Abstract

As part of iterative development, decision about “Is the software product ready to be released at some given release date?” have to be made at the end of each release, sprint or iteration. While this decision is critically important, so far it is largely done either informally or in a simplistic manner relying on a small set of isolated metrics. In addition, continuity in release readiness evaluation is not achieved and any problems related to release issues cannot be addressed proactively. In this thesis, an analytical approach called RELREA is proposed to address these gaps. The main characteristics of RELREA are: i) project specific release readiness attributes and metrics are selected in a systematic way, ii) overall release readiness is determined by aggregating and evaluating the degree of satisfaction of the most relevant readiness attributes, iii) continuous visibility on release readiness status, and iv) projection of release readiness at release time. Case studies with proprietary and open source software projects are performed to evaluate the proposed approach and its implementation. The case study results show that the RELREA approach and its implementation are able to assist one product manager in evaluating, monitoring, and analyzing release readiness at any point of time in the release cycle.
Publications

Some of the ideas, materials, tables, figures used in this thesis have appeared previously in the following referred publications:

**Referred published and accepted papers:**


**Papers under review:**


Acknowledgements

This thesis would not have taken place without the continuous guidance and the help of several individuals who in one way or other contributed and extended their valuable assistance in the preparation and completion of this thesis. I am mostly grateful to my supervisor Dr. Guenther Ruhe for his valuable guidance, continuous support, and supervision. Without his inspiring enthusiasm and encouragement this thesis could not be successfully completed.

I would also like to thank my former supervisor Dr. Vahid Garousi for introducing me in the world of software engineering research. My heartfelt thanks and appreciation go to my colleagues in the Software Decision Support Laboratory for their benevolent effort to put valuable comments on this thesis. Discussing with them, from time to time, has really sharpened my thoughts’ process for the research.

Finally, this work is really indebted to the endless support from my parents, relatives, and one very special person. The distance among us was never a hindrance to their way of encouraging me. Last but not the least, the friends from my house, who are also from my home country, gave me a perfect milieu to study and flavor of my country to make me reminiscent of my family.
This thesis is dedicated to my father and mother for all their continuous support, love and inspiration.
### Table of Contents

Abstract .......................................................................................................................... ii
Publications ...................................................................................................................... iii
Acknowledgements ......................................................................................................... iv
Table of Contents ........................................................................................................... vi
List of Tables ................................................................................................................... ix
List of Figures and Illustrations ..................................................................................... xi
List of Abbreviations ..................................................................................................... xiii

#### CHAPTER ONE: INTRODUCTION

1.1 Motivation ................................................................................................................. 1
1.2 Difficulties in Evaluating Software Release Readiness ........................................ 2
1.3 Research Objectives ............................................................................................... 4
1.4 Overview of the Solution Approach ........................................................................ 5
1.5 Thesis Structure ....................................................................................................... 7

#### CHAPTER TWO: BACKGROUND AND LITERATURE REVIEW

2.1 Background .............................................................................................................. 8
  2.1.1 Iterative Software Development ........................................................................ 8
  2.1.2 Definition of Release in Iterative Development ............................................... 9
  2.1.3 Software Product Management ....................................................................... 10
  2.1.4 Definition of Release Readiness ....................................................................... 11
  2.1.5 Value of Continuous RR Evaluation in the Context of SPM ......................... 12
  2.1.6 Concept of Fuzzy Set Theory ............................................................................ 13
  2.1.6.1 The Use of Fuzzy Set Theory in Software Engineering ............................... 15
  2.1.7 Overview of the Aggregation Operators .......................................................... 18
  2.1.7.1 Mathematical and Behavioural Properties of Aggregation Operators .... 18
  2.1.7.2 Some Popular Aggregation Operators ....................................................... 21
  2.1.8 Overview of the Time Series Projection Methods ........................................... 23
  2.1.8.1 Simple Regression ..................................................................................... 24
  2.1.8.2 Exponential Smoothing ............................................................................ 24
  2.1.8.3 Time Series Decomposition ...................................................................... 25
  2.2 A Glossary of Terms Related to Software Release Readiness ............................ 26
  2.3 Literature Review .................................................................................................. 27
  2.3.1 Existing Release Readiness Approaches ......................................................... 27
  2.3.1.1 Checklist-based Approaches ..................................................................... 27
  2.3.1.2 Multi-dimensional Metrics Aggregation-based Approaches .................... 29
  2.3.1.3 Testing Metrics-based Approach ............................................................... 30
  2.3.1.4 Defect Prediction Model-based Approach ................................................ 33
  2.3.2 State of Validation of RR Metrics and Approaches ....................................... 34
  2.3.3 State of Tool Support for Evaluating Release Readiness ............................... 39
  2.3.4 Release Readiness Dimensions, Attributes and Metrics ............................... 40
  2.3.5 Discussion ....................................................................................................... 42
  2.4 Summary ............................................................................................................... 44

#### CHAPTER THREE: PROPOSED APPROACH ......................................................... 46
5.1.4.1 Defining the RR Attributes of RP2 ..................................................... 90
5.1.4.2 Collection of Data Using the Tool ................................................... 91
5.1.4.3 Defining Membership Functions ..................................................... 92
5.1.4.4 Defining Relative Weights of the RR Attributes ................................ 94
5.1.5 Analysis of the Results ........................................................................ 94
   5.1.5.1 CQ 1: Monitoring Degree of RR Over the Iterations ...................... 94
   5.1.5.2 CQ 2: Impact of Various Decision Strategies in Evaluating Overall RR ......................................................... 99
   5.1.5.3 CQ 3: Projected RR Score ............................................................... 102
   5.1.5.4 CQ 4: Providing Insights to the Product Manager about the Limiting Factors .................................................................................. 108
5.1.6 Threats to Validity ............................................................................... 111
   5.1.6.1 Construct Validity ............................................................................ 112
   5.1.6.2 Conclusion Validity .......................................................................... 112
   5.1.6.3 Internal Validity ............................................................................... 113
   5.1.6.4 External Validity .............................................................................. 113
5.2 Case Study 2: Monitoring Bottleneck Factors in Achieving RR ............... 113
   5.2.1 Definitions ......................................................................................... 114
   5.2.2 Case Study Objectives ...................................................................... 115
   5.2.3 Context of the Case Study Projects .................................................. 115
   5.2.4 Defining RR attributes, Metrics and Membership Function ............... 116
   5.2.5 Analysis of the Results ...................................................................... 117
      5.2.5.1 CQ 1: Frequency of Occurrence of BFs ....................................... 117
      5.2.5.2 CQ 2: Do Release Phases Significantly Influence the Frequency of BFs? .................................................. 118
      5.2.5.3 CQ 3: Are there any Significant Patterns in Transition between Consecutive BFs? .................................................. 119
   5.2.6 Threats to Validity ............................................................................. 120
5.3 Summary .................................................................................................. 121

CHAPTER SIX: SUMMARY AND FUTURE RESEARCH ......................................... 123
6.1 Summary of the Contributions ................................................................ 123
   6.1.1 RELREA Approach for Continuous Release Readiness Evaluation .... 123
   6.1.2 Studying Projection of RR with Time Series Projection Method ........ 124
   6.1.3 Implementation and Evaluation of RR analysis Tool for the Product Managers .................................................. 124
   6.1.4 Empirical Evaluation of Bottleneck Factors across Ten OSS Projects .. 125
6.2 Limitations and Applicability .................................................................. 125
6.3 Future Research Scope ........................................................................... 126

REFERENCES .................................................................................................. 128

APPENDICES A: CLARIFICATION OF THE CONTRIBUTIONS IN THE SECOND CASE STUDY .......................................................... 134
List of Tables

Table 2-1: A glossary of the terms used in this thesis .......................................................... 26
Table 2-2: Release readiness checklist based on RR criteria.................................................. 28
Table 2-3: Summary of the empirical evaluation of the RR metrics and approaches .............. 34
Table 2-4: Summary of the RR monitoring tools ................................................................. 40
Table 2-5: Classification of RR attributes and metrics found in the existing literature .......... 41
Table 3-1: RR dimensions and their related RR attributes ...................................................... 47
Table 3-2: Example of GQM paradigm for defining RR attributes and metrics [8]............. 47
Table 3-3: Commonly used shapes for the piecewise linear and piecewise monotonic membership functions ........................................................................................................... 51
Table 3-4: Selected linguistic quantifiers and corresponding $\alpha$ parameters ..................... 54
Table 3-5: Definition of the selected RR attributes, metrics definitions and relative weights used in the illustrative example............................................................................................ 62
Table 3-6: Definition of membership functions of the RR attributes .................................. 63
Table 3-7: Correspondence among decision strategy, order weights, degree of optimism, and aggregated RR scores ........................................................................................................... 67
Table 4-1: Technologies used in the development of the tool ............................................. 73
Table 5-1: An overview of the key characteristics of the case studies ................................. 86
Table 5-2: Duration of the studied iterations of RP2............................................................. 88
Table 5-3: Description and usage of the data sources of the case study project............... 89
Table 5-4: GQM based definition of RR attributes and their corresponding metrics of the case study project ..................................................................................................................... 91
Table 5-5: Definition of the membership functions for the selected RR attributes and metrics of the case study project ...................................................................................................... 93
Table 5-6: Definition of the relative weights of the RR attributes of the case study project..... 94
Table 5-7: A summary of the selected open source case study projects [56]....................... 115
Table 5-8: Determining RR metrics using GQM approach, corresponding goals, RR attributes, questions and acronyms [56]............................................................................... 116
List of Figures and Illustrations

Figure 2-1: Iterative software development life cycle ................................................................. 9
Figure 2-2: Release in iterative software development ................................................................. 10
Figure 2-3: Key components of the definition of software release readiness ............................. 12
Figure 2-4: Distribution of the studies according to the important RR dimensions .................... 41
Figure 3-1: Membership function of the fuzzy set A = {satisfaction of defect finding} ............ 49
Figure 3-2: Workflow of the proposed RELREA approach [8] .................................................. 57
Figure 4-1: MVC based three tier architecture of the tool .......................................................... 72
Figure 4-2: Tool interface for creating a new project ................................................................. 74
Figure 4-3: Settings interface of the tool for managing project information, RR attributes, membership functions, and relative weights .................................................. 74
Figure 4-4: Tool interface for managing the project info .......................................................... 75
Figure 4-5: Tool interface for managing the RR attributes and the metrics ................................ 76
Figure 4-6: Tool interface for managing the relative weights and customizing the pre-defined metrics .................................................................................................................. 77
Figure 4-7: Tool interface for managing the values of the metrics ............................................. 77
Figure 4-8: Tool interface for managing membership functions .............................................. 79
Figure 4-9: High level use cases of the tool ............................................................................. 80
Figure 4-10: Tool interface for monitoring and analyzing RR ..................................................... 81
Figure 4-11: Tool interface for visualizing long term trends with the moving average smoothing .................................................................................................................. 81
Figure 4-12: Tool interface for studying projection of RR ......................................................... 82
Figure 4-13: Tool interface for gaining insights about the limiting factors .............................. 83
Figure 5-1: Screenshot of the “manage metrics values” interface of the tool for the case study project .................................................................................................................. 92
Figure 5-2: Comparison of RR of RP2 at the end of the iterations ............................................. 96
Figure 5-3: Screen of the tool showing the evolution of the overall RR and the degree of satisfaction of the individual RR attributes of RP2 over the iterations ............................................. 97

