to investigate which kind of data source (data from Question and Answer sites or data from App reviews) is more suitable for the application of the OIRE method, we ran an experiment to test the accuracy of Component 2 (Sentiment Mining) as the starting point of our validation research. Our reason for choosing Component 2 is that Component 2 is the process required by the two models offered by the OIRE method (the sentiment analysis model and the Kano-like analysis model). In the experiment, we used a small set of 250 user questions and comments collected from Stack Overflow as our test data set 1. We used another small set of 250 reviews collected from the Google Play store and the Apple App Store as our test data set 2. We manually labeled the test inputs by attaching a sentiment score to all 500 data text lines. To do so, we used the criteria shown in Table 17, Chapter 4.

Tables 23 and 24 show the actual and predicted classifications of data set 1 and data set 2, respectively. Gray table cells show the numbers of those cases where data text lines were classified correctly by the dictionary-based method. We see from Tables 23 and 24 that the overall prediction accuracy was similar (71.6% and 70.8%). We also observe that the highest prediction accuracy for both data set 1 and data set 2 was achieved for text lines expressing very positive emotions.

Another observation we made is related to the distribution of sentiments in the two data sets. Data set 1, containing data collected from a Question & Answer site, had considerably more text lines expressing neutral emotions than data set 2, which contained data collected from App reviews. Also, data set 1 contained more text lines expressing positive or very positive emotions and fewer text lines

expressing negative or very negative emotions than data set 2. We think this is because when people post a question, they usually describe a problem and need an answer, so most of the posts are written in an objective mode describing facts, rather than in a subjective mode expressing emotions. Even if someone wants to express feelings in a question/answer forum, it is difficult for someone to have a positive feeling when they have a problem. However, when writing an app review, the sentiments expressed are often related to features and if a feature is good/bad, more positive/negative sentiments will be expressed. Thus, app reviews are potentially more comprehensive with regard to the expression of sentiments. Based on that, we think that input stemming from reviews (such as those found in App stores) is a more suitable data source for our purposes than Question & Answer sites. Therefore, in this chapter, we will use reviews collected from App stores as input data for the validation.

Table 23. Prediction accuracy based on data set 1 (Stack Overflow)

		Predicted Sentiment Classification					Accurate	Text lines	Accuracy
		Very Positive	Positive	Neutral	Negative	Very Negative			
Actual Sentiment Classification	Very Positive	10	2	0	0	0	10	12	83%
	Positive	2	30	1	8	1	30	42	64%
	Neutral	1	15	61	9	1	61	87	70%
	Negative	1	12	14	68	0	68	95	72%
	Very Negative	1	2	0	1	10	10	14	71%
Overall							179	250	71.6%

Table 24. Prediction accuracy based on data set 2 (Google Play and Apple App Store)

		Predicted Sentiment Classification					Accurate	Text lines	Accuracy
		Very Positive	Positive	Neutral	Negative	Very Negative			
Actual Sentiment Classification	Very Positive	86	3	2	1	1	86	93	92%
	Positive	8	30	2	4	2	30	46	65%
	Neutral	7	3	27	6	2	27	45	60%
	Negative	13	10	0	24	4	24	51	47%
	Very Negative	5	0	0	0	10	10	15	67%
Overall							177	250	70.8%

6.2. Use Cases

In this section, we will introduce what the OIRE method can do for software organizations, specifically how it supports software product owners and developers in making decisions will. We describe three typical use cases representing the application of the OIRE method using a prototype of OIRE-S because we believe these three kinds of use cases are most often faced by product managers and product owners.

6.2.1. Use Case 1

The OIRE method can help product owners (PO) to better understand end users' feelings about an existing product. For example, when the PO needs a quick

and simple way to understand the distribution of end users' sentiment regarding a particular feature of a product, they can use the OIRE method. Table 25 shows the context of Use Case 1, while Table 26 presents the description of Use Case 1.

Table 25. Context of Use Case 1

Role	Product Owner (PO)
Situation	The PO has a product.
Question	The PO needs a quick and simple way to understand the end users' sentiment distribution regarding a feature of the product.
Initial Input	The PO has text (sentences) related to this feature.
Component 2	Input: A list of sentences (lines of text) where each line relates to a known feature. Output: The lines of text in files have a score (-2 to +2).
Component 4	Input: The output of Component 2. Output: The distribution (e.g. bar chart) of sentiment polarity of the feature.
Answer to the Question	According to the visualization of Component 4, the user can directly notice the end users' feelings about the feature.

Table 26. Use Case 1

Name	Find end users' feeling about one feature.
ID	UC-1
Description	The PO wants to add or delete one feature from their product, and wants to know the end users' feelings about the feature.
Actors	PO
Main course	1. A user chooses a text file to upload into OIRE-S. 2. OIRE-S prompts the user to confirm upload of the text file. 3. User confirms to upload the text file (see EX1) 4. OIRE-S stores the file. (see EX2) 5. Show notification "file is uploaded successfully" Component 2: 6. OIRE-S automatically calculates the sentiment score of each text line of the input file. 7. OIRE-S shows the diagram of the distribution of sentiment polarity. Component 4: 8. OIRE-S shows all the results provided by Component 2 of the OIRE method.
Exceptions	EX1: User decides to upload another file. 1. Return user to Main Course step 1 EX2: OIRE-S fails on uploading the file to the system 1. OIRE-S notifies the user that an error has occurred (e.g. format problem). 2. Return user to Main Course step 1

6.2.2. Use Case 2

The OIRE method can classify features into Kano categories based on the Kano-like model. For example, when the PO wants to add or delete a feature from their product and wants to know the end users' feelings about that feature, they can use

the OIRE method to analyze a single feature. Table 27 shows the context of Use Case 2, while Table 28 presents the description of Use Case 2.

Table 27. Context of Use Case 2

Role	Product Owner (PO)
Situation	The PO has a product
Question	The PO wants to add or delete a feature from their product, and wants to know end users' feelings about the feature.
Initial Input	The PO has text (sentences) related to this feature.
Component 1	Input: A list of sentences (lines of text) where each line relates to a known feature. Output: The lines of text are classified as "functional" or "dysfunctional".
Component 2	Input: The output of Component 1. Output: The lines of text in files have a score (-2 to +2).
Component 3	Input: The output of Component 2. Output: Kano-like classification of the feature contained in the input file of Component 1.
Component 4	Input: a) The output of Component 1. b) The output of Component 2. c) The output of Component 3. Output: The distribution (e.g., bar chart) of the sentiment polarity of the feature. The probabilities with which the feature is classified to each Kano category, which represents end the users' feelings.
Answer to the Question	According to the visualization of Component 4, the user can directly notice the end users' feelings about the feature and adding or deleting this feature would have what kind of impact on the users' feeling about the product.

6.2.3. Use Case 3

The OIRE method can prioritize classified features following the theory of the traditional Kano model. For example, when the PO plans to release a new version of an existing product, they might want to understand which new features should be developed first and which features should be deleted. Then they can use the OIRE method to prioritize the features. Another similar example is the situation where a product developer (PD) wants to know what features are necessary and most popular (most positive feelings from end users) in existing products. To decide which feature should be developed first, the PD can use the OIRE method to prioritize the features. Since these two scenarios have the same work process, we will only show one of them in this section. Table 29 shows the context of Use Case 3, while Table 30 presents the description of Use Case 3.

6.3. Application of the Use Cases

To show how the OIRE method interacts with the end user, in this section, we will present three applications of the use cases proposed in Section 6.2. We will use real-world input data to simulate the process and show the resulting output of the OIRE method for each use case application. To get the input data, we downloaded reviews of multiple apps from the Apple App Store and from a Chinese

Table 28. Use Case 2

Name	Find end users' feelings about a particular feature.
ID	UC-2
Description	The PO wants to add or delete a feature from their product and wants to know end users' feelings about this feature.
Actors	PO
Main course	1. User chooses a text file to upload to OIRE-S. 2. OIRE-S prompts the user to confirm upload of the text file. 3. User confirms to upload the text file (see EX1) 4. OIRE-S stores the file. (see EX2) 5. Show notification "file is uploaded successfully" Component 1: 6. In order to confirm the proportion of functional and dysfunctional text of feature related text lines, user randomly selects some text lines (e.g. 20% of the input file, but up to 20 lines of text) from the whole text and manually classifies them into "functional" or "dysfunctional" categories by following the classification examples shown in Table 11. (See AC 1) 7. OIRE-S uses the data manually classified by the user as a test dataset to test the accuracy of the different algorithms. (e.g. SVM, DT, RF). 8. OIRE-S outputs the classification results of the different algorithms. (EX3) 9. User compares the classification results. 10. User selects the best performing algorithm to do the classification. 11. OIRE-S uses the algorithm chosen by the user to classify all text lines into "functional" or "dysfunctional" categories and to split the initial input file into a functional file and a dysfunctional file. 12. OIRE-S shows the classification results. (EX4) Component 2: 13. OIRE-S automatically calculates the sentiment score of each text line of functional and dysfunctional categories. 14. OIRE-S shows a diagram of the distribution of sentiment polarity. (See AC2, AC3 EX5) 15. OIRE-S converts the sentiment scores into Kano scores. Component 3: 16. OIRE-S uses the Kano-like model to classify the feature into Kano categories. Component 4: 17. OIRE-S shows all the results provided by Components 1 to 3 of the OIRE method.
Alternate course	AC1: Zero functional or dysfunctional data. 1. All text lines are classified into the same class, either "dysfunctional" or "functional". 2. Transfer user to Main Course step 12. AC2: Zero functional data. 1. Show the sentiment polarity of the feeling about the feature if it not exists. (More positive the sentiment polarity, the worse the feature.) AC3: Zero dysfunctional data. 1. Show the sentiment polarity of the feeling about the feature if it exists. (More negative the sentiment polarity, the worse the feature.) 2. OIRE-S notifies user that this is the end of the analysis.
Exceptions	EX1: User decides to upload another file. 1. Return user to Main Course step 1. EX2: OIRE-S fails to upload the file to the system. 1. OIRE-S notifies user that an error has occurred (e.g. format problem). 2. Return user to Main Course step 1 EX3: User decides to redo the manual classification process. 1. Return user to Main Course step 6 EX4: User wants to choose a different classifier. 1. Return user to Main Course step 10 EX5: Zero functional or dysfunctional data. 1.OIRE-S notifies the user that this is the end of the OIRE method.

app monitoring platform named Kuchuan[1]. We manually extracted features from the app descriptions and then used an appropriate online synonym dictionary[2]. We classified the synonymous features into one class and selected from the reviews the lines of text related to this feature class as input for the use cases. If the review text was in Chinese, we translated it into English. We followed the work process shown in Section 4.5.

Table 29. Context of Use Case 3

Role	Product Developer (PD)
Situation	The PD wants to develop a new product that contains several features.
Question	The PD wants to know what features are necessary and more popular (more positive feelings from end users) in existing products and which features should be developed first.
Initial Input	The PD has text (sentences) related to the features (existing in competitors' products).
Component 1	Input: Several files; and each file has a list of sentences (lines of text) with each line related to a known feature. Output: The lines of text are classified as "functional" and "dysfunctional".
Component 2	Input: The output of Component 1. Output: The lines of text in the files have a score (-2 to +2).
Component 3	Input: The output of Component 2. Output: Kano-like classification of the features contained in the input file of Component 1.
Component 4	Input: a) The output of Component 1. b) The output of Component 2. c) The output of Component 3. Output: The distribution (e.g., bar chart) of the sentiment polarity of each feature. The probability with which each feature is classified into each Kano category. User can rank features according to their own standards.
Answer for the Question	From the visualization of Component 4, the user can directly see whether features are popular (more positive user feelings) or unpopular (more negative user feelings). According to the ranking result, the user can decide which features should be developed first (ranking higher) and which are should be deleted (ranking lower).

In this section, we chose three apps, Pinterest, Meituan, and WeChat, as the input sources for our use cases. The reasons for choosing these three apps are that they are well-known (either in the Western world or in China) and that they have plenty of customer reviews that can be used in our analysis.

6.3.1. Application of Use Case 1

Regarding Use Case 1, we chose reviews of the app Pinterest collected from App Store, as they are openly available for research from the Website of the Swinburne University of Technology[3]. There are 9178 lines of review text. As the example

[1] http://android.kuchuan.com/page/detail/index (accessed: 30-April-2018)

[2] http://www.thesaurus.com/ (accessed: 15-October-2018)

[3] http://researchbank.swinburne.edu.au/vital/access/manager/Repository/swin:35267 (accessed: 15-October-2016)

Table 30. Use Case 3

Name	Prioritize the user's feelings about certain features.
ID	UC-3
Description	The PD wants to develop a new product that contains several features. The PD wants to know what features are necessary and which are more popular (more positive feelings from end users) in existing products, as well as which feature should be developed first.
Actors	PD
Main course	1. User chooses up to 10 text files into upload into OIRE-S. 2. OIRE-S prompts user to confirm upload of the text files. 3. User confirms upload of the text files (see EX1) 4. OIRE-S stores the files. (see EX2) 5. Show notification "Files are uploaded successfully" Component 1: For each input file, OIRE-S implements step 6 to step 12 (described in the main course of Use Case 2) in parallel. Component 2: 13. OIRE-S automatically calculates the sentiment score of each text line of functional and dysfunctional files. 14. OIRE-S shows a diagram of the distribution of the sentiment polarity per feature. (See AC2, AC3 EX5) 15. OIRE-S converts the sentiment scores into Kano scores. Component 3: 16. OIRE-S uses the Kano-like model to classify each feature into the Kano categories. Component 4: 17. OIRE-S shows the all results provided by Component 2 and 3 of the OIRE method. 18. User can rank features based on the output from Component 3 according to their own standard.
Alternate course	AC1: Zero functional or dysfunctional data. 1. OIRE-S defaults all text lines to the same class, either "dysfunctional" or "functional". 2. Transfer user to Main Course step 12. AC2: Zero functional data. 1. Show the sentiment polarity of the feelings about the feature if it not exists. (More positive the sentiment polarity, the worse the feature.) AC3: Zero dysfunctional data. 1. Show the sentiment polarity of the feelings about the feature if it exists. (More negative the sentiment polarity, the worse the feature.) 2. OIRE-S notifies user that this is the end of the analysis.
Exceptions	EX1: User decides to upload another file. 1. Return user to Main Course step 1. EX2: OIRE-S fails to upload the file to the system. 1. OIRE-S notifies user that an error has occurred (e.g., a format problem). 2. Return user to Main Course step 1 EX3: User decides to redo the manual classification process. 1. Return user to Main Course step 6 EX4: User wants to choose a different classifier. 1. Return user to Main Course step 10 EX5: Zero functional or dysfunctional data. 1. OIRE-S notifies user that this is the end of the OIRE method.

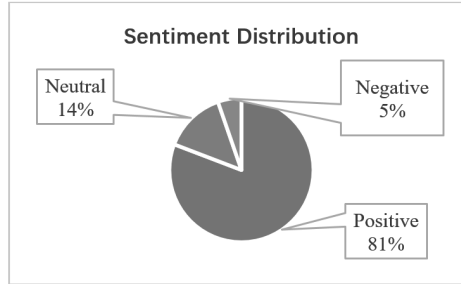


Figure 40. Sentiment distribution

app feature for implementing Use Case 1, we chose the first word combination that looked like a feature and occurred very frequently in the review texts. We ended up with the feature "already pinned". Then we classified synonymous features into one feature class. The resulting feature class contained "already pinned, pinned already, pinned before, already posted, posted already, repeated pinned, repeated post". Next, we searched all sentences containing the words in the feature class. In total, we extracted 151 lines of text related to the feature "already pinned". These 151 text lines constitute the initial input for our example application of Use Case 1.

Input: One file with 151 text lines relating to the elements in the feature class of the feature "already pinned" of the app Pinterest.

Approach: We first cleaned the data. We changed all capital letters to lowercase and removed punctuations and other strange characters, such as "@:(". Then we used Component 2 of the OIRE method to process the input.

Output: Table 31 shows the output when applying Use Case 1. We see from Table 31 that 93 text lines of the input show very positive feelings, accounting for the largest proportion. There are two lines of text showing very negative emotions, accounting for the smallest proportion. Next, we classified very negative emotions into negative emotions and very positive ones into positive emotions. In Figure 40, three categorized scales are shown. We see that 81% of the text lines showed positive emotions, 5% negative emotions, and the other 14% neutral emotions.

Table 31. Output of Use Case 1

Sentiment	Very Negative	Negative	Neutral	Positive	Very Positive	Total
Times	2	6	21	29	93	151

Next, we will show the distribution of extreme emotions. We classified those emotions that are not very negative and not very positive as non-extreme emotions. In Figure 41, we see that 37.1% of the text lines showed non-extreme emotions. 1.3% of the text lines were very negative and 61.6% showed very positive emotions.

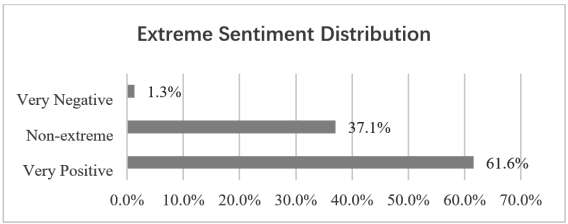


Figure 41. Extreme sentiment distribution

6.3.2. Application of Use Case 2

Regarding Use Case 2, we chose reviews of the app Meituan collected from the Chinese platform Kuchuan. The review texts are freely available from this platform. At the point in time when we conducted our analysis, there were 6,977 reviews. As the example app feature for implementing Use Case 2, we chose "Alipay" in the same manner as we chose the feature in the application of Use Case 1. Since "Alipay" is a proprietary feature, no synonyms for this feature are expected to exist, and thus the synonym tool was not needed in this situation. Next, we searched all the sentences containing "Alipay" in the reviews. In the end, we extracted 54 lines of text. Since the reviews were in Chinese, we translated these 54 lines of text into English. These 54 translated text lines constitute the initial input for our example application of Use Case 2.

Input: One file with 54 text lines relating to the feature "Alipay" of the app Meituan.

Approach: First, we randomly selected 12 text lines (22% of the input text lines) from the input text and manually classified them into functional and dysfunctional categories by following the classification standard we had already used in one of our previous studies [109]. These 12 labeled text lines were used as the test data set in Component 1 to select the best machine learning method, while the remaining 42 lines of text were then predicted using the best machine learning method in Component 1 of the OIRE method. In total, the OIRE method offers five ready-to-use machine-learning algorithms. We used a confusion matrix [93] to calculate the performance of each algorithm. Table 32 shows the definition of the confusion matrix. Based on the predicted results of each algorithm, we calculated the accuracy of each trained classification algorithm as well as the FPV, which shows the proportion of the functional text lines predicted correctly, and the DPV, which shows the proportion of the dysfunctional text lines predicted correctly. Table 33 shows the performance of each machine-learning algorithm. We see that SVM has the highest accuracy (75%) and relatively balanced FPV and DPV values. There are three algorithms that have the second best accuracy of 66%. Since the Random Forest algorithm has high accuracy (66%) and the most balanced values for FPV and DPV (resulting in a standard deviation of 0), we decided to use Random Forest together with SVM in our application example to increase the reliability of the output results. We only kept those cases where

the predictions of the two algorithms were consistent. We found that 39 out of the 42 lines of text were classified into the same categories when using both SVM and Random Forest together. 17 lines of text were classified as functional, and 22 were classified as dysfunctional. On top of this, we added the 12 manually classified text lines. In the end, we had one file containing 20 functional and 31 dysfunctional text lines. This file was the input for Component 2. Then we used Component 2 and Component 3 of the OIRE method to process this input file.

Output: The visualization of the results of Components 1 and 2 when applying Use Case 2 is shown in Figure 42. Table 34 shows the Kano-like classification output of Use Case 2.

Table 32. The confusion matrix used to assess the performance of supervised machine learning algorithms

		Predicted Condition	
		Functional	Dysfunctional
True Condition	Functional	True Functional (TF)	False Functional (FF)
	Dysfunctional	False Dysfunctional (FD)	True Dysfunctional (TD)
FPV		TF / (TF+FF)	
DPV		TD / (FD+TD)	
Accuracy		(TF+TD) / (TF+FF+FD+TD)	

Table 33. Performance of supervised machine learning algorithms

Method	Indicator	Value (%)	Standard Deviation (σ)
Naive Bayes	FPV	0	0.458
	DPV	88	
	Accuracy	66	
MaxEnt	FPV	100	0.282
	DPV	44	
	Accuracy	66	
Decision Trees	FPV	33	0.115
	DPV	55	
	Accuracy	50	
Random Forest	FPV	66	0
	DPV	66	
	Accuracy	66	
SVM	FPV	100	0.176
	DPV	66	
	Accuracy	75	

To assign a Kano category to the feature "Alipay", we applied Component 3. The results are shown in Table 34. The first row of Table 34 shows the probability distribution of the Kano categories when applying the traditional Kano table (5*5 matrix as shown in the upper part of Figure 43). The probability of category "I" is rather high (62%). We found that this is mainly due to the fact that the proportion

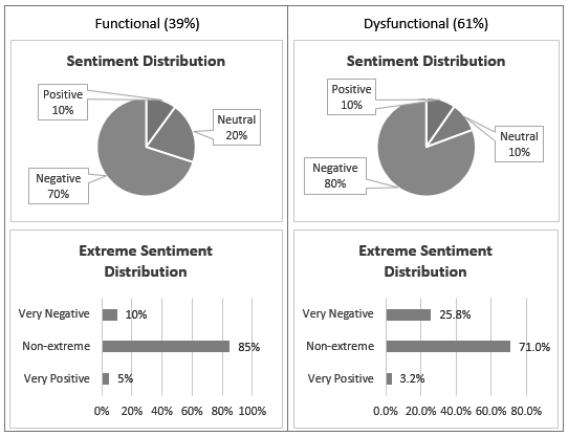


Figure 42. Visualization of the results of Components 1 and 2 of Use Case 2

of category "I" entries in the traditional Kano table is relatively high (9 out of 25). Considering that the application of the traditional Kano model is based on face-to-face interviews where the user is passive when expressing opinions, this result might not be surprising. However, unlike the traditional Kano model, the Kano-like model is based on comments that the users themselves actively publish on the Internet. We believe that when one wants to take the initiative to express an opinion rather than answer a question passively, the willingness to express an emotion is stronger. Therefore, we simplified the traditional Kano table from a 5*5 matrix to a 3*3 matrix (Kano-like table). Figure 43 shows how we modified the Kano table. We classified emotion ratings into three levels: Negative, Neutral, and Positive. When we ran Component 3 using the Kano-like table, the results looked like what is shown in the second data row of Table 34. We used red and italics to represent the results processed by using a 3*3 matrix Kano-like table. We see from Table 34 that when we used the traditional Kano table (5*5 matrix), the Kano-like classification output was that the probability of this feature being is classified into category "M" was 22.5%. For category “A”, the probability was 3.6%, for “O” 1.4%, and for “I” 62%. When we used the Kano-like table (3*3 matrix), the Kano-like classification output was that the probability of this feature being classified into category "M" was 37.8%. For category “A”, it was 2.3%, for “O” 19%, and for “I” 4.5%.

Table 34. Output of Use Case 2

Kano-like Classification		O	A	M	I	R
Probability (%)	5*5 matrix	1.4	3.6	22.5	62.0	10.5
	3*3 matrix	19.0	2.3	37.8	4.5	36.4

(5*5 matrix)		Dysfunctional Question				
		2	1	0	-1	-2
Functional Question	2	Q	A	A	A	O
	1	R	I	I	I	M
	0	R	I	I	I	M
	-1	R	I	I	I	M
	-2	R	R	R	R	Q

(3*3 matrix)		Dysfunctional Question		
		Positive (2, 1)	Neutral (0)	Negative (-1, -2)
Functional Question	Positive (2, 1)	Q	A	O
	Neutral (0)	R	I	M
	Negative (-1, -2)	R	R	Q

Figure 43. Modified Kano table

6.3.3. Application of Use Case 3

Regarding Use Case 3, we chose reviews of WeChat, again collected from the Kuchuan platform. At the time of our analysis, there were 31,966 reviews.

The features we chose as the example feature for implementing Use Case 3 were "Ads blocker", "Fingerprint", "Three days visible", and "Top circle post". Each feature had more than 20 lines of review text. This was the minimum we considered necessary for conducting a proper analysis. Since these features are proprietary features offered by the software developer, the synonym tool was again not needed in this situation. Next, we searched all the sentences containing "advertising" and "tourism" in the reviews. In the end, we extracted 68 lines of text related to "Ads blocker", 122 text lines related to "Fingerprint", 25 text lines related to "Three days visible", and 56 text lines related to "Top circle post". Since the text lines were in Chinese, we translated them into English.

Input: The input of Use Case 3 consisted of four text files containing text lines relating to four features of WeChat, i.e., "Ad blocker", "Fingerprint", "Three days visible", and "Top circle post".

Approach: We followed the same approach used in Section 6.3.2 and implemented Component 1-3 on each input file in parallel.

Table 35. Output of Use Case 3

Kano-like Classification		O	A	M	I	R
Probability (%)	Ads blocker	8.3	3.4	28.7	6.8	46.8
	Fingerprint	14.7	29.4	9.9	20.0	26.0
	Three days visible	8.8	0.0	26.5	0.0	64.7
	Top circle post	31.2	35.0	8.5	9.5	15.8

Output: Table 35 shows the Kano-like classification output when applying Use Case 3. Using the Kano-like table (3*3 matrix), the features "Ad blocker" and "Three days visible" had the highest probability of being classified into category

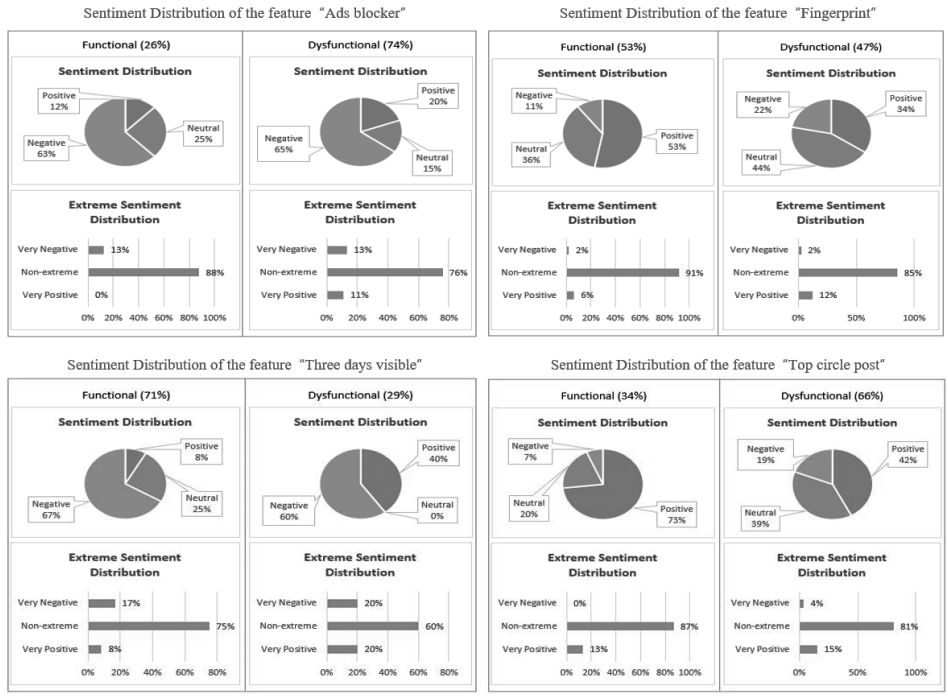


Figure 44. Visualization of the results of Components 1 and 2 of Use Cases 3

"R" (46.8%, resp. 64.7%), and the features "Fingerprint" and "Top circle post" had the highest probability of being classified into category "A" (29.4%, resp. 31.2%). The visualization of the results of Components 1 and 2 is shown in Figure 44.

6.4. Threats to Validity

We identified several threats to validity – to construct validity, internal validity, and external validity – which will be discussed in this section.

6.4.1. Construct Validity

The use cases presented in this validation study were not proposed by real product managers and product owners but rather by the authors, who put themselves into the shoes of product managers and product owners. Therefore, the use cases presented in this validation study might not represent the real perspective of product managers and product owners. This may lead to a gap between our analysis and the actual situation. However, in order to mitigate this threat, we showed our use cases to a small sample of product managers. The members of this convenience sample confirmed that the use cases made sense to them.