Figure 5-4: Screen of the tool visualizing individual attributes status at the end of the iterations ........................................................................................................................................ 97

Figure 5-5: Screenshot of the tool demonstrating the impact of various decision strategies at t = 44 weeks ................................................................................................................................................ 101

Figure 5-6: Screenshot of the tool presenting the distribution of OWA weights for various decision strategies for the RR at t = 44 weeks of the case study project ........................................... 101

Figure 5-7: Screenshot of the RR time series of the case study project .......................................................... 103

Figure 5-8: Moving average smoothing for visualizing the long term trends of the RR time series of the case study project ........................................................................................................... 104

Figure 5-9: Linear trend analysis of RR of RP2 .................................................................................................. 104

Figure 5-10: Autocorrelation plot of RR time series up to 16 lags ................................................................. 105

Figure 5-11: Screenshot of the tool demonstrating the Actual RR and the Projected RR ................. 106

Figure 5-12: Projection of RR of RP2 at the next week .................................................................................. 107

Figure 5-13: Projection of RR at the approximate release week of RP2 ......................................................... 107

Figure 5-14: Screen of the tool showing the impact of the individual RR attributes on overall RR over time .......................................................................................................................... 109

Figure 5-15: Occurrence frequency of negative impact of the RR attributes on overall RR .......... 110

Figure 5-16: A Pareto chart illustrating BNF of the studied RR attributes [56] ............................................ 117

Figure 5-17: Transition graph showing relative frequency of BFs transitions among RR attributes across the ten projects [56] ........................................................................................................ 120
### List of Abbreviations

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>RR</td>
<td>Release Readiness</td>
</tr>
<tr>
<td>RO</td>
<td>Research Objective</td>
</tr>
<tr>
<td>GQM</td>
<td>Goal Question Metric</td>
</tr>
<tr>
<td>HLT</td>
<td>Holt’s Linear Trend</td>
</tr>
<tr>
<td>HW</td>
<td>Holt Winter</td>
</tr>
<tr>
<td>OWA</td>
<td>Ordered Weighted Average</td>
</tr>
<tr>
<td>OSS</td>
<td>Open Source</td>
</tr>
<tr>
<td>SPM</td>
<td>Software Product Management</td>
</tr>
<tr>
<td>PM</td>
<td>Product Manager</td>
</tr>
<tr>
<td>ISPMA</td>
<td>International Software Product Management</td>
</tr>
<tr>
<td>RDBMS</td>
<td>Relational Database Management System</td>
</tr>
<tr>
<td>SCCM</td>
<td>Software Configuration and Change Management</td>
</tr>
<tr>
<td>LOC</td>
<td>Line of Code</td>
</tr>
<tr>
<td>DFR</td>
<td>Defect Find Rate</td>
</tr>
<tr>
<td>MF</td>
<td>Membership Function</td>
</tr>
<tr>
<td>MCDM</td>
<td>Multi Criteria Decision Making</td>
</tr>
<tr>
<td>RIM</td>
<td>Regular Increasing Monotone</td>
</tr>
<tr>
<td>PR</td>
<td>Projected Readiness</td>
</tr>
<tr>
<td>MVC</td>
<td>Model View Controller</td>
</tr>
<tr>
<td>ORM</td>
<td>Object Relational Method</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>MAPE</td>
<td>Mean Absolute Percentage Error</td>
</tr>
<tr>
<td>BF</td>
<td>Bottleneck Factor</td>
</tr>
<tr>
<td>BNF</td>
<td>Bottleneck Frequency</td>
</tr>
<tr>
<td>BNTF</td>
<td>Bottleneck Transition Frequency</td>
</tr>
<tr>
<td>CQ</td>
<td>Case-study Question</td>
</tr>
<tr>
<td>HP-FIR</td>
<td>High Priority Feature Implementation Ratio</td>
</tr>
<tr>
<td>LP-FIR</td>
<td>Low Priority Feature Implementation Ratio</td>
</tr>
<tr>
<td>IIR</td>
<td>Improvement Implementation Ratio</td>
</tr>
<tr>
<td>CCR</td>
<td>Code Churn Rate</td>
</tr>
<tr>
<td>BFR</td>
<td>Bug Fix Ratio</td>
</tr>
<tr>
<td>BSR</td>
<td>Build Success Rate</td>
</tr>
<tr>
<td>PCR</td>
<td>Pull Request Completion Ratio</td>
</tr>
</tbody>
</table>
Chapter One: Introduction

1.1 Motivation

To achieve business success of products, in-time and in-quality of releasing software is a key concern of software companies [1]. A slip in release may cause substantial loss in revenue and in the worst case, failure of the complete software product [2]. One of the main challenges of the software product manager is knowing the current state of the readiness of the product based on objectivity [3]. In most cases, Release Readiness (RR) is evaluated at the end of the development cycles (i.e. milestones, sprints, etc.) relying on a set of isolated metrics. This approach has some major consequences:

- Firstly, there is no continuity in monitoring product readiness.
- Secondly, any problem related to release issues cannot be addressed proactively by the development team.
- Thirdly, if RR falls below expectations it is unknown which are the limiting factors.

Therefore, to support informed release decision and better product management, it is essential to evaluate and analyze RR starting from the initial phases of the development lifecycle. Monitoring RR through the release cycle is also important for making projection of RR at the release date. It will allow the product manager to make better decision about the release window of the software product. In addition, the product manager can gain insights about the factors that need to be tuned for improving RR. Based on these ideas, this thesis mainly focuses on the continuous evaluation and monitoring of software RR so that the product manager can identify and address the RR issues proactively.
1.2 Difficulties in Evaluating Software Release Readiness

Due to the complex and unpredictable nature of software development, evaluating RR at any point in time during the release cycle is a challenging task. Some of the difficulties and fundamental issues in evaluating software releases readiness are listed below:

- **Unclear definition of RR:** One of the fundamental issues is that the definition of software release readiness is not well articulated in the literature. Most of the related works [1, 2, 4-6] in this domain mainly gauge with metrics related to defect tracking and testing in order to define release readiness. However, as a time dependent attribute of a software product, release readiness measures should include all important aspects (i.e. functionality, source code quality, testing, and documentation) of the software product [7]. This thesis follows [8], where RR is defined as an aggregated objective measure of the degree of satisfaction of the readiness attributes related to the current implementation (e.g. degree of functionality, degree of code stabilization, etc.) and quality (e.g. degree of test execution, degree of defect finding, etc.) of the software. According to this definition, the main challenge in evaluating RR is determining the degree of satisfaction of the RR attributes based on monitoring key product and process performance metrics.

- **Lack of knowledge about RR attributes and metrics:** Attributes and metrics for design evaluation [9, 10], code complexity analysis [11, 12], test allocation [13, 14], maintainability [15, 16], and general code health [17, 18] are defined, implemented, and validated with industry projects. However, the use of software product and process related metrics in evaluating RR is not well established yet. Analysis of related works (see Chapter Two:) of this domain shows that only a few defect tracking and testing related metrics (e.g. number of open defects, defect find rate, test execution rate, and test
passing rate) are identified and implemented for evaluating RR in industry projects. RR metrics related to implementation of functionality, source code quality, and documentation status dimensions are not clearly defined and validated yet. Therefore, defining the right set of RR attributes and metrics at the planning phase of the release is a challenging task in evaluating RR.

- **Fuzziness in evaluating RR attributes:** In the context of RR evaluation, it is important to evaluate the degree of satisfaction of the RR attributes. However, “degree of satisfaction” is not a well-defined term. This type of property is defined as fuzzy because precise measurement of this property cannot be defined in principal. For example, assume satisfaction of defect finding is a RR attribute and defect find rate is the associated objective measure. At a particular point of time, if the value of defect find rate is 0.8 (defects per day), then what will be the degree of satisfaction of defect finding? It is difficult to provide an exact value for the “degree of satisfaction” of this RR attribute.

- **Proper choice of aggregation operator:** Aggregation operators for source code level metrics are identified and validated [19]. However, which aggregation operator is suitable to determine the overall degree of RR by combining the individual degree of satisfaction of the RR attributes has not been studied yet. Thus, the proper choice of aggregation operator is another challenge in evaluating RR. Ideally, an average type flexible aggregation operator is required which should have the capability of integrating both relative weights of the attributes and the behaviour of the decision makers along with general mathematical properties. In this thesis, applicability of Ordered Weighted Average (OWA) [20] operators in determining overall degree of RR is studied in real life software project.
1.3 Research Objectives

Based on the above discussion, the goal of this thesis is to design and develop an analytical approach for evaluating RR at any point of time in the release cycle, making projection of RR at the release date and making suggestions about the factors limiting the RR. To achieve this goal, the following research objectives are defined:

- **RO1**: To review the existing literature related to software release readiness.
  - **RO1.1**: To identify and classify the existing release readiness facets, i.e., tools, techniques, methods, approaches, etc.
  - **RO1.2**: To identify and organize the RR attributes and metrics into RR dimensions.

- **RO2**: To design an approach that allows evaluating RR at any point in time during the release cycle.
  - **RO2.1**: To select the RR attributes in a systematic way.
  - **RO2.2**: To determine the degree of satisfaction of the RR attributes by monitoring key product and process performance metrics.
  - **RO2.3**: To select an appropriate aggregation operator for combining the individual level of satisfaction of the RR attributes.

- **RO3**: To study the projection of RR at the release date so that the product manager can make proactive decisions if any RR issue is identified.
  - **RO3.1**: To identify an appropriate time series projection methods for studying the projection of RR at the release date.

- **RO4**: To design and implement a RR assessment tool which integrates the continuous RR evaluation method from RO2.
RO4.1: To automate the steps involved in the RR evaluation approach from RO2.

RO4.2: To design an interactive visual interface for monitoring, analyzing, and projecting RR.

RO5: To perform case studies with real life software projects to evaluate the RR evaluation approach from RO2 and the RR assessment tool from RO4.

RO5.1: To demonstrate the applicability of the approach from RO2 and RR assessment tool from RO4 in investigating RR issues with real life software project.

RO5.2: To identify the frequencies and patterns of the bottleneck factors in achieving RR using the approach from RO2.

1.4 Overview of the Solution Approach

To address the challenges discussed above, in this thesis, an analytical RR evaluation approach called RELREA is presented. The key characteristics of RELREA are:

- Project specific RR attributes and metrics are selected in a systematic way.
- Unlike the existing methods (see Section 2.3.1), overall RR is determined by aggregating and evaluating the degree of satisfaction of the RR attributes.
- Continuous visibility on RR status.
- Projection of RR into future.

In the proposed approach, product specific RR attributes can be defined from four important RR dimensions: implementation status, testing scope and status, source code quality, and documentation status. RR dimensions are identified by reviewing and categorizing the RR attributes found in the existing literature and tools. These dimensions cover both important artefacts (e.g., features, source code, documents, etc.) and development activities (e.g., coding,
building, testing, etc.). The metrics corresponding to the RR attributes are defined based on the Goal Question Metric (GQM) [21] paradigm which is the de-facto standard for performing software measurement. In the context of RR, our goal is to evaluate the status of the overall readiness. The goal is refined by the questions corresponding to the RR attributes. Available data associated with every question are used to answer them in a quantitative way.