6.4.2. Internal Validity

Many decisions made during the application and validation of the OIRE method were choices made by the author, who might be biased in their view of what is typical and what is adequate. Researcher bias might also apply to the interpretation of the results achieved.

When the amount of input data is limited, the amount of test data that can be used in Component 1 of the OIRE method is also limited. This affects the performance and choice of the best machine learning algorithm, and thus may have an impact on the application of the model.

Because the OIRE method was designed on the basis of text mining of English-language texts, translation was required for implementing the model on input containing non-English text. The translation process will naturally affect the accuracy of the OIRE method to some extent.

To check the accuracy of the individual components of the OIRE method, we used the PPS method [88] to check the performance of 20% of the text lines (not fewer than five lines of text) contained in the input. Then we manually checked the accuracy of the result achieved by each component. Tables 36 to 38 show the accuracy (Accuracy = Correct prediction / Samples * 100%) of the results of applying Use Cases 1 to 3.

According to our implementation approach for the OIRE method, which we introduced in Section 6.3.2, to enhance reliability, we may discard some text lines after running Component 1, so the number of text lines contained in the result is lower than that in the input.

We see from Table 36 that the accuracy of Application 1 (only implementing Component 2) was 80%. The accuracy of the output of Application 2 is shown in Table 37. It shows that the accuracy of implementing Component 1 was 70%, while the accuracy of implementing Component 2 was 80%. The overall accuracy of Application 2 was 60%. The accuracy of the output of Application 3 is shown in Table 38. We see from Table 38 that the accuracy of implementing Component 1 was between 63% and 83%, while the accuracy of implementing Component 2 was between 67% and 88%. The overall accuracy of Application 3 was between 50% and 61%.

Table 36. Accuracy of the application results of Use Case 1

Application Name	Feature Name	Lines of Text	Samples	Correct Prediction	Accuracy
Pinterest	already pinned	151	30	24	80%

6.4.3. External Validity

To minimize the impact of the input on the experiments, we randomly selected the objects of analysis, including the apps and the features of the selected apps. However, the number of use cases is rather small, which limits the representativeness

Table 37. Accuracy of the application results of Use Case 2

Application Name	Feature Name	Lines of Text	Samples	Correct Prediction		Accuracy	
Meituan	Alipay	51	10	Component 1	7	70%	60%
				Component 2	8	80%	

Table 38. Accuracy of the application results of Use Case 3

Application Name	Feature Name	Lines of Text	Samples	Correct Prediction		Accuracy	
WeChat	Ad blocker	62	12	Component 1	9	75%	50%
				Component 2	8	67%	
	Fingerprint	88	18	Component 1	15	83%	61%
				Component 2	12	67%	
	Three days visible	17	5	Component 1	4	80%	60%
				Component 2	4	80%	
	Top circle post	41	8	Component 1	7	63%	50%
				Component 2	8	88%	

of our results.

6.5. Discussion

Although the output of the OIRE method will almost always produce results that differ from those of the traditional Kano model, our approach could be considered as providing richer output. For example, let's assume an extreme case where you get 100 paired answers, with 40 answers leading to category "I", 39 answers leading to category "M", and 21 answers leading to category "A". The output of the traditional Kano model would be that this feature should be categorized as "I". If only this final categorization is conveyed, one will not know that 39% of the interviewees considered this feature to be "M" and 21% of the interviewees considered this feature to be "A".

We know that software organizations need to consider a variety of information (e.g., cost) when making decisions. The end users' perception of a product's features is just one of many attributes. So the limitation of the OIRE method is that it cannot provide users with all the information that supports decision-making regarding the prioritization of features. Also, when we talked to product owners and product managers as well as owners of companies, we received the feedback that the idea of the OIRE method is interesting, and in the context of the rapid development of methods in artificial intelligence and data mining, the OIRE method may have the potential for a variety of extensions. On the other hand, however, based on their work experience, they believed that the usefulness of the OIRE method highly depends on the quality (and amount) of review texts, and that this may be a limitation. One cannot expect users to write comprehensive reviews. They will typically focus on aspects of an app that they perceive as either very

exciting or disappointing; in addition, they are subjective (i.e., not necessarily fair and unbiased). They might even occasionally give negative reviews just because they are in a bad mood.

6.6. Conclusion

Overall, the accuracy of the OIRE method was satisfactory. The accuracy of Component 1 and Component 2 of the OIRE method exceeded 63% and the highest value was 88%. The overall accuracy of the OIRE method was between 50% and 61%.

We also demonstrated that the OIRE method can produce results for typical use cases of product managers and product owners. Due to the fact that no real product managers and product owners used the OIRE method yet during this study stage, we cannot conclusively state how easy to use the method would be perceived, and how useful its results would be considered. Thus, in the follow-up chapter, we will conduct a more comprehensive study to evaluate the OIRE model.

Such a study would also be useful to help us further improve the way the results of the various components of the OIRE method are presented in OIRE-S. For example, by checking the inputs and outputs when applying Use Cases 1 to 3, we found that when users comment on a feature, they may either evaluate the feature itself or the way the app implements the feature. For example, regarding the feature "Ad blocker" analyzed in the application of Use Case 3, the users were not necessarily dissatisfied with the feature itself, but rather with the way the app implements the feature. Therefore, we believe that showing more detailed outputs of each component of the OIRE method to users would help improve their comprehensive judgment and help them draw more accurate conclusions about apps and individual app features.

6.7. Summary

In this chapter, we first discussed the selection of the input source. We found that the input collected from app reviews was more suitable for the OIRE method. Next, we presented three typical use cases and a proof-of-concept of the method demonstrating the applicability of the OIRE method. We used real-world data collected from the Internet as the input source. Our findings were as follows: The accuracy of Component 1 and Component 2 of the OIRE method exceeded 63%, with the highest value being 88%. The overall accuracy of the OIRE method was between 50% and 61%. For the selected use cases, the OIRE method had the potential to help stakeholders make better-informed RE decisions. In the next chapter, we will present how we evaluated the OIRE method in industry using OIRE-S.

7. EVALUATION

In previous chapters, we introduced the design, implementation, and preliminary validation of the OIRE method. We also developed a prototypical web application based on the OIRE method, OIRE-S, to provide tool support for users of the OIRE method. In this chapter, we will evaluate the OIRE method using OIRE-S. We conducted a case study with two Chinese companies that plan to have software apps developed by suppliers. In addition, we conducted an interview study, interviewing two other stakeholders about the case study. The case study tried to solve real problems. The interview study did not include any actual application of OIRE and OIRE-S. We used Support Vector Machine (SVM[1]) as the default machine algorithm in the "Sentence Classification" module of OIRE-S since SVM showed stable performance in previous experiments.

This chapter is structured as follows. In Section 7.1, we will describe the design of this evaluation study. In Section 7.2, we will outline the results of the case study and the interview study we conducted to evaluate the OIRE method. In Section 7.3, we will discuss the results of the evaluation study. In Section 7.4, we will present and discuss threats to validity. Section 7.5 concludes this evaluation study.

7.1. Study Design

In Chapter 6, we reported an initial validation of the OIRE method based only on a simulated use case study [112]. The objective of the evaluation studies presented in this chapter was to perform a realistic evaluation of the OIRE method together with its prototypical tool support OIRE-S in industry. For this purpose, we carried out a case study and an interview study. We analyzed the usefulness of the OIRE method together with software development stakeholders. We designed the study in line with the guidelines for case study research provided by Runeson and Höst [84]. Figure 45 shows the elements of the study.

As shown in Figure 45, we first identified the research question. Then we designed a case study and an interview study in an industrial setting. The case study was conducted in two Chinese companies in which we evaluated the OIRE method. We presented the OIRE results to the stakeholders in the two case companies and discussed the findings with them. Then we presented the OIRE method and the case study results to two other stakeholders and interviewed them to get further feedback. We agreed with all the stakeholders that the information provided by them would be anonymized.

7.1.1. Research Question

The research question was formulated as follows:

[1] We use the package "e1071" developed by TU Wien, Austria

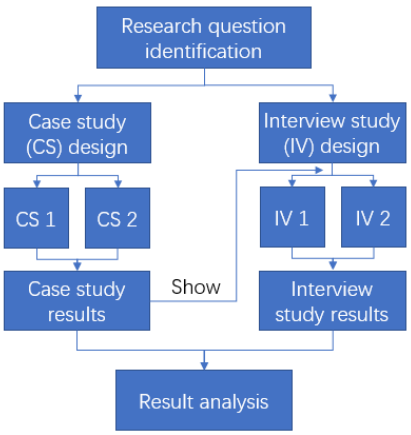


Figure 45. The elements of the study

Qu 7.0: Is the OIRE method useful to decision-makers in industry?

7.1.2. Case Study Design

The case study was conducted to explore how stakeholders without software development competence would appreciate the OIRE method.

Case and Unit of Analysis. We conducted the case study in China, involving two case companies as the unit of analysis. The choice of the case companies was purely opportunistic. The author had established contact with these companies previously. One stakeholder each was involved per case, i.e., customer C1 (in case 1) and customer C2 (in case 2). We call the stakeholders "customers" because they plan to develop or contract the development of a new app and would like to better understand what users of similar existing apps think about certain features. To provide the requested information to the stakeholders, we employed the OIRE method and its tool support OIRE-S.

Procedure. The evaluation procedure followed in each case consisted of eight steps: 1) Identify existing competitors; 2) Collect Chinese-language review data; 3) Identify features; 4) Extract feature-related text from reviews (and translate into English); 5) Select analysis function ("Sentiment analysis" function and/or "Kano-like analysis" function); 6) Process data and generate analysis results; 7) Present analysis results (and translate into Chinese); 8) Collect stakeholder feedback, translate feedback into English, and summarize.

The input data for each case was collected from the online review platform, Kuchuan[2].

After completion of the case study, the results of the case study served as input material for two follow-up interviews. The outcomes from the case study will be presented in Section 7.2 (Results).

[2]http://android.kuchuan.com/page/detail/index (accessed: 15-October-2018)

7.1.3. Interview Study Design

The interview study was conducted to explore how stakeholders in software companies would appreciate the OIRE method. Note, however, that, unlike the case study, the interview study did not include any actual application of OIRE and OIRE-S. In this section, we will characterize the interview participants and then describe the interview procedure.

Participants. We conducted two interviews with stakeholders in the Chinese software industry. The first stakeholder, P, was a product manager in a large software company. He had worked as a software product manager for over three years. The second stakeholder, M, was co-founder (and manager) of a start-up information technology company. One participant was involved in each interview, i.e., P in interview 1 and M in interview 2.

Procedure. To provide the required information to the participants, we communicated via phone and online chat tools. The procedure of each interview consisted of three steps: 1) Introduce the OIRE method; 2) Show the case study results from C1 and C2 (in Chinese); 3) Interview the participants, collect their responses, translate the responses into English, and summarize.

The interviews consisted of three main questions: 1) In the process of software development, what difficulties or problems do you face? 2) After knowing about the OIRE method, what do you think about the OIRE method? Do you think that the OIRE method would help you to solve the above problems, and why? 3) After knowing the case study results, how do you evaluate the OIRE method?

In the interview, we expanded the three main interview questions into several open questions. Each interview lasted about 15 minutes.

7.2. Results

This section reports the results of the case study and the interview study. We will first present the execution of the case study and its results as well as the feedback from the stakeholders in the case companies regarding Qu 7.0 in Section 7.2.1. Next, we will present the execution of the interview study and the feedback from the participants regarding Qu 7.0 in Section 7.2.2.

7.2.1. Case Study

In each case of the case study, we first communicated with C1, respectively C2, to understand their business situation and the basic requirements for their planned applications. Then we gathered the input data and used OIRE-S as the tool support of the OIRE method to analyze it. After the analysis, we presented the results and our suggestions to C1 and C2. Finally, we collected feedback about the usefulness of the OIRE method from C1 and C2. Note that OIRE-S was not given to the stakeholders but was used by the author of this thesis to answer the questions of the stakeholders when using the OIRE method.

Case 1: Shipping System. Stakeholder C1 was working for a shipping company and planned to develop an online shipping app. We designed Case 1 to help C1 understand better whether their ideas about the features to be included in the planned software app have the potential to make it a success.

Based on their description of the context, C1 put forward the following list of features c1f.1 to c1f.8 they thought should be included in the anticipated app:

c1f.1) Based on "Automatic Identification System" and "Vessel Traffic Service", the app shall provide enterprises with information about the position of a ship.

c1f.2) Based on a ship's location on the nautical chart, the app shall show information about weather phenomena such as typhoons, etc.

c1f.3) Login feature.

c1f.4) The core content of the platform shall consist of displaying cargo pallets and fuel information on a nautical chart, providing fuel stations, fuel prices, inventory content, etc.

c1f.5) Simple algorithms shall help businesses to calculate profits, shipping rates, wages, turnover, etc.

c1f.6) Other services, such as seafarer recruitment, crew qualification inquiry, etc. shall be offered.

c1f.7) The app shall post news about maritime and shipping companies.

c1f.8) The app shall offer advertisement, for example, by insurance companies or second-hand equipment trading companies, etc.

In the following, we will describe what happened during the eight steps of the case study procedure for Case 1.

Step 1: Identify existing competitors. The business area is very mature, with a relatively small number of competitors. We asked C1 to select the most interesting competing apps. C1 provided four target apps: CX[3], CLL[4], XCW[5], and CBR[6].

Step 2: Collect Chinese review data. To get the input data, we retrieved user reviews related to the four selected apps from the Chinese app-monitoring platform Kuchuan. Since we found only two reviews about XCW, we removed this app from the list of existing competitors. Including all reviews of the other three apps from the Android platform and the Apple store in the past year, we collected 110 lines of text in total. Because of the small number of relevant text lines, we merged the data from the three competitive apps into one file, representing one synthetic competitive app.

Step 3: Identify features. We identified features in two ways. First, we analyzed the set of proposed features provided by C1. C1 proposed twelve features, i.e., "nautical chart", "weather", "typhoon", "login", "cargo pallets", "fuel",

[3]http://www.shipfinder.com/

[4]https://xy.ship56.net/

[5]http://www.xcw9898.com/

[6]http://m.msa.gov.cn

"calculation", "recruitment", "qualifications", "post maritime", "shipping news", "advertising".

Second, we used online NLP analysis platforms to analyze the original 110 lines of text and extracted the most frequent words as potential features. Since the input text was in Chinese, we used a Chinese online NLP platform[7]. The platform offered the 15 most frequent items, i.e., the Chinese words corresponding to "berth", "garbage", "update", "maritime affairs", "import and export", "edition", "maritime bureau", "port", "always", "system", "change", "visa", "thing", "departure", and "software". We first removed words from the 15 most frequent terms that cannot be considered as proper features, such as "always", "system", "change", "thing", "software", etc. Four out of the 15 terms remained, i.e., the Chinese words for "berth", "port (report)", "update", and "visa". We added these four items to the originally proposed twelve features, thus ending up with a list of 16 (=12+4) potential features.

Step 4: Extract feature related text from reviews. Then we manually did pattern matching on the original 110 lines of text and extracted feature-related text lines as input data for further analysis. Note that when we did the pattern matching, we included feature synonyms that we had derived manually[8]. In total, six features were retained. Two of the twelve features provided by C1 were found in the text lines, i.e., "login" and "nautical chart". As OIRE-S uses an English-language analysis tool, the Chinese feature-related text had to be translated into English using an online translator[9].

Step 5: Select analysis function. Two analysis functions are offered by the OIRE method/ OIRE-S, i.e., "Sentiment analysis" and "Kano-like analysis". After communicating with C1, we decided to use the "Sentiment analysis" function to analyze the polarity of the input data and to determine to what extent these features were liked or disliked by end users.

Step 6: Process data and generate analysis results. In order to check whether the overall sentiment regarding the apps was consistent with that regarding the identified six features, we analyzed whether the sentiment distribution of the text related to the six extracted features was consistent with the sentiment distribution exhibited by all 110 lines of text. Therefore, we performed the sentiment analysis separately for the total 110 lines of text (input 1) and for the 84 text lines related to the six features (input 2).

Input 1: One file with 110 text lines corresponding to the reviews of the three competitor apps.

Input 2: Six files with 84 text lines in total relating to the six extracted features.

Approach: We first cleaned the data. We changed all capital letters to lowercase and removed punctuations and strange characters, such as "@:(". Then we

[7]https://bosonnlp.com/product/intro (accessed: 22-October-2018)

[8]If the feature list had already been provided in English, we had used an English-language online synonym dictionary(http://www.thesaurus.com/).

[9]https://fanyi.baidu.com/ (accessed: 15-October-2018)

used the "Sentiment mining" module of OIRE-S to process the inputs.

Output 1: Table 39 shows the output after applying the "Sentiment mining" module to input 1. We see from Table 39 that 14% of the text lines showed Very Negative emotions, 55% Negative emotions, 14% Neutral emotions, 13% Positive emotions, and 5% Very Positive emotions.

Output 2: Table 40 shows the output after applying the "Sentiment analysis" function to input 2. We see from Table 40 that the feature "Visa" has the highest number (25%) in the "Very Negative" column. The feature "Nautical chart" has the highest number (75%) in the "Negative" column. The feature "Berth" has the highest number (9%) in the "Neutral" column. The feature "Login" has the highest number (36%) in the "Positive" column, and the feature "Visa" has the highest number (25%) in the "Very Positive" column.

Table 39. Overall sentiment distribution

Features	Number of Text Lines	Proportion				
		Very Negative	Negative	Neutral	Positive	Very Positive
Overall	110	14%	55%	14%	13%	5%

Table 40. Sentiment distribution of features

Features	Number of Text Lines	Proportion				
		Very Negative	Negative	Neutral	Positive	Very Positive
Berth	11	0%	64%	9%	27%	0%
Login	14	7%	57%	0%	36%	0%
Nautical chart	4	0%	75%	0%	25%	0%
Port (report)	10	20%	70%	0%	0%	10%
Update	37	11%	68%	8%	8%	5%
Visa	8	25%	50%	0%	0%	25%
Total	84	-	-	-	-	-

Step 7: Present analysis results. Based on the data shown in Tables 39 and 40, we summarized the analysis results and generated a report as feedback to C1. This report was translated into Chinese and provided the basis for our discussions with C1. Below we list the key findings c1r.1 to c1r.4, which were communicated to C1:

c1r.1) Table 39 shows that overall, negative emotion (14%+55%=69%) was greater than positive emotion (13%+5%=18%). Based on this, we concluded that the end users are not satisfied with the three existing (competitor) apps.

c1r.2) Table 40 shows that the number of review text lines related to the six features of interest accounted for more than 75% (84/110*100%) of the total review text. This indicates that the users' concern regarding the six features was much greater than that for other features. Based on this, we concluded that our analysis is relevant.

c1r.3) From Table 40, we see that the negative emotions related to the six features was greater than the positive emotions. This shows that the dissatisfaction with the six features was the main reason for end users' dissatisfaction with the three apps. Especially for the features "Nautical chart", "Port (report)", "Update",

and "Visa", the proportion of negative emotions (Very Negative + Negative) was even higher than the proportion of overall negative emotions (See Table 40, numbers in red). Based on this, we concluded that the identified six features of interest were either not well implemented or unwanted by the end users.

c1r.4) We found that the features we extracted were very different from the main twelve features proposed by C1. Only the feature "Nautical chart" and "login" were the same. Due to the limited amount of data, it was difficult to detect new features that would increase the end users' satisfaction with the planned app. However, based on the analysis, we were able to tell C1 which features would dissatisfy the end users. Thus, these negatively perceived features in the competing apps should either be avoided or improved/redesigned in such a way that they are perceived positively. We also suggested that C1 should consider other features offered by the three existing competing products.

Step 8: Collect stakeholder feedback, translate feedback into English, and summarize. We presented the analysis report to C1 and discussed the results. We summarize the feedback from C1 as follows:

c1s.1) C1 could see Tables 39 and 40 as part of the report. C1 appreciated that, unlike previous reports on requirements analysis, our report was quantitative and not qualitative. This made our report more "convincing" to C1. C1 perceived the analysis report to be "professional" and "reliable".

c1s.2) According to C1, the results of the analysis seemed to be reasonable. Features having many user reviews represent a higher degree of concern among users. Based on this, C1 considered the analysis results c1r.1, c1r.2, and c1r.3 to be reasonable.

c1s.3) C1 agreed with the analysis result c1r.4 and considered it as useful input to reflect upon their company's own ideas about the features to include in the planned app.

c1s.4) C1 would have liked to see new ideas that existing applications do not provide. However, since we were only able to retrieve a small set of relevant reviews, no feature requests were found among them.

Case 2: Fitness Application. Stakeholder C2 was co-founder of a fitness company with two fitness centers and more than 400 regular clients. C2 believed that fitness activities should not be confined to fitness centers but could happen in everyday life, for example while hiking, boating, and even shopping. Recording a user's activity at any time helps to analyze the user's fitness data. C2 thought that developing a fitness application would help their company serve existing customers and attract new ones at the same time.

C2 shared their ideas about features they planned to have in a new app:

c2f.1) Record sports tracks.

c2f.2) Analyze sports data, record changes.

c2f.3) Record and analyze health data, such as heart rate.

c2f.4) Store all data online.

c2f.5) Provide online sports courses.

c2f.6) Teach fitness knowledge.

c2f.7) Help users develop a scientific fitness training plan.

In the following, we will describe what happened during the eight steps of the case study procedure for Case 2.

Step 1: Identify existing competitors. We chose Codoon[10] as the existing main competitor app. In China, Codoon is a well-known sports application. The number of users exceeds 150 million. In 2015, the market share of Codoon in the industry was above 50%.

Step 2: Collect Chinese review data. To get the input data, we downloaded reviews of Codoon from the Chinese app-monitoring platform Kuchuan. Including all reviews of Codoon from the Apple store in the past 180 days, we collected 32,610 lines of reviews in total.

Step 3: Identify features. On Codoon's official website, there is a very detailed description of its functions. For this case, in addition to considering the potential features proposed by C2, we browsed the introductory description of Codoon on the official website to identify features.

C2 provided seven potential features, i.e., the Chinese words corresponding to "Sport tracks", "Sports data", "Health data", "Online storage", "Sports courses", "Fitness knowledge", and "Training plan". We manually extracted 16 additional features from the app descriptions, namely the Chinese words corresponding to "GPS / tracking", "Online shopping mall /shopping", "Social networking", "Online Marathon", "Guidance / demonstration", "Red packets /rewards", "Running shoes", "Map", "Trajectory", "Interface / layout", "Heart rate", "Sports data / index", "Video / audio", "Step counter", "Training courses", and "Sports knowledge". Next, we combined similar features. For example, "Sports tracks" and "GPS / tracking" are similar features.

At the end of this step, we had identified 17 features, i.e., "GPS / tracking", "Online shopping mall /shopping", "Social networking", "Online marathon", "Guidance / demonstration", "Red packets / rewards", "Running shoes", "Map", "Training plan", "Trajectory", "Interface / layout", "Heart rate", "Sports data / index", "Video / audio", "Step counter", "Training courses", and "Sports knowledge".

Step 4: Extract feature related text from reviews. Then we manually did pattern-matching on the original 32,610 lines of reviews and kept only those sentences that contained any of the 17 features. This left us with 494 lines of text. These lines of text were used as the input data for further analysis. Next, we translated the Chinese feature-related text into English using an online translator[11].

Step 5: Select analysis function. We extracted more features than C2 had proposed. In order not to increase future software development costs too much, C2 wanted to confirm which features were most effective in improving end users'

[10] https://www.codoon.com/ (accessed: 15-October-2018)

[11] https://fanyi.baidu.com/ (accessed: 15-October-2018)

satisfaction. In view of this, we decided that the "Kano-like analysis" function would provide the desired results.

Step 6: Process data and generate analysis results. In this step, we used the "Kano- like analysis" function to classify the 17 features into the Kano classification.

Input: 17 files with 494 text lines relating to the 17 extracted features. Of the 494 text lines, 449 text lines were related to exactly one feature, 44 text lines were wrelated to two features, and one text line was related to three features.

Approach: We first cleaned the data. We changed all capital letters to lowercase and removed punctuations and other strange characters, such as "@:(". Then we used the "Kano-like analysis" function of OIRE-S (Component 1-3 of the OIRE method) to process the inputs.

Output: Table 41 shows the Kano-like category classification after we applied the "Kano-like analysis" function to the input. In Table 41, we arranged the 17 features in descending order according to the value of column "O". In Table 41, features marked in red are those proposed by the customer. We see from Table 41 that the probability of features F12, F8, F17, F11, F10, and F9 being classified into category "O" is higher than that of being classified into other categories. The probability of features F4 and F14 being classified into category "M" is higher than their probability of being classified into other categories. The probability of features F13, F15, F1, F2, F16, F6, and F7 being classified into category "R" is higher than their probability of being classified into other categories.

We also see from Table 41 that features F3 and F5 are not classified into any category. To investigate the reason for this, we checked the sentiment distributions of F3 and F5 (see Table 42). According to the "Kano-like analysis" function, when the sentiment polarity of both functional and dysfunctional types of input is Positive or Very Positive at the same time, the input data is treated as contradictory and is discarded (cf. Section 6.3.2). We see from Table 42 that the text lines related to F3 and F5 were all identified as Very Positive or Positive. This is why the Kano-like model did not categorize these two features. In addition, we see that the number of positive reviews of F3 of the functional type (1+7=8) equals that of those of the dysfunctional type (3+5=8). The number of positive reviews of F5 of the functional type (7+22=29) is higher than that of those of the dysfunctional type (5+9=14). Therefore, we think that the users' satisfaction and dissatisfaction with F3 is more balanced, and that users consider the existence of F5 more satisfying.

Step 7: Present analysis results. Based on the data shown in Tables 41 and 42, we summarized the analysis results and generated a report as feedback to C2. This report was translated into Chinese and formed the basis for our discussions with C2. Below we list the key findings c2r.1 to c2r.5 communicated to C2:

c2r.1) From Table 41, we see that four features proposed by C2, i.e., F1, F13, F15, and F16, were classified into category "R".

c2r.2) Features classified into category "R" should not be developed. These features are F1, F2, F6, F9, F13, F15, and F16. Implementing these features may

Table 41. Kano-like category classification

Feature ID	Feature Name	Number of Text Lines	Probability (%)				
			O	A	M	I	R
F 3	Social networking	16	NA	NA	NA	NA	NA
F 5	Guidance / demonstration	43	NA	NA	NA	NA	NA
F 12	Heart rate	12	83.3	0	0	0	16.7
F 8	Map	15	60.6	0	24.2	0	15.2
F 17	Sports knowledge	21	60	0	10	0	30
F 11	Interface / layout	60	51.4	0	4.2	0	44.4
F 10	Trajectory	68	48	19.2	10.4	4.2	18.2
F 9	Training plan	16	47.4	0	5.3	0	47.3
F 13	Sports data / index	20	33.3	0	6.7	0	60
F 15	Step counter	100	28.9	11.6	4.4	1.8	53.3
F 1	GPS/ tracking	25	28.3	16.2	7.1	4	44.4
F 14	Video / audio	39	21.8	36.4	3.6	6.1	32.1
F 2	Online shopping mall /shopping	7	20	0	0	0	80
F 16	Training courses	61	15.9	23.9	4	6	50.2
F 4	Online marathon	15	0	100	0	0	0
F 6	Red packets /rewards	13	0	27.3	0	9.1	63.6
F 7	Running shoes	9	0	0	0	0	100

lead to high dissatisfaction among users. By checking users' reviews further, we found that the existence of F2 caused users to be dissatisfied, while the dissatisfaction with other features was due to poor performance. Hence, when developing these features, C2 should be very careful and strictly guarantee the quality of development; also, F2 should not be developed.

c2r.3) Features classified into category "O" should be developed first. These are F8, F9, F10, F11, F12, and F17. The existence or good performance of these features is of great help to improve product satisfaction. If these features are missing or perform badly, the degree of product dissatisfaction will increase significantly.

c2r.4) Features classified into category "A" should be developed as much as possible if funds allow. These features are F4 and F14. The existence and good performance of these features are essential for improving product satisfaction.

c2r.5) Features F3 and F5 could not be categorized automatically. When inspecting the results of the sentiment polarity analysis (in Table 42), we believe that the existence of F3 would not help improve user satisfaction. The existence of F5, on the other hand, may improve user satisfaction. Thus, we recommended developing F5, but not giving priority to the development of F3.