The concept of fuzzy set theory introduced by Zadeh [22] is employed for evaluating the degree of satisfaction of the RR attributes based on the objective measures defined using the GQM [21] paradigm. It allows evaluating the degree of satisfaction of the RR attributes in a 0 to 1 scale where 0 indicates not satisfied at all and 1 indicates full satisfaction of the attribute. Methods for selecting appropriate shapes and parameters of the membership functions are also suggested. As an aggregation operator, Ordered Weighted Average (OWA) [20] is applied. It allows the product manager to consider the relative weights of the attributes and product manager’s desired decision strategy (e.g. optimistic, neutral, pessimistic etc.) in evaluating overall RR.

In order to study the projection of RR at the release date, monitored RR over time is used. Holt’s Linear Exponential Smoothing (HLT) [23] and Holt-Winter (HW) [23] methods are employed for RR projection because trends and correlation are two key characteristics of the RR time series.

In order to evaluate the proposed approach, two case studies are performed that demonstrate the applicability of the proposed approach in real-life scenarios. The first case study evaluates the applicability of the proposed method and the prototype tool in monitoring and investigating RR issues from the product manager’s perspective. It also demonstrates the accuracy of the time series projection method for making projection of RR at the release date.
The second case study shows the applicability of the RELREA approach in monitoring bottleneck factors in the domain of Open Source (OSS) projects. Occurrence frequency and patterns of the bottleneck factors are empirically evaluated for ten OSS projects hosted in Github. The results of the case studies show that the RELREA and its implementations are able to assist the product manager in monitoring, projecting and analyzing RR through the release cycle. The proposed approach is also helpful in providing insights to the product manager about the current limiting factors if the expected readiness is not achieved.

1.5 Thesis Structure

- **Chapter 1** describes the motivation of this work and the difficulties in evaluating RR, introduces the research objectives, and provides an overview of the solution approach.
- **Chapter 2** provides a review of the related works depicting the limitations of the current approaches and outlines the necessity of the new approach.
- **Chapter 3** describes the fundamental concepts used in the proposed release readiness evaluation approach. The workflow of the proposed approach and an illustrative example are also presented to support deeper understanding of the approach.
- **Chapter 4** describes a prototype RR assessment tool which allows the product manager to monitor and investigate RR based on the approach presented in Chapter 3.
- **Chapter 5** evaluates the applicability and limitation of the proposed RR evaluation approach through two case studies with real life software projects. Demonstration of the tool for assisting the product manager to monitor and investigate the RR from various perspectives is also provided in this chapter.
- **Chapter 6** concludes the thesis, outlines the main contribution, applicability, and future research scope of the work.
Chapter Two: **Background and Literature Review**

In this chapter, an overview of the background concepts related to software release readiness is described. In addition, existing RR approaches, metrics, tools, and their empirical evaluation are identified and described by reviewing the existing literature. Discussion is also made on the gaps of the related works.

**2.1 Background**

In order to facilitate a better understanding of the contribution of this thesis it is essential to provide an overview of some fundamental concepts such as the principals of iterative development methodology, the definition of release in iterative development, the software product management, roles and responsibilities of software product manager, and the concepts and use of fuzzy set theory in software engineering. In the following sections an overview of these concepts are provided.

**2.1.1 Iterative Software Development**

Being late and budget overrun are very common phenomena in software development [24]. Consequently, management always looks for ways to plan and control the development activities. Many types of software development model (i.e. waterfall model, iterative model) are found in software engineering body of knowledge. The most widely adopted is the iterative development paradigm which supports the development of software incrementally with shorter iterations or cycles. It enables feedback from the stakeholders to the ongoing development activities of the software. This process provides the development team and management staffs a better understanding the system as well as customer’s expectation while development is progressing. As a result, the end product has greater chance to meet the expectation of all
stakeholders. Figure 2-1 shows the development lifecycle of the iterative approach with a four phase development process consisting of design, code, testing, and delivery.

![Diagram of iterative software development life cycle](image)

**Figure 2-1: Iterative software development life cycle**

**2.1.2 Definition of Release in Iterative Development**

Releasing software is the process of delivering the product into the operational environment for actual usage by its end users [1]. In iterative development, release of the software product consists of several iterations. Duration of a release cycle typically range from 2 to 12 months. In some iterative development methods, the end of an iteration is called an *iteration release* [25] where a stable, integrated, and tested partially completed version of the product is internally released. An illustration of the release process in iterative development is presented in Figure 2-2. As shown in Figure 2-2, a release cycle starts with the process of release planning where all important stakeholders of the project participate to prioritize and select an optimal set of features to be implemented in the next release considering the technical, resource, risk and budget constraints are met [26]. In the each iteration, a subset of the assigned features are implemented, tested, and demonstrated to the stakeholders of the project. In this way, over the iterations the software product evolves and become more mature and stable.
Usually at the end of the last iteration, a complete version of the software product is released for actual usage in the operational environment. This research focuses on evaluating and monitoring the state of RR through the iterations so that release related issues can be identified and addressed by the product manager at the early stage of the development.

![Figure 2-2: Release in iterative software development](image)

### 2.1.3 Software Product Management

Software Product Management (SPM) is “the discipline and role, which governs the software product (or solution or service) from its inception to the market/customer delivery in order to generate biggest possible value to the business” [27]. It creates a link between business and technical perspectives in the development of software products. Scope of the SPM includes product strategy, product planning, development, marketing, sales and distribution, service and supports, and strategic management. According to International Software Product Management Association (ISPMA) [28], SPM is the key success factor for the companies which develops or sells software product or services. Industry evaluation of SPM practices [29] reveals that software organizations are benefited by the application of SPM with reduced delays and faster product acceptance in the market.
SPM activities are surrounded and performed by the software product manager. She is also called the “embedded CEO” of the product. Her primary responsibility is to prepare and implement the business case of a software product with the consideration of technical aspects. She evaluates the products or product releases with respect to the overall contribution to business success. With the other stakeholders she has to also decide about the contents of a release, timeframe of a release, target market and price of the product. As described in [30], the product manager has to balance between projects, people and politics and his or her primary tools are road maps, requirements, milestone reviews, and the business case. According to a survey conducted in Microsoft [31], one of the main challenges of product managers in making informed decision is the lack of analytical tools and techniques for interpreting the large volume of development data stored in source code repositories to bug database. Therefore, this thesis focuses on objective evaluation of RR based on current development data to assist product managers in gaining insight about the current state of the software and making decision about product releases.

2.1.4 Definition of Release Readiness

RR is a time dependent attribute of software product which reflects the status of the current implementation and quality of the software. However, little is known about the formal definition of RR. Related works of this domain mainly gauge with defect tracking metrics to define RR. Ideally, RR should be defined with respect to the level of satisfaction of the product manager about the implemented functionality and quality of the product at a particular point of time in the release cycle. Based on this idea, from the product management perspective, this study defines RR as an objective measure which is determined by evaluating and aggregating the degree of satisfaction of a set of individual readiness attributes. Readiness attributes are the
aspects of software by which overall RR can be defined and judged. According to this definition, the main problem of evaluating release readiness is evaluating the degree of satisfaction of the RR attributes based on monitoring key product and process performance metrics. The key components (i.e. RR attributes, metrics, degree of satisfaction, and aggregation operator) of the definition are illustrated in Figure 2-3. Details about evaluating RR with this definition are described in Chapter 3.

![Diagram](image)

**Figure 2-3: Key components of the definition of software release readiness**

### 2.1.5 Value of Continuous RR Evaluation in the Context of SPM

Successfully applying iterative development methodology for releasing in-time and in-quality software product is a challenging task for the management staffs. It requires the product management and release engineering [32] team, i.e., project manager, product manager, release manager, and other to provide direction to the project so that each iteration progressively contributes in delivering the complete product. Otherwise, the product cannot be delivered within
the release date that may cause substantial loss in revenue and in the worst case, failure of the complete software product. Hence, knowing the current state of the RR of the product based on objectivity at any point of time is essential for the product manager of the software organizations. In [33], Port et al. illustrated the value of monitoring RR for final project success. Value of continuous RR evaluation is also discussed in [25]. The following list some of the important ones:

- It demonstrates which factors are not performing sufficiently well (e.g. related to test performance or implementation effort) and are likely to limit release readiness at any point in time during the release cycle.
- It allows addressing any problem related to release issues proactively by the development and management team.
- It enables forecasting product availability to the internal and external customers.
- In the competitive business world, it helps to preserve the product release window. It is also important for maintaining confidence customer in the product delivery cycles.

### 2.1.6 Concept of Fuzzy Set Theory

The concept of fuzzy set theory was introduced by Zadeh [22] in 1965 to represent non-statistical uncertainty and vagueness associated to data and information. Since then, it has been widely used to solve problems in various decision making environments. In software engineering, fuzzy set theory has been applied successfully to deal with the uncertainty associated to the aspects such as cost estimation [34], reliability prediction [35], and imprecise requirement analysis [36]. Definition of a fuzzy set is stated from [22] in the following:

**Definition 2-1:** A fuzzy set $A$ in a universe of discourse $X$ is characterized by a membership function $\mu_A(x)$ which associates a real number in the interval $[0, 1]$ to each element
that claims partial membership in the set. The function value $\mu_A(x)$ is termed as the “grade of membership” $x$ of in $A$.

Example: Let, $R$ be the real line, and $A$ be a fuzzy set of numbers which are much greater than 1. The term much is subjective and it is not possible to define it with precise values. Using the concept of membership function stated in Definition 2-1, one can provide a precise, subjective, characterization of $A$ by specifying $\mu_A(x)$ as a function on $R$. Some values of such a function might be: $\mu_A(0) = 0; \mu_A(5) = 0.01; \mu_A(100) = 0.95; \mu_A(500) = 1$.

The basic concept of the membership function used in fuzzy set theory has evolved over time. The operations (union, intersection, difference, complement, etc.) of crisp set are extended for the fuzzy sets where the operator’s works on the membership degree of the elements of the sets instead of the elements itself. In [37], Zadeh also introduced the concept of linguistic variables whose values are words or sentences in a natural or artificial language. For example, “age” is a linguistic variable where the values (i.e. young, not young, very young middle aged, not middle aged, old, not old, very old, etc.) are linguistic rather than numerical. In the following section, some usage of the linguistic variables in software engineering is discussed.

Another area or extension of fuzzy set theory is the concept of fuzzy logic which deals with reasoning that is approximate rather than fixed and exact. Compared to traditional Boolean values (1 or 0) in binary sets, fuzzy logic variables are based on “degree of truth” that ranges in degree between 0 and 1. As the mechanism is similar to our brain, it has been widely used in various expert systems and fuzzy controllers [38]. Details about the fuzzy logic and fuzzy systems can be found in [39].

From the beginning of the fuzzy sets, a criticism was made about the fact that the membership function of a basic fuzzy set has no uncertainty associated with it. In other words, it
is not possible to define the membership function with precision due to its subjective nature in principle. To overcome this limitation, in 1975 Zadeh [37] proposed more sophisticated kind of fuzzy sets which he called a type-2 fuzzy set. A type-2 fuzzy set allows incorporating the uncertainty about the membership functions of basic fuzzy set. In this type of fuzzy sets, membership functions are three-dimensional that provides additional degrees of freedom that make it possible to directly model uncertainties. However, the three-dimensional membership functions are difficult to understand and use because: 1) it is difficult to draw three-dimensional function; 2) there is lack of well-defined terms for communicating the idea of three-dimensional membership functions; and, 3) it is computationally more complicated than using the basic fuzzy sets [40]. As a result, the concept type-2 sets is less popular than the basic fuzzy sets.

In this thesis, mainly the concept of membership function of the basic fuzzy sets is applied. As described in Section 1.2, fuzziness in evaluating “degree of satisfaction” of the RR attributes based on objective measures is dealt with the concept of membership function stated in Definition 2-1. Details of the application of the membership function are described in Section 3.1.4.

2.1.6.1 The Use of Fuzzy Set Theory in Software Engineering

Fuzzy set theory has been successfully applied in various fields such as machine learning [41], multi criteria decision making [42], risk investigation [43], etc. The success of fuzzy set theory during the past decades has drawn the attention of the software engineers. Specifically, fuzzy set theory is used to deal with the uncertainties arising from different stages of the software development life cycle and various aspects (i.e. cost estimation [44], quality estimation [45], reliability modeling [46], requirement analysis [47], etc.) of software engineering. An overview of the application of fuzzy set theory in software engineering is provided below.
In [48], uncertainty associated with the software release planning is modeled with fuzzy set theory. It fuzzifies release planning aspects such as resource estimate for requirements, resource capacities, and the objective function. As a result, stakeholders of the project can define these aspects in the inherent uncertain software development environment. It also allows better trade-off analysis between the satisfaction of the objective and the satisfaction of the resource constraints. Product managers can choose the appropriate release plan with a good balancing among the release planning constraints.

Fuzzy set theory is used to rank the software engineering metrics. In [49], imprecise nature of the software engineering metrics and the ranking criteria (i.e. cost, benefit, credibility, experience, etc.) are modeled with triangular fuzzy numbers. Relative importance of the evaluation criteria are also modeled with fuzzy numbers. The developed fuzzy based methodology can be used to systematically rank the existing software engineering metrics with imprecise and inexact data.

The concept of linguistic quantifier and fuzzy rule based inference is applied in software cost prediction [44]. Software metrics which have impact on cost (or effort) such as size or complexity are modeled with linguistic variables. As an approximate reasoning technique fuzzy rules are used. Based on the fuzzy input of the inference engine, the conclusion is also made as fuzzy set, emphasizing the uncertainty of the cost. The rules developed for approximate reasoning reduces the dependence on historical data.

The concept of fuzzy linguistic terms and fuzzy logic is used in analyzing imprecise requirements. Clients usually describe the requirements using natural language. Fulfillment of a requirement cannot be measured with crisp values. In [47], fuzzy logic is employed in identifying the implicit relationship between requirements. It also supports making trade-off
between conflicting requirements. The approach is also useful to discover the priority between conflicting and imprecise requirements.

The fuzzy software reliability prediction model is proposed, where a software reliability model is represented as a mathematical optimization problem [46]. For the large and complex system, it is tough to remove all faults from it. Therefore, a time interval is estimated during which no failure will occur. Software reliability models (SRM) are used to determine the appropriate time interval. In [46], fuzzy optimization technique is used to select the best software reliability model.

Fuzzy set theory is employed to develop software quality assessment models [45]. The subjectivity and vagueness of the software quality attributes (i.e. maturity, accuracy, interoperability, stability, understandability, etc.) are represented with linguistic terms and fuzzy numbers. In [45], a fuzzy quality evaluating algorithm is presented where linguistic quantifier based rating scheme (i.e. extra low, very low, low, middle, high, very high, extra high) is used to quantify the degree of satisfaction of the imprecise quality attributes related to user satisfaction. The fuzzy set based evaluation reduces the degree of subjectivity of the evaluator.

The above discussion shows that fuzzy sets, fuzzy numbers, fuzzy linguistic variable and fuzzy rule based inference engine are widely used in software engineering. But it is not yet used to evaluate the degree of satisfaction of the RR attributes based on the available objective measures. In this research, the concept of fuzzy set theory is applied to capture the vagueness in determining the degree of satisfaction of the RR attributes based on the values of the objective metrics. More about the application of fuzzy set theory in RR evaluation is described in Section 3.1.4.
2.1.7 Overview of the Aggregation Operators

Aggregation operators are used to combine several input values into a single output value. In the context of this thesis, aggregation operator plays an important role in determining the overall RR evaluation of a software product by combining the individual evaluation of the RR attributes. Aggregation operation can be formalized as the following, where the output \( y \) and inputs \( a_1, a_2, ..., a_n \in [0, 1] \).

\[
Z = Aggr(a_1, a_2, ..., a_n)
\]  

(2-1)

In the above formulation, \( Aggr \) operator maps \( n \) dimensional input values between unit interval \([0, 1]\) to one dimensional value between 0 and 1: \([0, 1]^n \rightarrow [0, 1]\). An aggregation operator has to satisfy the following three properties:

i. \( Aggr(a) = a \)

ii. \( Aggr(0, ..., 0) = 0 \) and \( Aggr(1, ..., 1) = 1 \)

iii. \( Aggr(a_1, a_2, ..., a_n) \leq Aggr(c_1, c_2, ..., c_n) \) if \( x_i \leq c_i \) \( i = 1 \ldots n \)

An overview of the mathematical and behavioural properties of the aggregation operator is provided in the following sub-sections.

2.1.7.1 Mathematical and Behavioural Properties of Aggregation Operators

In literature [50, 51], mathematical and behavioural properties are studied to reflect different decision scenarios. By looking at those properties a proper aggregation operator can be chosen. The properties discussed in [52] are restated in the following:

- **Boundary Conditions**: The following equations are expects to be satisfied with this property:
The first condition means if we observe all the criteria are false or not satisfactory, the total aggregation has to be false or not satisfactory. Conversely, the second condition can be interpreted as if all the criteria are completely satisfactory then the total aggregation has to be also completely true or satisfactory.

- **Monotonicity**: In this property is expected to satisfy the following equations:

\[
Aggr(a_1, a_2, ..., a_n) \leq Aggr(c_1, c_2, ..., c_n) \text{ if } x_i \leq c_i \text{ } i = 1 ... n
\]

(2-4)

It indicates if the satisfaction level of one criterion increases then the final aggregation increases.

- **Associativity**: This property allows the criteria to be aggregated by different packages without influencing the overall evaluation. With three arguments this property can be written as follows,

\[
Aggr(a_1, a_2, a_3) = Aggr(Aggr(a_1, a_2), a_3) = Aggr(a_1, Aggr(a_2, a_3))
\]

(2-5)

- **Symmetry**: This property is also called commutativity or anonymity. In this case order of the arguments has no influence on the results. The property is stated in the following equation for every permutation * of \{1, 2...n\}:

\[
Aggr(A_{*(1)}, A_{*(2)}, ..., A_{*(n)}) = Aggr(A_1, A_2, ..., A_n)
\]

(2-6)

- **Neutral element**: This property expects if the aggregation operator has a neutral element \(e\), then it can be used as an argument that should not have any influence on the final result:
Idempotence: This property is also known as unanimity or agreement. It indicates if the same values is aggregated n times then the final result is expected the same value:

\[ Aggr(a_1, ..., a_{n-1}) = Aggr(a_1, ..., a_{n-1}) \] (2-7)

\[ Aggr(a, a, ..., a) = a \] (2-8)

Compensation: This property is also known as Pareto property. It expects that the aggregated score being between the maximum and minimum values of the arguments to be aggregated. It can be formalized as the following:

\[ \min(a_1, a_2, ..., a_n) \leq Aggr(a_1, a_2, ..., a_n) \leq \max(a_1, a_2, ..., a_n) \] (2-9)

Besides the above-mentioned mathematical properties, it is also meaningful to express the behaviour of the decision-maker. Two behavioural properties of the aggregation operators are stated in the following:

- **Decisional behaviour:** In this property, it is expected that the aggregation operator is able to include the behaviour of the decision-maker. For example: tolerant, optimistic, pessimistic, etc. With the optimistic decision strategy decision-maker is satisfied if some of the criteria are satisfied. Conversely, pessimistic decision strategy is used it when all the criteria need to be satisfied.

- **Weights on the arguments:** This property expects the possibility to express weights on the arguments. It helps decision-maker to put more emphasize on the important criteria in determining the overall evaluation.
2.1.7.2 Some Popular Aggregation Operators

In the context of this thesis, study on the available aggregation operators is important to choose an appropriate operator for determining overall RR evaluation. In the following some popular aggregation operators are introduced:

- **The arithmetic mean**: The most commonly used aggregation operator is the simple arithmetic mean or average. Formally it is stated in Eq. The aggregated score is smaller than the greatest argument and bigger than the smallest argument. Thus, the final aggregation is the middle value. As discussed in Section, this property is known as the compensation property. Other mathematical properties include: monotonicity, continuity, symmetry, associativity, and idempotence. However, it does not support neutral element and behavioral properties.

\[
Aggr(a_1, a_2, ..., a_n) = \frac{1}{n} \sum_{i=1}^{n} a_i
\]  

(2-10)

- **The weighted mean**: The weighted mean operator allows placing weights on the arguments to be aggregated. But to support weights with the arguments it loses the property of symmetry. It also does not allow the decision maker to include decision strategy in the aggregation process.

\[
Aggr_{w_1, w_2, ..., w_n}(a_1, a_2, ..., a_n) = \frac{1}{n} \sum_{i=1}^{n} (w_i a_i)
\]  

(2-11)

Where the weights are non-negative and \( \sum_{i=1}^{n} w_i = 1 \)

- **The minimum and the maximum**: The minimum operator (MIN) gives the smallest value of the arguments to be aggregated, while the maximum operator (MAX) gives the greatest one. Though the MIN and MAX operators are simple, they are meaningful in
different decision making context. For instance, in multi-criteria decision making environments, the MIN operator can translates a conjunctive (AND) attitude. And the MAX operator has a disjunctive (OR) behavior. This behavior allows to perform trade-offs between competing goals when compensation is allowed. Mathematical properties such as boundary conditions, monotonicity, symmetric, associative, and idempotent are supported by these operators. In [22], Zadeh suggests to combine the membership degree of fuzzy sets using the MIN operator. Since then, it has been applied in many decision making problems.

- **The Ordered Weighted Average (OWA) Operators:** The family of OWA operators is first introduced by Yager in [20]. Properties of these aggregation operators include idempotence, continuity, monotonicity, neutrality and compensativeness. The formal definition of the OWA operator is stated in the following:

\[
OWA(a_1, a_2, ..., a_n) = \sum_{i=1}^{n} w_i \cdot b_i
\]

Where \( a_1, a_2, ..., a_n \) are the attributes satisfying \( a_i \in [0,1] \); \( w_1, w_2, ..., w_n \) are the order weights satisfying \( w_i \in [0,1] \), \( \sum_{i=1}^{n} w_i = 1 \) and \( b_i \) is the ith largest element in the collection \( a_1, a_2, ..., a_n \). An important issue in the OWA is the ordering of the attributes scores \( a_i \) that makes it a nonlinear aggregation operation. Yager [20] showed that OWA operators lie between maximum (“or”) and minimum (“and”) of the scores to be aggregated. This allows an easy adjusting of the degree of optimism/pessimism by the appropriate choice of weights \( w_i \). A measure of the optimism or orness associated with \( W \) is stated in the following:
\[ \text{optimism/orness}(w_i) = \frac{1}{n-1} \sum_{i=1}^{n} (n-i)w_i \] (2-13)

The optimism/orness \((w_i)\) closer to one (similar to ‘or’ operator) adopts the optimistic decision strategy since higher values are given more importance in the aggregation.

Analogously, the closer this value is to zero (similar to ‘and’ operator) leads to the adoption of a pessimistic attitude where lower scores are given more importance in the aggregation. Therefore, different attitudes regarding optimism/pessimism can be incorporated by choosing an appropriate set of order weights \(w_i\).