Step 8: Collect stakeholder feedback, translate feedback into English, and summarize. We presented the analysis report to C2 and discussed the results. We summary C2's feedback as follows:

c2s.1) Since C2 did not have any knowledge about the traditional Kano model, we had to explain the meaning of the analysis results. When C2 understood the

Table 42. Sentiment distribution of F3 and F5

Feature ID	Feature Name	Type	Very Negative	Negative	Neutral	Positive	Very Positive
F 3	Social networking	Functional	0	0	0	1	7
		Dysfunctional	0	0	0	3	5
F 5	Guidance / demonstration	Functional	0	0	0	7	22
		Dysfunctional	0	0	0	5	9

exact meaning of the analysis results, they qualified our analysis as "professional" and also said that this analysis result was "definitely useful" for software development.

c2s.2) C2 said that the analysis results were partly "beyond their expectations". For example, they had never considered the feature "Online marathon" before.

c2s.3) C2 said that our analysis pointed out important features that had been overlooked. For example, they agreed that the feature "Interface" was important, but had not been thought of before.

c2s.4) C2 was eager to find new features that were not provided in other similar applications. However, based on our analysis report, we could not tell which features already existed and which were completely new features.

c2s.5) C2 mentioned that they did not feel that they would have to fully comply with our suggestions, but believed that our suggestions should definitely be taken into account.

7.2.2. Interview Study

We interviewed two other stakeholders, M and P. M and P were not involved in the case study, but agreed to be interviewed to help us understand the problems that product managers and small software company owners are facing. We were also interested in the difference between the existing requirements analysis processes used in the companies of the interviewees and the use of the OIRE method. We hoped to get meaningful feedback from them. In the following, we will present the results of each interview step (cf. Section 7.1.3). The three main interview questions were: 1) In the process of software development, what difficulties or problems do you face? 2) After knowing about the OIRE method, what do you think about the OIRE method? Do you think that the OIRE method would help you to solve the above problems, and why? 3) After knowing the case study results, how do you evaluate the OIRE method?

Interview 1: Interview with P.

Step 1: Introduce the OIRE method. In this step, we introduced the OIRE method to P, including the purpose, functions, and components of the OIRE method. P could ask us questions about the OIRE method at any time during the interview, and we answered any questions until they said they understood the OIRE method.

Step 2: Show the case study results that were developed with C1 and C2 (in Chinese). In this step, we showed the case study reports (in Chinese) to P and explained the content. P could ask us questions about the report at any time during

the interview, and we answered any questions until they said they understood the content of the reports.

Step 3: Interview participants collect participants' responses, translate the responses into English, summarize. In this step, we interviewed P by asking several open questions derived from the three main interview questions. We collected and translated the answers and summarize them as follows:

i1a.1) P indicated that end users and salespeople had so much initiative that they sometimes voiced unreasonable requirements on a product regardless of whether this was valuable or not. P even said that "customers are not reliable".

i1a.2) P indicated that frequent changes in the requirements engineering process of their company caused duplication of work. Changes in internal requirements could usually be avoided through timely communication. However, when changes come from the external market, such as upgrading and updating of competing products, P believed changes in requirements to be inevitable, resulting in waste of time, waste of money, and prolonged product development cycles.

i1a.3) P indicated that, by implementing the OIRE method, product managers could reject some meaningless requirements, such as those relating to poorly performing features existing in other products. Product managers could directly advise users to give up meaningless ideas.

i1a.4) P indicated that implementing the OIRE method could speed up the requirements prioritization process and shorten the development cycle.

i1a.5) P believed that it would be easier for a report with "numbers" to win the trust of users. This was considered a big advantage of OIRE analysis reports.

i1a.6) P told us that the existing requirements analysis process required several product managers to form a team. Each manager in this team conducted user surveys, competitor analyses, or data analyses separately. At least one team meeting was needed to discuss ideas, and each complete requirements analysis process might take a whole week. P believed that the OIRE method would require less manpower and less time than the existing requirements analysis processes. This would be helpful for enterprises to save money and time.

i1a.7) Although P believed the OIRE method to be useful, they insisted that user comments were "unreliable". According to their experience, it was very common for software companies to make fake comments in order to improve their software ratings. Therefore, the foundation for the use of the OIRE method might be flawed. P believed that the existing requirements analysis process had an advantage in terms of reliability, although it was time-consuming and laborious. However, P agreed that the OIRE method could be an effective complement to existing requirements analysis methods (interviews, questionnaires, log mining, brainstorm, etc.), especially for start-ups with limited resources.

i1a.8) P suggested that the analysis of log records was now an important part of the requirements analysis process and that it would be very useful if the OIRE method could help with log analysis.

i1a.9) At the end of the interview, P said that the OIRE method was very in-

teresting. P assumed that, in the context of the rapid development of machine learning and artificial intelligence techniques, the idea of the OIRE method may have potential extensions in many other fields, for example in human resources management.

Answers i1a.1 and i1a.2 relate to the first main interview question. Answers i1a.3 and i1a.4 relate to the second main interview question. Answers i1a.5 to i1a.9 relate to the third interview question.

Interview 2: Interview with M.

Step 1: Introduce the OIRE method. In this step, we introduced the OIRE method to M, including the purpose, functions, and components of the OIRE method. M could ask us questions about the OIRE method at any time during the interview, and we answered any questions until they said they understood the OIRE method.

Step 2: Show the case study results that were developed with C1 and C2 (in Chinese). In this step, we showed the case study report (in Chinese) to M and explained the content. M could ask us questions about the report at any time during the interview, and we answered any questions until they said they understood the content of the reports.

Step 3: Interview participants collect participants' responses, translate the responses into English, summarize. In this step, we interviewed M by asking several open questions derived from the three main interview questions. We collected and translated the answers and summarize them as follows:

i2a.1) M indicated that small companies are not rich in resources or funds. Therefore, it is difficult for their company to conduct large-scale surveys or face-to-face interviews to obtain user needs. Hence, "copying ideas is easy and happens frequently, so that not many new ideas are proposed."

i2a.2) M indicated that most of the product managers and decision makers in start-ups do not have enough experience or extensive product knowledge to support them in making the right decisions. For example, a product manager might have to handle products from various domains but they might only be deeply familiar with one specific domain.

i2a.3) M said that the idea of the OIRE method was quite new to them. M discovered that the OIRE method used reviews available online as the input data source. This saves time and money otherwise needed for conducting interviews, and also makes it possible to identify new ideas.

i2a.4) M appreciated that the OIRE method solves the problem of product managers (partly) lacking domain knowledge. For example, when users feel very positive about a specific feature in reviews, then this feature should be considered regardless of whether the product manager agrees with or understands the reasons for this assessment.

i2a.5) M believed that quantitative reports are more convincing than qualitative reports and are helpful for persuading users to give up worthless features.

i2a.6) M found the "Sentiment analysis" function of the OIRE method with

its two-dimensional analysis (functional and dysfunctional) to be a creative idea. Users do not only express their feelings about existing features but also about features that do not exist or that exist but do not perform well.

i2a.7) M indicated that in start-ups, due to the lack of resources, it is difficult to conduct large-scale requirements elicitation and analysis in the same way as large enterprises do. M thought that applying the OIRE method required fewer resources than the existing requirements analysis processes. This could benefit start-ups.

Answers i2a.1 and i2a.2 relate to the first main interview question. Answers i2a.3 and i2a.4 relate to the second main interview question. Answers i2a.5 to i2a.7 relate to the third interview question.

7.3. Discussion

In our research question Qu 7.0, we asked whether the OIRE method is useful to decision-makers in industry. In our discussions with C1, C2, P, and M, we received positive feedback.

C1 perceived the analysis results produced by the OIRE method to be valuable. They indicated that the analysis results were "professional", "reliable", and "convincing". They said that the analysis results supported their own ideas about the features they planned to have in a new application. They expected the OIRE method to identify new features. However, in Case 1, we were unable to discover new features due to the limited amount of data, which slightly disappointed C1.

C2 perceived the OIRE method to be useful. They said that the analysis results were "absolutely useful" and "beyond expectations". They indicated that they were willing to consider the analysis results. However, they thought that other factors not included in the analysis results needed to be considered as well. They expected the OIRE method to identify new features or new ideas. In Case 2, although the analysis results showed features that C2 had never thought of before, like C1, C2 was not fully satisfied. They had expected more.

P said that the OIRE method was very interesting and the analysis results were reliable. They indicated that the OIRE method required less manpower and less time than the existing requirements analysis processes used in their company. They believed that the OIRE method could speed up the requirements analysis process and shorten the development cycle. They agreed that the OIRE method can be an effective complement to the existing requirements analysis processes, especially for start-up companies with limited resources. However, for their own company, they preferred the existing requirements analysis processes. Currently, they are using log analysis to monitor the usage of their apps. P believed this to be more objective and reliable, though less explicit, user feedback. They suggested combining the OIRE method with log analysis. They also thought that the idea of the OIRE method could be extended to many other fields. These suggestions gave us new ideas for our future research.

Unlike P, M was from a start-up, and their company lacks funds and human resources (e.g., product managers). M said the OIRE method was a new and interesting idea for them. They believed that applying the OIRE method could benefit their company.

According to the feedback from the stakeholders, we observed that they answered the questions from their context-specific point of view. For example, P worked for a large software company. Their company has sufficient funds and human resources. This made it hard for P to understand the difficulties of small companies that lack resources. Therefore, although P evaluated the OIRE method as positive, they still preferred the existing requirements analysis methods currently in use. Unlike P, M worked for a start-up company with limited resources, which is why they showed more interest in the OIRE method.

In summary, although the OIRE method was perceived to be useful by all stakeholders, we believe that the OIRE method is more useful for decision-makers in small companies.

7.4. Threats to Validity

To analyze threats to validity, we followed the guideline proposed by Engström and Runeson [28]. We identified several threats to validity – to construct validity, internal validity, and external validity.

7.4.1. Construct Validity

When a researcher makes an assumption and uses interviewees' responses to confirm that assumption, the researcher is subconsciously more willing to accept answers that support their assumptions than answers that are unfavorable (researcher bias) [73]. This attitude may influence the wording of the questions, which may lead the interviewees to give a favorable answer. The inadequacy of the communication and the deviation of the understanding would affect the interviewees. The interviewees may feel the researcher's attitude and give the answers that meet the researcher's expectations best. Since we were aware of this potential bias, we deliberately asked control questions during the interviews that helped us re-evaluate the interviewees' answers and challenge wrong pre-conceptualizations.

7.4.2. Internal Validity

When the amount of input data is limited, the amount of training data that can be used in Component 1 of the OIRE method is also limited. To solve this problem, we used data that we had already validated in one of our published papers [109] as a sufficiently large default training data set in the design of the OIRE-S prototype. This means that the machine learning algorithm used in Component 1 of OIRE-S was trained better; however, since the training data and the predicted data were

collected from different sets of data, this may have affected the performance of the machine learning algorithms.

In this evaluation study, we only used one machine learning algorithm, SVM, as the default algorithm of OIRE-S. In our previously published paper [109], SVM appeared to be the best choice. However, there is no guarantee that other machine learning algorithms might not have performed better in the context of this evaluation study.

Although we used real-world data, there is a possibility that the data itself was already biased, making the OIRE method produce misleading output. For example, according to Afnan's research [3], 38% of developers they interviewed add "call-to-action" functionality to their app, asking end users for feedback, and 56% of developers only direct those end users who give a high app rating to the app store. This may bias app ratings and reviews towards favorable feedback. For example, those users who need to use an app often will presumably give more positive ratings.

In the case study, the input text was in Chinese. Since OIRE-S uses English-language analysis tools, the Chinese-language feature-related text needed to be translated into English by an online translator, Baidu Fanyi. The accuracy of the translation may affect that of the analysis results. We reviewed existing research papers on Baidu Fanyi. In Hu's research [99], a study was conducted on the translation quality of Baidu Fanyi based on several examples. Hu concluded that "the translation result is accurate." In Wang's research [37], the performance of several online translation tools was compared: Google Translate[12], Youdao Fanyi[13], Bing Microsoft Translator[14], and Baidu Fanyi. Wang compared the translation accuracy of the above tools at three levels: words, sentences, and articles. By analyzing many translation results, Wang concluded that when translating complex sentences, "Google translation is not accurate enough"; "the details of translation by Bing are not perfect"; "Baidu Fanyi provides a very satisfactory translation result." Based on their research, we think that the accuracy of Baidu Fanyi is relatively high, especially for translating complex sentences. This could reduce the negative impact of the translation process.

Finally, to ensure that the stakeholders would not be worried about us disclosing information or making statements that could shed a negative light on themselves or their company, we agreed to anonymize the identity of the interviewees and their companies.

7.4.3. External Validity

We contacted multiple companies and stakeholders, but for various reasons, most of them refused to cooperate with us. Eventually, we found two customers (C1

[12]https://translate.google.com

[13]https://fanyi.youdao.com

[14]https://www.bing.com/translator

and C2), one product manager (P), and one start-up company manager (M), who were willing to participate in our research. However, we understand that their views might not represent the views of many other companies and stakeholders. This may affect the generalizability of the conclusions of this evaluation study.

We interviewed four stakeholders who had different work experience and backgrounds. We believe that this helped us to collect feedback from different angles, and thus made the evaluation of the OIRE method more comprehensive.

In Case 1 and Case 2, we chose the competitor apps according to the information provided by C1 and C2. These choices might not have been representative (or comprehensive), and thus might have limited the representativeness of the preselected features.

7.5. Conclusion

In Chapter 7, we presented an evaluation of the OIRE method by means of a case study and an interview study.

First, we interviewed two stakeholders, C1 and C2, from two Chinese companies. Since the companies were not software companies, C1 and C2 acted as potential customers of a software app developer who had to communicate their needs to the app developers. We designed one case study with two cases analyzing the development needs of C1 and C2. To support the analysis of the development needs following the OIRE method, we used OIRE-S as tool support. We analyzed real input data collected from an online source and produced analysis results. By discussing the analysis results with C1 and C2, we collected their feedback about the OIRE method. According to their feedback, their evaluation of the OIRE method was positive, although both C1 and C2 had higher expectations regarding the OIRE method. Both C1 and C2 indicated that the analysis results of the OIRE method were useful. Both also mentioned that the OIRE method would be more valuable if it had a better capability of discovering new features.