In this thesis, aggregation operators are applied to combine the satisfaction degree of the individual RR attributes of a software product into an overall evaluation. Details about the selection and application of the aggregation operators are discussed in Section 3.1.5.

### 2.1.8 Overview of the Time Series Projection Methods

A time series is a sequence of observation on a particular variable typically measured at successive points in time. The measurements are usually taken at any regular interval (i.e. every hour, day, week, month, or year). Example of time series include: daily stock prices, monthly rainfall, annual profits of a company, etc. In the context of this thesis, monitored release readiness scores over time is the example of a time series. The aim of projecting time series data is to estimate how the sequence of observations will continue into future. In literature [53], different types of time series projection methods are proposed based on the patterns (i.e. trend patterns, seasonal patterns, and cyclic patterns) of a time series. Existing projection methods can be organized into three board categories: simple regression, time series decomposition, and exponential smoothing. An overview of these methods is provided from [54] in the following sub-sections.
2.1.8.1 Simple Regression

The basic time series projection method is the use of simple regression. In this technique, it is assumed that the forecast variable has a linear relationship with time. It is called “simple” regression because it only allows one predictor variable. In this method, a simple linear model is used to obtain projected values using the following equation:

\[ y_t = a_0 + a_1 x \]  \hspace{1cm} (2-14)

Where \( x \) is the value of the time interval for which we want to make projection. The equation of the line is obtained by applying regression technique (i.e. least squares estimation) on the time series data. The idea is that if we input a value of \( x \) the equation we obtain a corresponding projected score \( y_t \). This is also called the “fitted value” of \( y_t \) for the previously observed values. The technique is useful when a clear linear trend pattern is identified in the time series.

2.1.8.2 Exponential Smoothing

It is the most popular and successful time series projection methods. It was proposed in the late 1950s by Brown, Holt, and Winter [53]. In this method, projection is made using the weighted average of the past observations, with the weights decaying exponentially as the observations get older. It means the more recent the observation the higher the associated weight. Based on this idea, a set of methods is proposed in the literature. The most commonly used are: Simple Exponential Smoothing (SES), Holt’s Linear Trend (HLT) method, and Holt-Winters (HW) seasonal method. The selection of a method generally based on identifying key components (trend and seasonal) of the time series. For example, the SES method is appropriate when the time series has no trend or seasonal pattern. On the other hand, HLT method can be applied if the time series only has a linear trend component. HW method is selected when both
the trend and seasonal component is found in a time series. Details about these methods are discussed in Section 3.3.

2.1.8.3 Time Series Decomposition

This method assumes that actual time series $y_t$ at period $t$ is a function of three components: a trend component ($T_t$); a seasonal component ($S_t$); and an irregular or error component ($\varepsilon_t$). In order to obtain the observed values, aggregation of these components is described by an additive or a multiplicative model. For the additive decomposition, the decomposed time series can be written as:

$$y_t = S_t + A_t$$

(2-15)

Where $S_t$ is the seasonal component and $A_t = T_t + \varepsilon_t$ is the seasonally adjusted component.

And for the multiplicative decomposition, we can write:

$$y_t = S_t A_t$$

(2-16)

Where $A_t = T_t \varepsilon_t$. To make projection of a decomposed time series, forecast is made separately for the seasonal component and the seasonally adjusted component. It is assumed that the seasonal component will remain same in future or the rate of change is extremely slow. To forecast the seasonally adjusted component, any non-seasonal forecasting method such as linear regression or exponential smoothing may be used. The additive model is appropriate if the seasonality do not depend upon the level of the time series. On the other hand, if the seasonal fluctuations change over time, then multiplicative model should be used. Thus, the choice of the appropriate model is subject to the pattern of the time series to be forecasted.

In this thesis, time series exponential smoothing methods (HLT and HW) are used to know the RR of a software product at the pre-defined release date by analysing the patterns of
the monitored RR over time. Projection of RR allows the product manager to make proactive decisions regarding release window of a software product. Details about the application of time series projection method in the context of RR evaluation is discussed in Section 3.3.

2.2 A Glossary of Terms Related to Software Release Readiness

A glossary of terms related to software release readiness used in this thesis is provided in Table 2-1. These terms are frequently referred in various section of this thesis.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product</td>
<td>A product is a combination of goods and services, which a supplier/development organization combines in support of its commercial interests to transfer defined rights to a customer. [55]</td>
</tr>
<tr>
<td>Software product</td>
<td>Software Product Management (SPM) is “the discipline and role, which governs the software product (or solution or service) from its inception to the market/customer delivery in order to generate biggest possible value to the business” [27].</td>
</tr>
<tr>
<td>Management</td>
<td>Software release is the decision to deliver the developed product into operational environment for actual usage by its end user. [33]</td>
</tr>
<tr>
<td>Software release</td>
<td>Release readiness is an objective measure which is determined by aggregating the degree of satisfaction of a set of individual readiness attributes.</td>
</tr>
<tr>
<td>Release readiness</td>
<td>Attributes of the software development process and product by which release readiness can be defined and judged. Example includes: satisfaction of defect finding, satisfaction of feature implementation, satisfaction of test coverage, etc. [8]</td>
</tr>
<tr>
<td>attributes</td>
<td></td>
</tr>
<tr>
<td>Release readiness</td>
<td>An objective measure which is used to define the degree of satisfaction of the release readiness attributes. For example, % of LOC covered is an objective measure by which degree of satisfaction of test coverage is judged. [8]</td>
</tr>
<tr>
<td>metrics</td>
<td></td>
</tr>
<tr>
<td>Degree of satisfaction</td>
<td>Degree to which a RR attributes is satisfied based on the corresponding objective metrics. For example, if Defect find rate is 0.1 (defects per day), then degree of satisfaction of defect finding might be 30% [56]</td>
</tr>
</tbody>
</table>
Term | Definition
--- | ---
Decision strategy | It is the behaviour of the software product manager which has impact in evaluating (aggregating the satisfaction of a set of RR attributes) the overall RR of a software product. Decision strategies may range from optimistic to pessimistic. With an optimistic decision strategy, highly satisfied RR attributes are given more importance in the aggregation process. Conversely, using pessimistic decision strategy low satisfied RR attributes are given more importance in the aggregation process [57]

### 2.3 Literature Review

#### 2.3.1 Existing Release Readiness Approaches

The literature of this domain provides some techniques and approaches that aim to determine release readiness before releasing software. Those approaches can be organized into four broad categories: checklist-based approaches, multi-dimensional metrics aggregation-based approaches, testing phase-based approaches, and defect prediction model-based approaches. An overview of these approaches is provided below.

**2.3.1.1 Checklist-based Approaches**

The typical approach for evaluating RR at the end of the development is the use of checklist criteria (or questions). In this study, RR related checklist criteria were collected and reviewed from the articles found in the web and personal blogs of the software project and product managers. A classification of the available checklist criteria with respect to the development processes are shown in Table 2-2.

Table 2-2 shows that checklist criteria were proposed from development, build, test, documentation, reliability, risk analysis, and packaging perspectives. As the RR is highly associated to the testing processes, most of the reported criteria are related to the testing activities such as execution of test plan, remaining high priority bugs, documentation of bugs, etc.
Table 2-2: Release readiness checklist based on RR criteria

<table>
<thead>
<tr>
<th>Development process</th>
<th>Checklist criteria/Questions</th>
<th>Reference</th>
</tr>
</thead>
</table>
| **Implementation**  | - Are the planned features implemented?  
                      - Have system requirements been reviewed and approved by the designated approvers?  
                      - Verify all source code meets coding standard; run check style or other style-checker and do manual inspection. |
|                     | **Build**                   | [7, 58]   |
|                     | - All code must compile and build for all platforms.  
                      - Are the CIs of the entire release package identified? |
|                     | **Test**                    | [58]      |
|                     | - Zero high priority bugs.  
                      - For all open bugs, documentation in release notes with workarounds.  
                      - All planned QA tests run, at least 98% pass.  
                      - Number of open defects decreasing for last three weeks.  
                      - Feature x unit tested by developers, system tested by QA, verified with customers A, B before release.  
                      - Interoperability and serviceability tests performed. |
|                     | **Documentation**           | [7, 59]   |
|                     | - Are software requirements documented?  
                      - Are the user manuals prepared?  
                      - Have all remaining defects been documented? |
|                     | **Reliability**             | [58]      |
|                     | - Is the reliability assessment acceptable for both HW and SW? |
|                     | **Risk analysis**           | [59]      |
|                     | - Overall risk assessed and acceptable. |
|                     | **Packaging**               | [60]      |
|                     | - Creation of the releasable software media (CD-ROM, downloadable JAR file, etc.) required by customer.  
                      - Creation of install and configuration scripts as needed. |

Though the checklist criteria and questions are useful to assess RR at the end of the release cycle, software companies still need to establish their own process and objective measures to use them effectively. Moreover, most of the criteria are subjective and cannot be evaluated until the late stage of the release cycle. Therefore, using this approach it is not possible
to identify and address release related issues proactively. In addition, evaluations of the criteria are not aggregated, thus, it is still difficult for the product managers to understand the overall RR status of the software.

2.3.1.2 Multi-dimensional Metrics Aggregation-based Approaches

In this approach, a set of software process and product related metrics are defined from various dimensions and their values are normalized and aggregated into a single composite release readiness index. As this is one of the least studied areas of software engineering, only two studies are found in this direction. In [59], metrics were defined and aggregated from five dimensions (e.g. Software Functionality, Operational Quality, Known Remaining Defects, Testing Scope and Stability, Reliability) to calculate the software release readiness index. A linear aggregation model such as an additive model, multiplicative model or a hybrid model (combination of linear and multiplicative model) are suggested for aggregating the normalized metrics values. The authors claim that by setting measurable targets for each metric along a dimension, release readiness can be monitored throughout the development phases. Another work [61] of this approach proposed a release readiness metric called “ShipIt” which is defined in the interval [0, 1], where the value 1 indicates complete readiness. Metrics are collected from seven different stages (e.g. requirement analysis and design, coding, testing, quality assurance, manuals and documentation process, supervision, and support) of the development life-cycle and their weighted sum is taken to calculate the final readiness index.

Though this approach determines RR based on objective measures from various aspect of the software product, still there are some limitations. For instance, the works described above do not consider uncertainty in evaluating metrics with respect to RR. In reality it is tough to define the degree of satisfaction of the RR attributes precisely based on the current value of the
objective metrics. In principal, the degree of satisfaction is a fuzzy term which needs special consideration while evaluating or normalizing with objective measures. In both works, RR related metrics are defined in ratio (i.e. actual/planned) so that they can be easily normalized to [0, 1]. Also metrics are not defined in a goal oriented way. Normalized values of the metrics are aggregated in a simplistic manner with linear aggregation model. As one of the key applications of aggregated RR is to decide about the delivery (Go-no Go decision) of the software, it is essential to quantify how much risk is considered while combining the individual evaluation of the RR attributes. For example, the product manager is on safe side if the aggregation operator emphasizes more on the low satisfied attributes than the highly satisfied attributes. Conversely, the level of risk is high when the aggregation operator puts more emphasize on the highly satisfied attributes in determining the overall evaluation. Consideration of the level of risk in determining the overall RR also allows product managers to study the possible trade-off options between the outstanding (important and highly satisfied) attributes and poor (less important and poorly satisfied) attributes. Specifically, if time-to-market is critical for the business success of the software product. In the proposed approach, these gaps on evaluating and aggregating RR attributes are addressed with the concept of fuzzy set theory and non-linear Ordered Weighted Average (OWA) operator, respectively.