Next, we interviewed two other stakeholders (P and M) from two Chinese software companies. We collected their views about the OIRE method. Both P and M gave positive evaluations of the OIRE method. According to P and M, the application of the OIRE method could benefit small companies, especially start-ups. P and M agreed that the OIRE method could be an effective complement to existing requirements analysis methods. P said that the OIRE method had the potential to be applied in other fields. P thought that online reviews may contain fake content and therefore preferred the existing requirements analysis processes. M showed more interest than P. M believed the OIRE method to be useful for their company.

In summary, C1, C2, M, and P all agreed that the OIRE method is able to provide useful information to support their decision-making.

7.6. Summary

In this chapter, we evaluated the OIRE method using OIRE-S. Based on the results of the case study and the interview study, we conclude that the OIRE method provides helpful information for stakeholders, and thus is useful to decision-makers in industry, in particular as a complement to existing requirements analysis activities.

8. CONCLUSION AND FUTURE WORK

In this research, we conducted a systematic mapping study to survey the state of the art of Open Innovation (OI) in the sub-fields of software engineering, especially in requirements engineering. We found that quite a few studies had been done during a 15-year period (from the year 2003 to 2016). We also found that there was a lack of published research on the use of OI strategies in specific RE activities, i.e., prioritization and validation, as well as a lack of reported tool support. Therefore, we believe that our research can fill these gaps.

8.1. Summary of Contributions

To (semi-)automatically classify user reviews collected from online open sources, we propose the OIRE method. The OIRE method is the first major contribution of this research. The OIRE method comprises four components. For Component 1, we found a solution by applying machine learning to determine whether a text line potentially corresponds to an answer to a functional or dysfunctional question asked in the Kano model. For Component 2, we designed a dictionary-based method to classify the sentiment of each state such that we could assign an answer value to each text line previously classified as functional or dysfunctional. The dictionary-based method classifies text lines into five sentiment classifications, from Very Negative to Very Positive. We used machine learning methods together with the dictionary-based method to transfer the raw input text data into a format that can be used as the input for the Kano model. Based on the results of our application experiment, we found that the overall accuracy of Component 1 and Component 2 was 65%. We consider this performance to be acceptable. For Component 3, to find a method that produces results resembling those of the traditional Kano model, we proposed two Kano-like models, i.e., the Half-Kano model and the Deformed-Kano model, for dealing with unpaired answers to functional and dysfunctional questions. In order to analyze the performance of the two proposed models compared to that of the traditional Kano model, we ran several simulations with synthetic data. According to the simulation results, we found that the results of using the Deformed-Kano model were always close to the results of the traditional Kano model. Hence, we think that this Kano-like model can be used as an approximation of the traditional Kano model in situations where the input to the Kano model is unpaired or partly missing. Component 4 visualizes the outputs of Components 1 to 3.

The second major contribution of this research is the validation of the OIRE method. We first discussed the selection of the input source. We found that the input collected from app reviews was most suitable for the OIRE method. Next, we presented three typical use cases and a proof-of-concept of the OIRE method demonstrating the applicability of the OIRE method using real-world data collected from the Internet. We found that the overall accuracy of the OIRE method

was between 50% and 61%. For the selected use cases, the OIRE method had the potential of helping software engineers and managers make better-informed RE decisions. We also demonstrated that the OIRE method can produce results for typical use cases of product managers and product owners.

We integrated the dictionary-based method and machine learning techniques with the Kano-like model. Using the R and PHP languages, we designed a web-based prototypical system, OIRE-S. In the final stage of this research, we interviewed industry people and completed a case study and an interview study to evaluate the OIRE method using OIRE-S. This is the third major contribution of this research. According to our evaluation results, we found that the OIRE method can be implemented not only for product managers and software company owners but also for customers with software development needs. The industry people we interviewed considered the analysis results of the OIRE method convincing, and they agreed that the OIRE method could be an effective supplement to traditional requirement analysis methods, especially for start-ups with limited resources. In summary, although the OIRE method was perceived to be useful by all stakeholders, we believe that the OIRE method is most useful for decision-makers in small companies.

8.2. Future Work

Our research opens up a number of directions for the improvement of the OIRE method. In the following, we will outline a few of these possibilities.

In this research, we used several existing machine learning algorithms for sentence classification. We observed that existing classification methods did not perform well in this specific classification process. We believe that better algorithms could improve the accuracy of sentence classification and the effectiveness of the OIRE model as a whole.

In this research, we designed a new dictionary-based algorithm for sentiment analysis. We think that there is room for improving the accuracy of this algorithm.

The Kano-like model is a completely new method that has the potential to be applied in multiple fields. The application of the Kano-like model in other fields (e.g., business, management) could be one direction for our future research.

The OIRE method is part of a unique decision-support system that provides information to stakeholders that otherwise could only be obtained manually (and thus too costly) through methods such as the traditional Kano model. We see potential for integrating the OIRE method with other automated analysis methods, such as methods that automatically analyze usage logs.

BIBLIOGRAPHY

[1] Philip Achimugu, Ali Selamat, Roliana Ibrahim, and Mohd Naz'ri Mahrin. A systematic literature review of software requirements prioritization research. *Information and software technology*, 56(6):568–585, 2014.

[2] Mack W Alford. Software requirements engineering methodology (srem) at the age of two. In *The IEEE Computer Society's Second International Computer Software and Applications Conference, 1978. COMPSAC'78.*, pages 332–339. IEEE, 1978.

[3] Afnan AlSubaihin, Federica Sarro, Sue Black, Licia Capra, and Mark Harman. App store effects on software engineering practices. *IEEE Transactions on Software Engineering*, 2019.

[4] Mohamed Aly. Survey on multiclass classification methods. *Neural networks*, 19:1–9, 2005.

[5] Berander PA Andrews. Requirements prioritization, engineering and managing software requirements, a. aurum and c. wohlin, eds, 2005.

[6] Charles Berger. Kano's methods for understanding customer-defined quality. *Center for quality management journal*, 2(4):3–36, 1993.

[7] Tanmay Bhowmik. Stakeholders' social interaction in requirements engineering of open source software. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 467–472. IEEE, 2014.

[8] Mattia Bianchi, Alberto Cavaliere, Davide Chiaroni, Federico Frattini, and Vittorio Chiesa. Organisational modes for open innovation in the biopharmaceutical industry: An exploratory analysis. *Technovation*, 31(1):22–33, 2011.

[9] Yuri Borgianni and Federico Rotini. Towards the fine-tuning of a predictive kano model for supporting product and service design. *Total Quality Management & Business Excellence*, 26(3-4):263–283, 2015.

[10] Randall D Brandt. A procedure for identifying value-enhancing service components using customer satisfaction survey data. *Add Value to Your Service, Chicago: American Marketing Association*, pages 61–65, 1987.

[11] Ernest R Cadotte and Normand Turgeon. Dissatisfiers and satisfiers: suggestions from consumer complaints and compliments. *Journal of consumer satisfaction, Dissatisfaction and Complaining Behavior*, 1(1):74–79, 1988.

[12] Abhijit Chakraborty, Mrinal Kanti Baowaly, Ashraful Arefin, and Ali Newaz Bahar. The role of requirement engineering in software development life cycle. *Journal of emerging trends in computing and information sciences*, 3(5):723–729, 2012.

[13] Chun-Chih Chen and Ming-Chuen Chuang. Integrating the kano model into a robust design approach to enhance customer satisfaction with product

design. *International journal of production economics*, 114(2):667–681, 2008.

[14] Betty HC Cheng and Joanne M Atlee. Research directions in requirements engineering. In *2007 Future of Software Engineering*, pages 285–303. IEEE Computer Society, 2007.

[15] Henry Chesbrough and Adrienne Kardon Crowther. Beyond high tech: early adopters of open innovation in other industries. *R&d Management*, 36(3):229–236, 2006.

[16] Henry Chesbrough, Wim Vanhaverbeke, and Joel West. *Open innovation: Researching a new paradigm*. Oxford University Press on Demand, 2006.

[17] Henry William Chesbrough. *Open Innovation: The New Imperative for Creating and Profiting from Technology*. Harvard Business Press, 2003.

[18] John W Creswell. Research design: Qualitative, quantitative, and mixed methods approaches. *Canadian Journal of University Continuing Education*, 35(2), 2009.

[19] Sanjiv Das and Mike Chen. Yahoo! for amazon: Extracting market sentiment from stock message boards. In *Proceedings of the Asia Pacific finance association annual conference (APFA)*, volume 35, page 43. Bangkok, Thailand, 2001.

[20] Sajib Dasgupta and Vincent Ng. Mine the easy, classify the hard: a semi-supervised approach to automatic sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 701–709. Association for Computational Linguistics, 2009.

[21] Koen De Backer, Vladimir López-Bassols, and Catalina Martinez. Open innovation in a global perspective: what do existing data tell us? *OECD Science, Technology and Industry Working Papers*, 2008(4):0_1, 2008.

[22] Dian Retno Sari Dewi, Joana Debora, and Martinus Edy Sianto. Dealing with dissatisfaction in mathematical modelling to integrate qfd and kano's model. In *IOP Conference Series: Materials Science and Engineering*, volume 277, page 012009. IOP Publishing, 2017.

[23] Zhendong Dong. Hownet knowledge database. `http://www.keenage.com/html/e_index.html`. Accessed February 2, 2017.

[24] Henry Edison, Nauman Bin Ali, and Richard Torkar. Towards innovation measurement in the software industry. *Journal of Systems and Software*, 86(5):1390–1407, 2013.

[25] Fatma İpek Ek and Şeniz Çıkış. Integrating the kano model into architectural design: quality measurement in mass-housing units. *Total Quality Management & Business Excellence*, 26(3-4):400–414, 2015.

[26] Sascha El-Sharkawy and Klaus Schmid. A heuristic approach for supporting product innovation in requirements engineering: a controlled experiment. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 78–93. Springer, 2011.

[27] Charles R Emery and Robert G Tian. Schoolwork as products, professors as customers: a practical teaching approach in business education. *Journal of Education for Business*, 78(2):97–102, 2002.

[28] Emelie Engström, Per Runeson, and Andreas Ljung. Improving regression testing transparency and efficiency with history-based prioritization–an industrial case study. In *2011 Fourth IEEE International Conference on Software Testing, Verification and Validation*, pages 367–376. IEEE, 2011.

[29] Aleksander Fabijan, Helena Holmström Olsson, and Jan Bosch. Commodity eats innovation for breakfast: a model for differentiating feature realization. In *International Conference on Product-Focused Software Process Improvement*, pages 517–525. Springer, 2016.

[30] Donald Firesmith. Prioritizing requirements. *Journal of Object Technology*, 3(8):35–48, 2004.

[31] Samuel A Fricker, Ernest Wallmüller, and Ina Paschen. Requirements engineering as innovation journalism: a research preview. In *2016 IEEE 24th International Requirements Engineering Conference (RE)*, pages 335–340. IEEE, 2016.

[32] Margareta Friman and Bo Edvardsson. A content analysis of complaints and compliments. *Managing Service Quality: An International Journal*, 13(1):20–26, 2003.

[33] Cecilia Garibay, Humberto Gutiérrez, and Arturo Figueroa. Evaluation of a digital library by means of quality function deployment (qfd) and the kano model. *The Journal of Academic Librarianship*, 36(2):125–132, 2010.

[34] Deepa Gopal, Aron Lindberg, and Kalle Lyytinen. Attributes of open source software requirements–the effect of the external environment and internal social structure. In *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pages 4982–4991. IEEE, 2016.

[35] Eduard C Groen, Joerg Doerr, and Sebastian Adam. Towards crowd-based requirements engineering a research preview. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 247–253. Springer, 2015.

[36] Eduard C Groen, Norbert Seyff, Raian Ali, Fabiano Dalpiaz, Joerg Doerr, Emitza Guzman, Mahmood Hosseini, Jordi Marco, Marc Oriol, Anna Perini, et al. The crowd in requirements engineering: The landscape and challenges. *IEEE software*, 34(2):44–52, 2017.

[37] Wang Hai. A comparative study on the most frequently used online translation dictionaries. *Social Science Theory*, (3):140–143, 2017.

[38] Elad Harison and Heli Koski. Applying open innovation in business strategies: Evidence from finnish software firms. *Research Policy*, 39(3):351–359, 2010.

[39] Frederic Herzberg, Bernard Mausner, and Barbara B Snyderman. *The motivation to work*. Wiley, 1959.

[40] Mokter Hossain and Ilkka Kauranen. Open innovation in smes: a systematic literature review. *Journal of Strategy and Management*, 9(1):58–73, 2016.

[41] Mahmood Hosseini, Keith Phalp, Jacqui Taylor, and Raian Ali. The four pillars of crowdsourcing: A reference model. In *2014 IEEE Eighth International Conference on Research Challenges in Information Science (RCIS)*, pages 1–12. IEEE, 2014.

[42] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM, 2004.

[43] Jianxin Jiao and Chun-Hsien Chen. Customer requirement management in product development: a review of research issues. *Concurrent Engineering*, 14(3):173–185, 2006.

[44] Ding Junwu, Yang Dongtao, and Bao Zhenqiang. Research on capturing of customer requirements based on innovation theory. *Physics Procedia*, 24:1868–1880, 2012.

[45] Andreas Stefan Kain, Rafael Johannes Kirschner, Alexander Lang, Udo Lindemann, et al. Facing the open innovation dilemma–structuring input at the company's border. In *DS 68-1: Proceedings of the 18th International Conference on Engineering Design (ICED 11), Impacting Society through Engineering Design, Vol. 1: Design Processes, Lyngby/Copenhagen, Denmark, 15.-19.08. 2011*, pages 487–498, 2011.

[46] N. KANO. Attractive quality and must-be quality. *Hinshitsu (Quality, the Journal of Japanese Society for Quality Control)*, 14:39–48, 1984.

[47] Joachim Karlsson and Kevin Ryan. A cost-value approach for prioritizing requirements. *IEEE software*, 14(5):67–74, 1997.

[48] Joachim Karlsson, Claes Wohlin, and Björn Regnell. An evaluation of methods for prioritizing software requirements. *Information and software technology*, 39(14-15):939–947, 1998.

[49] Barbara Kitchenham, Pearl Brereton, and David Budgen. Mapping study completeness and reliability-a case study. 2012.

[50] Eric Knauss, Daniela Damian, Alessia Knauss, and Arber Borici. Openness and requirements: opportunities and tradeoffs in software ecosystems. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 213–222. IEEE, 2014.