2.3.1.3 Testing Metrics-based Approach

In this approach, a set of testing (or defect tracking) related metrics (i.e. number of open defects, defect find rate, test passing rate, and defect density) are used to develop various type of RR indicators which are meant to be used by the management staffs as tool for making confident release decision. Trends of the metrics are also studied for identifying and addressing release related issues proactively. In [5], Staron et al. proposed time-to-release as a RR indicator for
agile development based on number of defects, defect removal rate, test execution rate, and test pass rate. They provided the following equation to calculate a time-to-release indicator:

\[
\text{Time to Release} = \frac{\# \text{ defects}}{\text{defect removal rate} - (\text{test execution rate} - \text{test pass rate})} \tag{2-17}
\]

In Eq. (2-17), all metrics are defined for the last four weeks of the release cycle. However, assumptions such as traceability between test cases to requirements and traceability between test cases to work packages have narrowed down its applicability in other domains. Moreover, a single value as RR indicator would not give proper insight to all the complexities involved in the decision making of software releases.

In [6], the answer to the question “Is your software ready for release?” is provided by a decision technique that specifies the number of test hours before software release in which no additional test failures are permitted to be found. The idea is that the longer the testing proceeds without finding a failure, the greater the likelihood becomes that the number of remaining failures is very small. With this method, if no failures are seen in the target test time, then the software is declared ready to release. On the other hand, if even one failure is detected in the target test time, the software will not be released and conventional testing is continued. The method is derived from exponential model of software reliability which requires three inputs: the target projected average number of customer failures, the total number of test failures detected so far, and the total test-execution hours up to the last failure. Eq. (2-18) shows the calculation of zero failure test hours with respect to these inputs. The main limitation of the model is that it mainly focuses on post release quality based on failure intensity (or reliability) as RR indicator. However, release readiness measure should reflect the state of the software before releasing with
the consideration of other aspects such as quality of the source code, test coverage, build status, amount of functionality, etc.

\[
\text{Zero Failure Test Hours} = \frac{\ln \left( \frac{\# \text{ customer problems}}{0.5 + \# \text{ customer problems}} \right)}{\ln \left( \frac{0.5 + \# \text{ customer problems}}{\# \text{ test} + \# \text{ customer problems}} \right)} \times \text{test hours to last problems} \quad (2-18)
\]

In this direction, Pearse et al. [2], identified and applied four objective metrics (code turmoil, test passing rate, defect find rate, and # of open defects) to assess the RR of an embedded system developed in HP lab. With their method, metrics are subjectively evaluated based on the experiences of the experts and mapped into a 1 to 10 point risk scale, with 1 representing the least risk and 10 the most risk. A spider chart is used to visualize and compare the metrics of the current project with past projects. According to the authors, spider chart helps the management staffs to understand the risk associated to each metrics to deliver the product on predefined release date. One of the main drawbacks of this method is that it largely depends on the past projects data and experts (for mapping risk) which are not always available in reality. In the proposed, instead of using past projects, RR related issues are identified based on the monitored RR scores of the project itself.

Overall, the works discussed above are straightforward and very easy to use. However, they are very specific to the testing phase of the release-cycle. In particular, their main focus is to provide support for making confident release decision at the late stage of the development. In contrast, this thesis focuses more on evaluating and monitoring RR from the beginning of the project so that RR issues can be identified and addressed earlier by the product manager.
2.3.1.4 Defect Prediction Model-based Approach

In this approach, number of remaining defects is used as proxy method to decide if a piece of software is ready to be released. Defect prediction models based on source code related metrics are used to predict the number of remaining defects in a system. The idea is that by comparing predicted number of defects and number of defects discovered in testing, the software manager can decide whether the software is ready to be released or not. In [62], a set of source code related metrics (i.e. depth of inheritance tree, response for a class, number of parents of a class, etc.) are collected from different layers (e.g. data access layer, presentation layer, business logic layer) and provided to the neural network based prediction model for estimating the remaining defects. With the number of remaining defects, the prediction model also provide the maintenance cost which includes the amount of time required to make the changes in order to correct the defects remaining in the software. In this direction, Wild and Brune [63] predicted the number of remaining defects based on the degree of code change made during the last 4-6 weeks. The authors argue that for reliable release decision it is essential to consider the recent changes made in source code level because there is a linear relationship between change and remaining defects.

However, the methods discussed above do not support RR evaluation properly. The main focus of RR evaluation is to know the overall status of the project at a particular point of time in the release cycle in a quantitative way. In this regard, knowledge about the number of remaining defect can be used only for evaluating the satisfaction of RR attributes related to defect finding. The overall status of the project may depend on other attributes such as satisfaction of bug fixing, satisfaction of test coverage, satisfaction of codebase stabilization, satisfaction of feature implementation, satisfaction of test pass rate, etc. Therefore, with this approach, it is only
possible to evaluate RR partially. The proposed approach of this thesis considers evaluating and monitoring RR considering the all important aspects (i.e. implementation of functionality, source code quality, build status, testing status, and documentation) of a release.

2.3.2 State of Validation of RR Metrics and Approaches

In this section, discussion is made about the empirical validation of the RR approaches and metrics reported in the related works. A summary of the studies including research method, system under study, purpose of the study, and key findings are presented in Table 2-3. As we can see from the summary, the main purpose of the study is to validate the RR metrics with real life software projects.

Table 2-3: Summary of the empirical evaluation of the RR metrics and approaches

<table>
<thead>
<tr>
<th>Ref.</th>
<th>Research method</th>
<th>System under study</th>
<th>Purpose of the study</th>
<th>Key findings</th>
</tr>
</thead>
</table>
| [28] | Case study      | DataFinder (a RDBMS product) | Reviewing the actions and measurements taken for the assessment of the readiness of the product ship | – Open and close bug rates, percentage of test passing, and test coverage metrics are effective in RR assessment.  
– It is essential to define RR strategy at the early stage of the development life-cycle. |
| [62] | Case study      | Warehouse management applications and networked information system | To verify the accuracy of the defect prediction model and to quantify the relative importance of the source code related metrics | – Defect prediction can be used as proxy method of RR indicator.  
– Source code related metrics are effective in defect prediction. |
| [4]  | Case study      | Electronic design automation products | To validate the metrics used for determining RR of the software product. | – Quality metrics such as defect find rate and severity 1 open defects are effective in providing early indications of RR related problems.  
– As soon as possible, tracking of the RR metrics should be started. |
<table>
<thead>
<tr>
<th>Ref.</th>
<th>Research method</th>
<th>System under study</th>
<th>Purpose of the study</th>
<th>Key findings</th>
</tr>
</thead>
</table>
| [6]  | Case study        | 75 product release in Motorola | To validate the zero-failure model for achieving RR               | − Out of 75 product release only 12 products have the post release failure history.  
− The technique is effective in monitoring testing related problems.                                                                                                                                       |
| [2]  | Case study        | HP LaserJet product      | To validate the RR metrics and approach                          | − Metrics such as defect find rate, test passing rate, open defects, code turmoil are effective for understanding RR.  
− Comparison of metrics states with past project’s metrics states are effective.                                                                                                                             |
− In determining time-to-release open defects, defect removal rate, deviation between test pass rate and test execution rate are effective.                                                                                                           |

In [28], a case study was reported where actions and measurements taken for assessing the readiness of the product ship were reviewed for a RDBMS enhancement product called DataFinder. The management team of the product defined and measured open and close bug rates, percentage of test passing, and test coverage to know the current readiness state of the product. Beside these defects related metrics they also tracked the metrics such as task productivity rates, and feature productivity rates to understand the state of the development activities. By analyzing the trends of these metrics, management was able to predict exactly how long system tests would take, and when the product ship date would occur. The case study also revealed that it is required to decide about the important RR metrics at the beginning of the project when release date is most important to management.
Another case study is presented in [4], which revealed that the quality metrics such as *defect find rate, severity 1 open defects, and weighted defect density* can be used effectively to indicate problems related to release and help managers to make changes to the plan for completion of the release. These metrics were validated with the data from the electronic design automation related products developed by Mentor Graphics. Monitoring the trends of these metrics, the management was able to identify that the next release of the product was having difficulties two months before the completion date. Based on that objective measures, management decided to delay the release until the target quality requirements are met. The key lessons learned from the study includes: i) RR metrics need to be defined clearly and quantitatively at the planning phase of the release, ii) metrics need to be tracked as soon as possible so that trends can be observed as early as possible, and iii) trends of the metrics are helpful for convincing the senior corporate management to revise the schedule or quality of the product.

Evaluation of the release decision technique stated in Eq. (2-18) was evaluated with 75 product releases in Motorola [6]. As described in Section 2.3.1.3, the technique provides the failure free testing hours required before releasing software based on target projected number of customer failures, total number of test failures detected so far, and test execution hours up to the last failures. The model was derived from the exponential model of reliability [64]. Out of the 75 product releases, 12 product releases had enough post release failure history. According to the author, the method can be used not only as proactive device for making release decision but also as reactive method for monitoring testing activities. Overall the results of the validation indicated that the estimation of failure free test hours is an effective indicator of release readiness of software products.
A case study of using metrics to manage the end game of a software project is reported in [2]. The purpose of the study was to show the electiveness of metrics such as number of open defects, test passing rate, defect find rate, and code turmoil in understanding the status of the software and providing insights about potential risk of release. A corporate history of these metrics were developed to support comparison of the current project with similar past projects. The study was performed with an HP LaserJet product for which meeting the firmware release date was as important as meeting the quality requirements. The risks associated with the selected metrics were monitored and compared with three similar past projects on the same week before the release date. Using this approach, potential risk associated with current product release was identified and necessary action was taken to limit those risks. For instance, the value of the code turmoil (i.e. amount of addition/deletion in the code base) metric of the project was high compared to the past projects, indicating that a large amount of code was being changed in the code base by the development team. To limit the risk associated with this metric, management communicated the concerns about the rate of code change to the engineering community. In addition, they monitored the code check-in process and imposed authorization for pre-check-in. Necessary action was also taken to manage the risk associated to the other metrics. Key findings of the case study include: i) Testing and defect tracking related metrics were effective for understanding risk associated with releasing software products, ii) creating database of past project metrics was critical to support product release management, iii) looking at the metrics together in a graph is helpful to understand the big-picture of the project, iv) it is required to track the RR metrics through entire project, not just at the end.

Validity of the testing and defect tracking metrics is also studied with products developed using agile and lean methodologies. An action research is reported in [5] where metrics such as
number of defects, defect removal rate, test execution rate and test pass rate are used to calculate the amount of time (number of weeks) needed in achieving release readiness as stated in Eq. (2-17). The time-to-release RR indicator was used in a project at one of the units of Ericsson AB in one of its largest projects. The management of the project found it as a trustworthy Key Performance Indicator (KPI) in observing organization’s capability to continuously deliver the customer value. However, the RR indicator is very specific to agile methodology and validated at a single company. Therefore, other organizations need to be careful before using this method.

As discussed in Section 2.3.1.4, a defect prediction model based on source code related metrics is also proposed as a proxy method to predict release readiness. Evaluation of the source code related metrics in predicting defects is reported in [62]. Evaluation of the model was performed with the source code metrics collected from Warehouse management applications and networked information system. The results of the study revealed that depth of inheritance tree, response for a class, and numbers of parents have relatively more importance in defect prediction model.

Overall, the studies revealed that it is important to start monitoring RR as early as possible, and testing and defect tracking related metrics are effective in monitoring RR. The usage of the key RR metrics in industry for identifying and managing release related issues were also reported. However, little evidence is found about metrics related to implementation of functionality and source code quality. Moreover, the impact of the individual RR attributes on overall RR and empirical validation of the occurrence frequency and patterns of the attributes which limit the RR are not studied yet. Thus, more empirical validation of the RR attributes and metrics is required with large set of real world projects.
2.3.3 State of Tool Support for Evaluating Release Readiness

Tool support allows product managers to truly evaluate the approach and see if it is useful for them. As this is one of the least studied areas of software engineering, very little tool support is provided from the research community. In fact, none of the RR approach described in Section 2.3.1 provided a clear link to access a demo of the tool. Only one paper describes about a visualization based tool which is designed to support continuous RR monitoring. After searching in the web, only two software organizations are found that integrate a RR analysis component as supplementary feature of their main product. A summary of the capabilities of those tools are presented in Table 2-4.

Boarland introduces TeamInspector [65] that extracts metrics related to code analysis, test coverage, standard compliance, and build trends to verify software is ready to release. In this context, MKS developed PTC integrity [25] which extracts and visualizes metrics related to functionality, standard compliance, and budget and schedule to verify RR. As shown in Table 2-4, both TeamInspector and PTC integrity support automatic data collection from the integrated Software Configuration and Change Management (SCCM) tools. In TeamInspector, the required data is collected from tools such as Perforce®, Subversion® and Borland StarTeam®. PTC integrity collects data from the tools specific to the MKS products. The other tool does not integrate existing project management tools; therefore, automatic data collection is not supported. These commercial tools also provide dashboards for monitoring RR metrics across the product portfolio. The dashboards of these tools include interactive charts (e.g. bar chart, pie chart, etc.) and the product manager can drill-down to the artefacts related to the metrics to identify the root cause of the problem. The metrics are also customizable. However, the main limitation of these tools is that they do not evaluate and aggregate satisfaction of the monitored
RR metrics. Moreover, they do not provide effective ground for exploring release readiness issues. For instance, the projection of RR into the future is not supported by any of these tools. Thus, product managers still have to face difficulty in determining the state of the product readiness.

Table 2-4: Summary of the RR monitoring tools

<table>
<thead>
<tr>
<th>Attributes</th>
<th>TeamIspector [65]</th>
<th>PTC integrity[25]</th>
<th>SRI[59]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic data collection</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Data sources</td>
<td>Only Borland products</td>
<td>Only PTC product</td>
<td>-</td>
</tr>
<tr>
<td>Customization of RR attributes/metrics</td>
<td>Yes</td>
<td>Yes</td>
<td>-</td>
</tr>
<tr>
<td>Dashboard for monitoring RR</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Drilldown capability</td>
<td>Yes</td>
<td>Yes</td>
<td>Limited to RR factors</td>
</tr>
<tr>
<td>Aggregation of RR attributes/metrics</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Exploration of RR issues</td>
<td>Limited</td>
<td>Limited</td>
<td>Limited</td>
</tr>
<tr>
<td>Projection of RR</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

2.3.4 Release Readiness Dimensions, Attributes and Metrics

RR attributes and corresponding metrics were extracted and organized into four important RR dimensions by reviewing the existing literature and articles on RR domain. Identified RR dimensions are: implementation status, testing scope and status, source code quality, and documentation scope and status. As shown in Figure 2-4, most of the studies (14 out of 16) reported RR attributes and metrics related to testing scope and status followed by implementation status (9 out of 16). Comparatively less studies reported RR attributes and
metrics related to source code quality (5 out of 16) and documentation scope and status (4 out of 16). Detail classification of the RR attributes and metrics are presented in Table 2-5. This classification of RR attributes can be used as guideline for selecting the RR attributes for any kind of software products. In the proposed RR evaluation approach, this classification scheme is referred as a baseline for the product managers.

![Distribution of the studies according to the important RR dimensions](image)

**Figure 2-4: Distribution of the studies according to the important RR dimensions**

**Table 2-5: Classification of RR attributes and metrics found in the existing literature**

<table>
<thead>
<tr>
<th>RR Dimensions</th>
<th>RR Attributes</th>
<th>Metric definition</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation status</td>
<td>Implementation accuracy</td>
<td>– Feature implementation ratio</td>
<td>[2, 5, 25, 55, 59, 61, 63, 65]</td>
</tr>
<tr>
<td></td>
<td>Change request implementation</td>
<td>– % of change request implemented</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Deviation of implementation effort</td>
<td>– Effort deviation ratio</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Continuous integration and build status</td>
<td>– % of successful builds</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Source code stability</td>
<td>– Code churn/code turmoil</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bug fixing</td>
<td>– Bug fix rate</td>
<td></td>
</tr>
<tr>
<td>Testing scope and status</td>
<td>Test progress</td>
<td>– Test execution rate</td>
<td>[1, 2, 4, 5, 7, 10]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>– Test pass rate</td>
<td></td>
</tr>
</tbody>
</table>
## 2.3.5 Discussion

The readiness approaches described in Section 2.3.1 shows that only nine studies were from academia, indicating that this is one of the least studied areas of software engineering. None of the studies provide a unified method that can be used for continuous monitoring of RR from the early development phase of the software product. Only two studies addressed continuous release readiness evaluation with the consideration of readiness attributes from multiple dimensions and aggregated the normalized metrics values to determine the overall RR index. However, normalization and aggregation of the metrics in a simplistic manner has narrowed down their applicability in evaluating RR. Hence, more comprehensive RR attribute

<table>
<thead>
<tr>
<th>Source code quality</th>
<th>Obtained test coverage</th>
<th>Overall defect finding and severity of the defects</th>
<th>Defect found for a specific activity</th>
<th>Functionality test coverage</th>
<th>Structure of the source code</th>
<th>Code inspection/review</th>
<th>Refactoring status</th>
<th>Code complexity</th>
<th>Documentation scope and status</th>
<th>Technical documentation status</th>
<th>User manual status</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>% LOC covered by unit test</td>
<td>Defect find rate</td>
<td># of defects found by code review</td>
<td>% of feature tested</td>
<td>Lack of cohesion in methods</td>
<td>% method reviewed</td>
<td># of code smells per class</td>
<td>Average method complexity</td>
<td>Not Found</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>% method covered</td>
<td></td>
<td></td>
<td></td>
<td>Number of parents of a class</td>
<td></td>
<td></td>
<td>Average class complexity</td>
<td></td>
<td>Not Found</td>
<td></td>
</tr>
<tr>
<td></td>
<td>% block covered</td>
<td></td>
<td></td>
<td></td>
<td>Depth of inheritance tree</td>
<td></td>
<td></td>
<td>Cyclomatic complexity</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

25, 28, 55, 59, 61, 65-67, 66, 68, 69, 70
evaluation and aggregation techniques are needed in the RR evaluation approaches to increase their applicability into product management.

Studies of the other approaches (i.e. checklist and testing metrics based approach) focused on release readiness evaluation at the late stage of the release cycle relying on few isolated testing metrics. So, the management staffs have less scope to address the release related issues proactively. In addition, RR metrics are limited to defect tracking and test execution processes. Therefore, a more comprehensive RR approach is required with the consideration of all important aspects (i.e. functionality, source code quality, testing, documentation, packaging, etc.) of the software product. In addition, more investigation is needed from the researchers and practitioners of this domain to identify and validate the important RR attributes and metrics related to source code quality and implementation of functionality dimensions.

One of the main purpose of the RR evaluation is to enable managers to make many kind of prediction, assessments, and trade-offs about releasing the software product. However, none of the studies of the classified approaches reported a method for studying projection of RR based on the monitored RR scores so that the product manager can make proactive decisions regarding RR improvement or release window of the product. Methods for monitoring and identifying occurrence frequency and patterns of the bottleneck factors based on the past behaviour of the individual RR attributes are addressed by the related works. Therefore, it is essential to explore these areas with empirical studies to provide guideline to the product managers.

The results also indicate that there is very limited tool support. In fact no research paper provides a link to access a demo of the tool. Only two commercial tools are available. Having tool support would allow users to evaluate the method and see if it is applicable for them. Specifically, providing tool support for RR evaluation with the integration of the existing
development and project management tools (i.e. Github\textsuperscript{1}, Jira\textsuperscript{2}, and Team Foundation Server\textsuperscript{3}) is expected to the product managers.

In this thesis, an analytical approach for supporting continuous RR evaluation is described, addressing most of the gaps and limitations discussed above. Tool support and evaluation of the proposed approach is also presented through two comprehensive case studies.

2.4 Summary

In this chapter, background topics and related works in the area of software release readiness are discussed. The literature review revealed that this is one of the least studied (only nine studies from academia) areas of software engineering and there is no uniform method which can be applied for evaluating release readiness. In most of the approaches, RR is studied at the late stage of the release cycle relying on few isolated testing related metrics. Only two studies normalized the RR related metrics and aggregated those in a simplistic (linear way) manner to provide composite software readiness index. None of the approaches studied projection of RR at the release date, which is important for making proactive decisions regarding software releases. Analysis on empirical validation of the RR strategies revealed that number of defects, test passing rate, defect find rate, and code turmoil are effective metrics for evaluating RR. However, little is known about the impact of these metrics on overall RR through the release cycle. In addition, validation of the implementation and source code related metrics in RR evaluation is not found. The related works also indicate that there is very limited tool support. Only two commercial tools are available which extract and visualize metrics related to functionality, code

\textsuperscript{1} https://github.com/
\textsuperscript{2} https://www.atlassian.com/software/jira
\textsuperscript{3} http://msdn.microsoft.com/en-ca/vstudio/ff637362.aspx
analysis, test coverage, standard compliance, and budget and schedule to verify that software is ready to release.

Overall, the existing approaches and corresponding methods do not enable managers to gain insight about release related problems based on evaluating and monitoring RR through the release cycle. In the next chapter, to mitigate these limitations an analytical approach is described which allows evaluating RR at any point of time during the release cycle, making projection of RR at the release date, and gaining insights about the limiting factors in achieving RR.
Chapter Three: Proposed Approach

In this chapter, an analytical approach called RELREA is proposed that allows evaluating RR at any point of time during the release cycle. The fundamental concepts used in the proposed approach are discussed at the beginning of the chapter. The workflow of the approach and an illustrative example are also presented in this chapter.

3.1 Preliminaries

Fundamental concepts required to develop an analytical approach for evaluating, monitoring, and projecting RR are introduced in the following sub-sections.