[51] Lun-Wei Ku, Tung-Ho Wu, Li-Ying Lee, and Hsin-Hsi Chen. Construction of an evaluation corpus for opinion extraction. In *Proceedings of the 5th NTCIR Workshop Meeting, December 6-9*, pages 513–520, 2005.

[52] Jaison Kuriakose and Jeffrey Parsons. How do open source software (oss) developers practice and perceive requirements engineering? an empirical study. In *2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE)*, pages 49–56. IEEE, 2015.

[53] Katja Landgraf and Roland Jochem. Innovation management needs an interoperable requirements management. In *International IFIP Working Conference on Enterprise Interoperability*, pages 5–19. Springer, 2012.

[54] Dean Leffingwell and Don Widrig. *Managing software requirements: a unified approach*. Addison-Wesley Professional, 2000.

[55] Laura Lehtola, Marjo Kauppinen, and Sari Kujala. Requirements prioritization challenges in practice. In *International Conference on Product Focused Software Process Improvement*, pages 497–508. Springer, 2004.

[56] Shugang Li and Yueming Li. A sentiment analysis of online reviews based on the word alignment model: A product improvement perspective. In *2018 2nd IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pages 2226–2231. IEEE, 2018.

[57] Zhai Li-li, Hong Lian-feng, and Sun Qin-ying. Research on requirement for high-quality model of extreme programming. In *2011 International Conference on Information Management, Innovation Management and Industrial Engineering*, volume 1, pages 518–522. IEEE, 2011.

[58] Johan Linåker, Björn Regnell, and Hussan Munir. Requirements engineering in open innovation: a research agenda. In *Proceedings of the 2015 International Conference on Software and System Process*, pages 208–212. ACM, 2015.

[59] Johan Linåker and Krzysztof Wnuk. Requirements analysis and management for benefiting openness. In *2016 IEEE 24th International Requirements Engineering Conference Workshops (REW)*, pages 344–349. IEEE, 2016.

[60] Sonia M Lo, Han-Ping Shen, and James C Chen. An integrated approach to project management using the kano model and qfd: an empirical case study. *Total Quality Management & Business Excellence*, 28(13-14):1584–1608, 2017.

[61] Dario Lorenzi and Cristina Rossi. Assessing innovation in the software sector: proprietary vs. foss production mode. preliminary evidence from the italian case. In *IFIP International Conference on Open Source Systems*, pages 325–331. Springer, 2008.

[62] Neil Maiden. User requirements and system requirements. *IEEE Software*, 25(2):90–91, 2008.

[63] Elsa Marcelino-Jesus, Joao Sarraipa, Carlos Agostinho, and Ricardo Jardim-Goncalves. A requirements engineering methodology for technological innovations assessment. In *ISPE CE*, pages 577–586, 2014.

[64] Kurt Matzler and Hans H Hinterhuber. How to make product development projects more successful by integrating kano's model of customer satisfaction into quality function deployment. *Technovation*, 18(1):25–38, 1998.

[65] Josip Mikulić and Darko Prebežac. A critical review of techniques for classifying quality attributes in the kano model. *Managing Service Quality: An International Journal*, 21(1):46–66, 2011.

[66] Francesca Montagna. How should requirements be defined to have real innovation? *Procedia CIRP*, 21:527–532, 2014.

[67] Satoshi Morinaga, Kenji Yamanishi, Kenji Tateishi, and Toshikazu Fukushima. Mining product reputations on the web. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 341–349. ACM, 2002.

[68] Tobias Müller-Prothmann. Give lead users the lead. integration of requirements engineering into innovation processes. *Journal of Bone & Mineral Research the Official Journal of the American Society for Bone & Mineral Research*, 27(3):2773–2783, 2012.

[69] Hussan Munir, Krzysztof Wnuk, and Per Runeson. Open innovation in software engineering: a systematic mapping study. *Empirical Software Engineering*, 21(2):684–723, 2016.

[70] Paula Nascimento, Rodrigo Aguas, Daniel Schneider, and Jano De Souza. An approach to requirements categorization using kano's model and crowds. In *Proceedings of the 2012 IEEE 16th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, pages 387–392. IEEE, 2012.

[71] Tetsuya Nasukawa and Jeonghee Yi. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*, pages 70–77. ACM, 2003.

[72] Kateryna Neulinger, Anna Hannemann, Ralf Klamma, and Matthias Jarke. A longitudinal study of community-oriented open source software development. In *International Conference on Advanced Information Systems Engineering*, pages 509–523. Springer, 2016.

[73] Raymond S Nickerson. Confirmation bias: A ubiquitous phenomenon in many guises. *Review of general psychology*, 2(2):175–220, 1998.

[74] Bashar Nuseibeh and Steve Easterbrook. Requirements engineering: a roadmap. In *Proceedings of the Conference on the Future of Software Engineering*, pages 35–46. ACM, 2000.

[75] Richard L Oliver. *Satisfaction: A Behavioral Perspective on the Consumer*. ME Sharpe, 2010.

[76] Barbara Paech and Bernd Reuschenbach. Open source requirements engineering. In *14th IEEE International Requirements Engineering Conference (RE'06)*, pages 257–262. IEEE, 2006.

[77] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics, 2002.

[78] Anna Perini, Angelo Susi, and Paolo Avesani. A machine learning approach to software requirements prioritization. *IEEE Transactions on Software Engineering*, 39(4):445–461, 2012.

[79] Kai Petersen and Nauman Bin Ali. Identifying strategies for study selection in systematic reviews and maps. In *2011 International Symposium on Empirical Software Engineering and Measurement*, pages 351–354. IEEE, 2011.

[80] Kai Petersen, Robert Feldt, Shahid Mujtaba, and Michael Mattsson. Systematic mapping studies in software engineering. In *Ease*, volume 8, pages 68–77, 2008.

[81] Andrew J Reagan, Brian Tivnan, Jake Ryland Williams, Christopher M Danforth, and Peter Sheridan Dodds. Benchmarking sentiment analysis methods for large-scale texts: a case for using continuum-scored words and word shift graphs. *arXiv preprint arXiv:1512.00531*, 2015.

[82] Björn Regnell, Martin Höst, Fredrik Nilsson, and Henrik Bengtsson. A measurement framework for team level assessment of innovation capability in early requirements engineering. In *International Conference on Product-Focused Software Process Improvement*, pages 71–86. Springer, 2009.

[83] Clotilde Rohleder, Jing Lin, Indra Kusuma, and Gülru Özkan. Business analytics in innovation and product lifecycle management: poster paper. In *IEEE 7th International Conference on Research Challenges in Information Science (RCIS)*, pages 1–2. IEEE, 2013.

[84] Per Runeson and Martin Höst. Guidelines for conducting and reporting case study research in software engineering. *Empirical software engineering*, 14(2):131, 2009.

[85] Thomas L Saaty. What is the analytic hierarchy process? In *Mathematical models for decision support*, pages 109–121. Springer, 1988.

[86] Walt Scacchi. Understanding requirements for open source software. In *Design requirements engineering: A ten-year perspective*, pages 467–494. Springer, 2009.

[87] Alexander Schroll and Andreas Mild. A critical review of empirical research on open innovation adoption. *Journal für Betriebswirtschaft*, 62(2):85–118, 2012.

[88] AMM Sharif Ullah and Jun'ichi Tamaki. Analysis of kano-model-based customer needs for product development. *Systems Engineering*, 14(2):154–172, 2011.

[89] Chris J Skinner. Probability proportional to size (pps) sampling. *Wiley StatsRef: Statistics Reference Online*, pages 1–5, 2014.

[90] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642, 2013.

[91] Ian Sommerville. Software engineering 9th edition. *ISBN-10137035152*, 2011.

[92] Huaming Song, Chao Chen, and Qin Yu. Research on kano model based on online comment data mining. In *2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA)*, pages 76–82. IEEE, 2018.

[93] Stephen V Stehman. Selecting and interpreting measures of thematic classification accuracy. *Remote sensing of Environment*, 62(1):77–89, 1997.

[94] Richard M Tong. An operational system for detecting and tracking opinions in on-line discussion. In *Working Notes of the ACM SIGIR 2001 Workshop on Operational Text Classification*, volume 1, 2001.

[95] Gerson Tontini. Integrating the kano model and qfd for designing new products. *Total Quality Management*, 18(6):599–612, 2007.

[96] Peter D Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 417–424. Association for Computational Linguistics, 2002.

[97] Terry G Vavra. *Improving your measurement of customer satisfaction: A guide to creating, conducting, analyzing, and reporting customer satisfaction measurement programs*. ASQ quality press, 1997.

[98] Chong Wang, Maya Daneva, Marten van Sinderen, and Peng Liang. A systematic mapping study on crowdsourced requirements engineering using user feedback. *Journal of Software: Evolution and Process*, page e2199, 2019.

[99] Hu wei. Baidu translation english-chinese translation quality improvement strategy. *China Science and Technology Information*, (z4):133–134, 2015.

[100] Janyce Wiebe et al. Learning subjective adjectives from corpora. *Aaai/iaai*, 20(0):0, 2000.

[101] Roel Wieringa, Neil Maiden, Nancy Mead, and Colette Rolland. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. *Requirements engineering*, 11(1):102–107, 2006.

[102] Theresa Wilson, Janyce Wiebe, and Rebecca Hwa. Just how mad are you? finding strong and weak opinion clauses. In *aaai*, volume 4, pages 761–769, 2004.

[103] Krzysztof Wnuk, Dietmar Pfahl, David Callele, and Even-André Karlsson. How can open source software development help requirements management gain the potential of open innovation: an exploratory study. In *Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*, pages 271–280. ACM, 2012.

[104] Krzysztof Wnuk and Per Runeson. Engineering open innovation–towards a framework for fostering open innovation. In *International Conference of Software Business*, pages 48–59. Springer, 2013.

[105] K. Yang. *Voice of the Customer: Capture and Analysis*. Six sigma operational methods series. McGraw-Hill Education, 2007.

[106] Huishi Yin. A study plan: open innovation based on internet data mining in software engineering. In *Proceedings of the 2015 International Conference on Software and System Process*, pages 192–193. ACM, 2015.

[107] Huishi Yin and Dietmar Pfahl. A preliminary study into research about open innovation with focus on the field of computer science. In *Proceedings of the 2015 International Conference on Software and System Process*, pages 204–207. ACM, 2015.

[108] Huishi Yin and Dietmar Pfahl. Evaluation of kano-like models defined for using data extracted from online sources. In *International Conference on Product-Focused Software Process Improvement*, pages 539–549. Springer, 2016.

[109] Huishi Yin and Dietmar Pfahl. A method to transform automatically extracted product features into inputs for kano-like models. In *International Conference on Product-Focused Software Process Improvement*, pages 237–254. Springer, 2017.

[110] Huishi Yin and Dietmar Pfahl. Open innovation in software requirements engineering: A mapping study. In *2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS)*, pages 5–10, Nov 2017.

[111] Huishi Yin and Dietmar Pfahl. A preliminary study on the suitability of stack overflow for open innovation in requirements engineering. In *Proceedings of the 3rd International Conference on Communication and Information Processing*, pages 45–49. ACM, 2017.

[112] Huishi Yin and Dietmar Pfahl. The oire method - overview and initial validation(accept*). In *25th APSEC 2018*. ACM, 2018.

[113] Hong Yu and Vasileios Hatzivassiloglou. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 129–136. Association for Computational Linguistics, 2003.

[114] He Yubing. The review and prospect of open innovation re-search. *Science of Science and Management of S. & T*, 36(3):3–12, 2015.

[115] Daniel Zacarias. The complete guide to the kano model-prioritizing customer satisfaction and delight. `https://foldingburritos.com/kano-model/`. Accessed June 15, 2016.

[116] Fang Lan Zhang, Chen Xi Jia, and Yanshan University. Products innovation method based on classification and importance evaluation of user needs. *Packaging Engineering*, 2017.

Appendix A. COLLECTION OF THE LINKS TO RESEARCH MATERIALS

A.1. Code

Source code of OIRE-S:
https://figshare.com/s/9bc19c086449be76ed90

A.2. Document

Feature-related texts used in Chapter 6:
https://figshare.com/s/64dbc8e3538e704efb22

ACKNOWLEDGEMENT

I spent some precious years at the beautiful University of Tartu. I completed my PhD studies, increased my knowledge and learning ability, and gained a wealth of valuable life experience. The help and support from my supervisor, my family, and close friends made it possible for me to successfully complete my studies.

First of all, I sincerely thank my supervisor, Professor Dietmar Pfahl. It was my good fortune to have such a good supervisor. Without his guidance and careful and timely feedback, I would not have had a chance to complete and publish my studies. I have benefited from his immense knowledge, rigorous academic attitude, and unremitting dedication to doing research.

Besides my supervisor, I would like to express my appreciation to Professor Marlon Dumas for his insightful comments and encouragement, but also for all his help which motivated me to widen my research in many ways.

Furthermore, I am extremely grateful to my family for supporting me spiritually throughout this research and my life in general.

In the past years, I suffered from depression. When I was extremely pessimistic, my supervisor encouraged me to move forward. My parents and my dear friends, especially Dr. Chen, took care of me when I was not feeling well. I am deeply thankful to all those who gave me help and encouragement. Their help was akin to a lifeguard rescuing a drowning person from the deep sea.

SUMMARY IN ESTONIAN

Kano-sarnase mudeli kasutamine avatud innovatsiooni saavutamiseks nõuete analüüsi protsessis

Kui viiakse läbi nõuete analüüsi (inglise k Requirements Engineering, lühend RE), siis sageli järjestatakse nõuded nende olulisuse alusel (inglise k requirements prioritization), et saada selgust, milliste välja pakutud nõuetega funktsioon peaks tarkvaral olemas olema, seega sõltub tarkvara analüüsist tarkvara majandusliku väärtuse suurendamisega seotud otsuste tegemine. Tänapäeval arenevad tooted väga kiiresti ning ka nõuete olulisuse alusel järjestamine (inglise k requirements prioritization) on muutunud kiiremaks. Ettevõtted sooviksid saada kasutajatelt kiiret tagasisidet selle kohta, mis peaks olema järgmises mudelis olemas. Üks häid lahendusi sellele on Kano mudel (inglise k Kano model). Kano mudel selgitab välja kasutajate rahulolu ja toodete tunnuste vahelise suhte. See meetod liigitab kasutajate eelistused nende tähtsuse järjekorras, seega toetab see ka nõuete olulisuse järjekorra moodustamist. Aga Kano mudeli rakendamine on kallis ja aeganõudev ning seda ei saa kiiresti korrata. Veelgi enam – see mudel on keeruline väikeste ettevõtete jaoks, sest neil ei tarvitse olla piisavalt rahalisi jm vahendeid, et kasutajatega ühendust võtta ja neid intervjueerida. See omakorda paneb väikesed ettevõtted, eriti just idufirmad, ebavõrdsesse olukorda suurte ettevõtetega.

Et sellele probleemile lahendust leida ja Kano mudeli kasutuselevõttu lihtsamaks ning odavamaks teha, arvame, et Kano mudelit tuleks arendada kahel viisil. Esiteks tuleks kasutada tasuta võrgus leiduvaid kirjalikke andmeid, mida saaks asendada intervjueeritavatelt kogutud vastustega. Teiseks – selleks, et hakkama saada võrgust kogutud kirjalike andmete suure mahuga, ning et kaasa aidata korrapärastele analüüsidele, peaks andmete analüüsimine olema automaatne.

Selle uurimuse eesmärk on välja pakkuda meetodeid, et kasutajate avamusi, mis on võrgus saadavatest vabadest allikatest kogutud, (semi-)automaatselt liigitada, ja seda selleks, et aidata otsustajatel otsustada, millised tarkvara nõuded järgmises mudelis kindlasti olemas peaksid olema. Et seda uurimuse eesmärki saavutada, pakume me välja *avatud innovatsiooni nõuete analüüsi* (inglise k Open Innovation in Requirements Engineering, lühend OIRE) meetodi, mille abil saavad tarkvarafirmad parema ülevaate kasutajate vajadustest ja sellest, kuivõrd rahul on nad olemasolevate toodetega. Põhiline OIRE meetodi puhul on Kano mudelile sarnane mudel (inglise k Kano-like model). See mudel matkib traditsioonilist Kano mudelit erinedes ainult selle poolest, et ta kasutab võrgust leitud ülevaateid selle asemel, et kasutada fookus- gruppidega tehtud intervjuusid. Kasutame masinõpet ja tundmusanalüüsi meetodit, et analüüsida tekstiridu, mis vastavad Kano mudelile sarnasele mudeli sisendile.

OIRE meetodi eesmärk on aidata tarkvaraarendajatel mõista, kui palju lõppkasutajad võiksid hinnata hetkel-arenduses-olevat tarkvara toodet. Samuti alandab pooleldi automatiseeritud lähenemine RE kulusid. Me arvame, et OIRE meetod

on kasulikum just väikestele firmadele, millel on väiksed meeskonnad ning väike müügi ja ostjate uurimisega seotud eelarve.

Panustame selle uurimusega ka sellega, et esitleme kolme tüüpilist kasutusviisi ja OIRE meetodi idee tõendamine näitlikustab OIRE meetodi rakendatavust Internetist kogutud päris-maailma andmetele. Võtame vaatluse alla ka võimalike sisendandmete sobivuse OIRE meetodiga.

Kolmas selle uurimuse panus seisneb prototüüpse veebipõhise süsteemi, OIRE Süsteemi (inglise k OIRE System, lühend OIRE-S) disainimises ja rakendamises, ning OIRE meetodi hindamises kasutades OIRE Süsteemi. Me oleme juhtinud üht uurimustööd koos kahe Hiina ettevõttega ja üht intervjuu-uurimust koos kahe teise asjaosalisega. Uurimustöö ja intervjuude-uurimuse tulemused näitasid, et OIRE meetodit peeti kõikide asjaosaliste poolt kasulikuks, ning et seda peeti kasulikuks just väikeste ettevõtete otsustajatele.

CURRICULUM VITAE

Personal data

Name:	Huishi Yin
Date of birth:	07.10.1981
Citizenship:	Chinese

Education

2015–2019	University of Tartu, Faculty of Science and Technology, doctoral studies, specialty: Computer Science.
2012–2014	University of Tartu, Faculty of Mathematics and Computer Science, master studies, specialty: Software Engineering.
2001–2005	Hebei University of Technology, China, bachelor studies, specialty: Information System and Information Management

Employment

2005–2010	Network System Engineer, Network Research Center of Tsinghua University (China Educational Reach Network Center)

Scientific work

Main fields of interest:

- Requirements engineering
- Sentiment analysis
- Data mining

ELULOOKIRJELDUS

Isikuandmed

Nimi:	Huishi Yin
Sünniaeg:	07.10.1981
Kodakondsus:	Hiina

Haridus

2015–2019	Tartu Ülikool, loodus- ja täppisteaduste valdkond, doktoriõpe, eriala: informaatika.
2012–2014	Tartu Ülikool, matemaatika-informaatikateaduskond, magistriõpe, eriala: tarkvaratehnika.
2001–2005	Hebei Tehnikaülikool, Hiina, bakalaureuseõpe, eriala: Information System and Information Management

Teenistuskäik

2005–2010	Network System Engineer, Network Research Center of Tsinghua University (China Educational Reach Network Center)

Teadustegevus

Peamised uurimisvaldkonnad:

- Nõuete analüüs
- Sentiment-analüüs
- Andmekaeve

LIST OF ORIGINAL PUBLICATIONS

1. Yin H. A Study Plan: Open Innovation Based on Internet Data Mining in Software Engineering. Proceedings of the 2015 International Conference on Software and System Process. ACM, 2015: 192-193.
2. Yin H, Pfahl D. A Preliminary Study into Research about Open Innovation with Focus on the Field of Computer Science. Proceedings of the 2015 International Conference on Software and System Process. ACM, 2015: 204-207.
3. Yin H, Pfahl D. Evaluation of Kano-like Models Defined for Using Data Extracted from Online Sources. Product-Focused Software Process Improvement: 17th International Conference, PROFES 2016, Trondheim, Norway, November 22-24, 2016, Proceedings. Springer International Publishing, 2016: 539-549.
4. Yin H, Pfahl D. A Method to Transform Automatically Extracted Product Features into Inputs for Kano-Like Models. International Conference on Product-Focused Software Process Improvement. Springer, Cham, 2017: 237-254.
5. Yin H, Pfahl D. A Preliminary Study on the Suitability of Stack Overflow for Open Innovation in Requirements Engineering. Proceedings of the 3rd International Conference on Communication and Information Processing. ACM, 2017: 45-49.
6. Yin H, Pfahl D. Open Innovation in Software Requirements Engineering: A Mapping study. 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS). IEEE, 2017: 5-10.
7. Yin H, Pfahl D. The OIRE Method-Overview and Initial Validation. 2018 25th Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2018: 1-10.

DISSERTATIONES INFORMATICAE PREVIOUSLY PUBLISHED IN DISSERTATIONES MATHEMATICAE UNIVERSITATIS TARTUENSIS

19. **Helger Lipmaa.** Secure and efficient time-stamping systems. Tartu, 1999, 56 p.
22. **Kaili Müürisep.** Eesti keele arvutigrammatika: süntaks. Tartu, 2000, 107 lk.
23. **Varmo Vene.** Categorical programming with inductive and coinductive types. Tartu, 2000, 116 p.
24. **Olga Sokratova.** Ω-rings, their flat and projective acts with some applications. Tartu, 2000, 120 p.
27. **Tiina Puolakainen.** Eesti keele arvutigrammatika: morfoloogiline ühestamine. Tartu, 2001, 138 lk.
29. **Jan Villemson.** Size-efficient interval time stamps. Tartu, 2002, 82 p.
45. **Kristo Heero.** Path planning and learning strategies for mobile robots in dynamic partially unknown environments. Tartu 2006, 123 p.
49. **Härmel Nestra.** Iteratively defined transfinite trace semantics and program slicing with respect to them. Tartu 2006, 116 p.
53. **Marina Issakova.** Solving of linear equations, linear inequalities and systems of linear equations in interactive learning environment. Tartu 2007, 170 p.
55. **Kaarel Kaljurand.** Attempto controlled English as a Semantic Web language. Tartu 2007, 162 p.
56. **Mart Anton.** Mechanical modeling of IPMC actuators at large deformations. Tartu 2008, 123 p.
59. **Reimo Palm.** Numerical Comparison of Regularization Algorithms for Solving Ill-Posed Problems. Tartu 2010, 105 p.
61. **Jüri Reimand.** Functional analysis of gene lists, networks and regulatory systems. Tartu 2010, 153 p.
62. **Ahti Peder.** Superpositional Graphs and Finding the Description of Structure by Counting Method. Tartu 2010, 87 p.
64. **Vesal Vojdani.** Static Data Race Analysis of Heap-Manipulating C Programs. Tartu 2010, 137 p.
66. **Mark Fišel.** Optimizing Statistical Machine Translation via Input Modification. Tartu 2011, 104 p.
67. **Margus Niitsoo**. Black-box Oracle Separation Techniques with Applications in Time-stamping. Tartu 2011, 174 p.
71. **Siim Karus.** Maintainability of XML Transformations. Tartu 2011, 142 p.
72. **Margus Treumuth.** A Framework for Asynchronous Dialogue Systems: Concepts, Issues and Design Aspects. Tartu 2011, 95 p.
73. **Dmitri Lepp.** Solving simplification problems in the domain of exponents, monomials and polynomials in interactive learning environment T-algebra. Tartu 2011, 202 p.

74. **Meelis Kull.** Statistical enrichment analysis in algorithms for studying gene regulation. Tartu 2011, 151 p.
77. **Bingsheng Zhang.** Efficient cryptographic protocols for secure and private remote databases. Tartu 2011, 206 p.
78. **Reina Uba.** Merging business process models. Tartu 2011, 166 p.
79. **Uuno Puus.** Structural performance as a success factor in software development projects – Estonian experience. Tartu 2012, 106 p.
81. **Georg Singer.** Web search engines and complex information needs. Tartu 2012, 218 p.
83. **Dan Bogdanov.** Sharemind: programmable secure computations with practical applications. Tartu 2013, 191 p.
84. **Jevgeni Kabanov.** Towards a more productive Java EE ecosystem. Tartu 2013, 151 p.
87. **Margus Freudenthal.** Simpl: A toolkit for Domain-Specific Language development in enterprise information systems. Tartu, 2013, 151 p.
90. **Raivo Kolde.** Methods for re-using public gene expression data. Tartu, 2014, 121 p.
91. **Vladimir Šor.** Statistical Approach for Memory Leak Detection in Java Applications. Tartu, 2014, 155 p.
92. **Naved Ahmed.** Deriving Security Requirements from Business Process Models. Tartu, 2014, 171 p.
94. **Liina Kamm.** Privacy-preserving statistical analysis using secure multi-party computation. Tartu, 2015, 201 p.
100. **Abel Armas Cervantes.** Diagnosing Behavioral Differences between Business Process Models. Tartu, 2015, 193 p.
101. **Fredrik Milani.** On Sub-Processes, Process Variation and their Interplay: An Integrated Divide-and-Conquer Method for Modeling Business Processes with Variation. Tartu, 2015, 164 p.
102. **Huber Raul Flores Macario.** Service-Oriented and Evidence-aware Mobile Cloud Computing. Tartu, 2015, 163 p.
103. **Tauno Metsalu.** Statistical analysis of multivariate data in bioinformatics. Tartu, 2016, 197 p.
104. **Riivo Talviste.** Applying Secure Multi-party Computation in Practice. Tartu, 2016, 144 p.
108. **Siim Orasmaa.** Explorations of the Problem of Broad-coverage and General Domain Event Analysis: The Estonian Experience. Tartu, 2016, 186 p.
109. **Prastudy Mungkas Fauzi.** Efficient Non-interactive Zero-knowledge Protocols in the CRS Model. Tartu, 2017, 193 p.
110. **Pelle Jakovits.** Adapting Scientific Computing Algorithms to Distributed Computing Frameworks. Tartu, 2017, 168 p.
111. **Anna Leontjeva.** Using Generative Models to Combine Static and Sequential Features for Classification. Tartu, 2017, 167 p.
112. **Mozhgan Pourmoradnasseri.** Some Problems Related to Extensions of Polytopes. Tartu, 2017, 168 p.

113. **Jaak Randmets.** Programming Languages for Secure Multi-party Computation Application Development. Tartu, 2017, 172 p.
114. **Alisa Pankova.** Efficient Multiparty Computation Secure against Covert and Active Adversaries. Tartu, 2017, 316 p.
116. **Toomas Saarsen.** On the Structure and Use of Process Models and Their Interplay. Tartu, 2017, 123 p.
121. **Kristjan Korjus.** Analyzing EEG Data and Improving Data Partitioning for Machine Learning Algorithms. Tartu, 2017, 106 p.
122. **Eno Tõnisson.** Differences between Expected Answers and the Answers Offered by Computer Algebra Systems to School Mathematics Equations. Tartu, 2017, 195 p.

DISSERTATIONES INFORMATICAE UNIVERSITATIS TARTUENSIS

1. **Abdullah Makkeh**. Applications of Optimization in Some Complex Systems. Tartu 2018, 179 p.
2. **Riivo Kikas**. Analysis of Issue and Dependency Management in Open-Source Software Projects. Tartu 2018, 115 p.
3. **Ehsan Ebrahimi**. Post-Quantum Security in the Presence of Superposition Queries. Tartu 2018, 200 p.
4. **Ilya Verenich**. Explainable Predictive Monitoring of Temporal Measures of Business Processes. Tartu 2019, 151 p.
5. **Yauhen Yakimenka**. Failure Structures of Message-Passing Algorithms in Erasure Decoding and Compressed Sensing. Tartu 2019, 134 p.
6. **Irene Teinemaa**. Predictive and Prescriptive Monitoring of Business Process Outcomes. Tartu 2019, 196 p.
7. **Mohan Liyanage.** A Framework for Mobile Web of Things. Tartu 2019, 131 p.
8. **Toomas Krips.** Improving performance of secure real-number operations. Tartu 2019, 146 p.
9. **Vijayachitra Modhukur.** Profiling of DNA methylation patterns as biomarkers of human disease. Tartu 2019, 134 p.
10. **Elena Sügis.** Integration Methods for Heterogeneous Biological Data. Tartu 2019, 250 p.
11. **Tõnis Tasa.** Bioinformatics Approaches in Personalised Pharmacotherapy. Tartu 2019, 150 p.
12. **Sulev Reisberg.** Developing Computational Solutions for Personalized Medicine. Tartu 2019, 126 p.