3.1.1 Software Release Readiness Attributes

RR attributes are the aspects of software product by which RR can be defined and judged. They can be defined uniformly by considering both organization goals and customer expectation of a software product. As described in Section 2.3.4, RR attributes and metrics available in literature are identified and categorized into four important RR dimensions: implementation status, testing scope and status, source code quality, and documentation status. An overview of the attributes of these dimensions is provided in Table 3-1. As Table 3-1 shows, they cover both important artefacts (e.g., features, source code, documents, etc.) and development activities (e.g., coding, building, testing, etc.) of a software project.
### Table 3-1: RR dimensions and their related RR attributes

<table>
<thead>
<tr>
<th>Readiness dimensions</th>
<th>Overview of related Release Readiness Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation status</td>
<td>Attributes related to feature implementation, change request implementation, coding effort, continuous integration, build trends, etc.</td>
</tr>
<tr>
<td>Testing scope and status</td>
<td>Attributes related to defect finding, defect fixing, test coverage, test effort, etc.</td>
</tr>
<tr>
<td>Source code quality</td>
<td>Attributes related code review, coding style, code smells, refactoring, code complexity, etc.</td>
</tr>
<tr>
<td>Documentation scope and status</td>
<td>Attributes related to user manual, design documents, test specification, test case documentation, etc.</td>
</tr>
</tbody>
</table>

#### 3.1.2 Goal Question Metric Paradigm

The Goal-Question-Metric (GQM) paradigm [21] is the de-facto standard to perform software measurement. In its essence, it guides the process of designing an effective measurement program and finding insights from the data collected. In the context of release readiness analysis, our goal is to evaluate the status of the overall readiness. The RR attributes corresponding to questions were refining the goal. Available data associated with every question are used to answer them in a quantitative way. An example of defining objective metrics for the testing dimension is shown in Table 3-2.

### Table 3-2: Example of GQM paradigm for defining RR attributes and metrics [8]

<table>
<thead>
<tr>
<th>Goal</th>
<th>RR Attribute</th>
<th>Question (related to readiness attribute)</th>
<th>Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessing satisfaction of testing</td>
<td>Satisfaction of defect finding</td>
<td>Is the testing activity reducing the defects?</td>
<td>Defect find rate</td>
</tr>
<tr>
<td></td>
<td>Satisfaction of unit testing</td>
<td>To what extent source code are covered by unit test?</td>
<td>% Line of Code (LOC) covered</td>
</tr>
</tbody>
</table>
3.1.3 Relative Importance of the RR Attributes

Relative importance or weights of the RR attributes are very much specific to the context of the software project. Therefore, the product manager should analyze the definition of success of the product before defining them. She can either directly specify these weights or a normal pairwise comparison method can be used. In [71], Saaty proposed the Analytical Hierarchy Process (AHP) that provides a systematic method of determining relative weights based on pairwise comparison among the attributes. Due to the dynamic and complex nature of software development, the relative importance of the RR attribute may change throughout the release cycle. For instance, at the early stage of the release cycle relative importance of the RR attributes related to feature implementation can be higher compared to the RR attributes related to testing (quality) of the features. However, if time-to-market (for competitive business world) is crucial for the business success of the product, then before the scheduled release date RR attributes related to testing activities would get more importance over the implementation related RR attributes.

3.1.4 Evaluation of RR Attributes with the Concept of Fuzzy Set Theory

The concept of fuzzy set theory and the membership function described in Section 2.1.6, can be used to evaluate the degree of satisfaction of the RR attributes. For example, suppose we want to evaluate the satisfaction of defect finding attribute. In this regard, defect arrival rate (DAR) (defects/day) is an objective metric which can be used to evaluate the attribute. Let, \( X = R^+ = \{ \text{positive real number that can be represented as value of DFR} \} \) and \( A = \{ \text{satisfaction of defect finding} \} \) be a fuzzy set representing the attribute.

As shown in Figure 3-1, its membership function is defined as \( \mu_A(x) \) which means DFR less than 1 defects/day is considered as full satisfaction of defect finding, while defect arrival rate
more than 10 defects/day is considered the attribute is not satisfied at all. Values of $DFR$ between, 1 and 10 are considered satisfaction to some degree.

According to this definition, if the value of $DFR$ is 5 defects/day, then the “degree of membership” or “degree of satisfaction of defect finding” will be 0.56. The definition of this membership function is context specific because different projects may have different perceptions about the satisfaction of defect finding. Thereby, degree of satisfaction of the readiness attributes can be determined by defining the membership functions.

$$
\mu_{A}(x) = \begin{cases} 
1 & x < 1 \\
\frac{10 - x}{10 - 1} & 1 \leq x \leq 10 \\
0 & x > 10 
\end{cases}
$$

(Figure 3-1: Membership function of the fuzzy set $A = \{\text{satisfaction of defect finding}\}$)

3.1.4.1 Elicitation of Membership Function

Defining membership function is one of the fundamental issues associated with the application of fuzzy set theory. Success of a solution approach largely depends on the proper definition of membership functions. In literature, different types of membership function elicitation techniques are proposed. According to [72, 73], those techniques can be organized into three board categories: perception-based, clustering-based, and heuristic-based. In perception-based technique, a group of people are asked to provide their rating about the concept (i.e. tall person, short person, etc.) represented by the fuzzy set. The answers are polled and an average is taken to construct the membership function. This technique is also known as natural
way of constructing membership function. The main limitation of this technique is that it requires substantial amount of effort and human involvement to construct the membership function. In the case of fuzzy clustering techniques, membership functions are constructed from a given set of data. Sophisticated clustering algorithms are applied to organize the available data into different sub-groups. By analyzing the overlapping among the sub-groups a proper shape is constructed to define the membership function. This technique can be applied only when adequate amount of data is already collected.

In the application of fuzzy set theory, the most frequently applied technique is the heuristic-based one. In this technique, predefined shapes for membership functions are used. Two broadly used categories of heuristic-based membership function are piecewise linear functions and piecewise monotonic functions [72, 73]. By selecting an appropriate shape and providing its specific parameters, membership functions can be elicited easily for a particular fuzzy set. Considering the complexity of the first two techniques, in the proposed approach piecewise linear functions and piecewise monotonic functions are suggested for characterising the degree of satisfaction of the RR attributes. Example of the commonly used shapes for these two categories of functions include: increasing functions, decreasing functions, triangular functions, trapezoidal functions, and smoother functions such as symmetric Gaussian functions and sigmoidal functions. Mathematical and graphical representations of these functions are shown in Table 3-3. From these shapes the product manager can easily identify an appropriate one for a RR attribute. Application of these shapes in the context of RR evaluation is discussed in Section 3.4.3.
Table 3-3: Commonly used shapes for the piecewise linear and piecewise monotonic membership functions

<table>
<thead>
<tr>
<th>Shape Type</th>
<th>Shape Name</th>
<th>Shape</th>
<th>Definition</th>
</tr>
</thead>
</table>
| Piecewise linear functions | Deceasing  | ![Graph](image1) | \[\mu_A(x) = \begin{cases} 
1 & x < p \\
\frac{q-x}{q-p} & p \leq x \leq q \\
0 & x > q 
\end{cases} \] |
|                         | Increasing | ![Graph](image2) | \[\mu_A(x) = \begin{cases} 
0 & x \leq 0 \\
\frac{x}{p} & 0 \leq x \leq p \\
1 & x > p 
\end{cases} \] |
|                         | Triangular | ![Graph](image3) | \[\mu_A(x) = \begin{cases} 
0 & x < p \\
\frac{x-p}{m-p} & p \leq x \leq m \\
\frac{q-x}{q-m} & m \leq x \leq q \\
0 & x > q 
\end{cases} \] |
|                         | Trapezoidal | ![Graph](image4) | \[\mu_A(x) = \begin{cases} 
0 & x < p \text{ or } x > r \\
\frac{x-p}{q-p} & p \leq x \leq q \\
1 & q \leq x \leq r \\
\frac{s-x}{s-r} & r \leq x \leq s 
\end{cases} \] |
| Piecewise monotonic functions | Gaussian  | ![Graph](image5) | \[\mu_A(x) = e^{-\frac{(x-p)^2}{2\sigma^2}}\] |
|                         | Sigmoidal  | ![Graph](image6) | \[\mu_A(x) = \frac{1}{1 + e^{-a(x-p)}}\] |
3.1.5 Aggregation of Values of Membership Function

As discussed in section 3.1.4 and 3.1.4.1, values of the membership functions represent the degree of satisfaction of the RR attributes. We need to aggregate the individual evaluation of the readiness attributes to obtain an overall evaluation. In the literature, many aggregation operators are proposed. An overview of the properties of those operators can be found in [52]. Selection of the aggregation operator depends on the context of its use. In the context of evaluating overall RR, we need an average type aggregation operator which should have the capability of including the following special considerations along with the general mathematical properties (i.e. boundary conditions, monotonicity, continuity, associativity, symmetry, neutral element, idempotence, etc.).

- The first consideration is the relative importance (or weights) of the readiness attributes. This allows the product manager to emphasize important readiness attributes, so that its influence on the overall readiness will be higher.

- The second consideration is the desired decision strategy of the product manager. For example, a pessimistic product manager desires that most of the attributes be satisfied. Conversely, an optimistic product manager may become satisfied if some of the attributes are satisfied.

In this research, linguistic quantifier guided Ordered Weighted Average (OWA) [20] is applied as an aggregation operator because of its capability of incorporating the above special considerations. In the domain of Multi-Criteria Decision Making (MCDM) OWA operators have been applied successfully during the last decades [74]. They are also applied in evaluating various aspects (e.g. risk, cost, etc.) of software engineering [75-79]. A brief overview of the
linguistic quantifier based OWA operator and its applicability in RR evaluation is discussed below.

3.1.5.1 OWA Operators Based on Linguistic Quantifier Guided Decision Strategies

One of the main challenges of applying OWA operators is determining the appropriate order weights according to the decision maker’s behavior. In [57] Yager shows that the behavior of the decision maker can be addressed with the concept of linguistic (or fuzzy) quantifiers proposed by Zadeh [37]. A relative linguistic quantifier \( Q \), such as \textit{most}, \textit{many}, \textit{half}, \textit{some}, and \textit{few} can be used to define a particular decision strategy. The linguistic quantifier \( Q \) is represented as a fuzzy subset of the unit interval, where for a given proportion of the attributes \( p \in [0,1] \) of the total attributes, \( Q(p) \) indicates the extent to which the proportion \( p \) satisfies the semantics defined in \( Q \). For example, given \( Q = \text{“most”} \), if \( Q(0.7) = 1 \), then we can say that satisfaction of 70% of the total attributes satisfies the idea conveyed by the quantifier “most”, whereas \( Q(0.55) = 0.25 \) indicates that the proportion 55% is barely compatible (only 0.25) with this concept. Regular Increasing Monotone (RIM) quantifiers are especially interesting for their use in OWA operators. Properties of these quantifiers are:

i. \( Q(0) = 0 \)

ii. \( Q(1) = 1 \)

iii. If \( p_1 > p_2 \) then \( Q(p_1) \geq Q(p_2) \)

In order to define the linguistic quantifier, the simplest and most commonly used is the exponential function which is defined in Eq. (3-1).

\[
Q(p) = p^\alpha, \alpha > 0 \text{ and } p \in [0,1]
\]

(3-1)

The parameter \( \alpha \) is associated with a set of decision strategies. By changing the parameter \( \alpha \), one can generate different types of RIM quantifiers and associated order weights
between two extreme case of the “At least one” and “All” quantifiers. A family of RIM quantifiers and corresponding $\alpha$ parameters, OWA weights, and their degree of optimism are presented in Table 3-4. In [57], Yager suggested the following way to compute order weights $w_i$ with the use of a RIM quantifier $Q$:

$$w_i = Q \left( \frac{i}{n} \right) - Q \left( \frac{i-1}{n} \right), i = 1, \ldots, n \tag{3-2}$$

Where $n$ the number of attributes to be aggregated and $Q(p)$ is evaluated according to Eq. (3-1). As shown in Table 3-4, by applying Eq. (3-2) the OWA weights can be calculated for a specific decision strategy.

**Table 3-4: Selected linguistic quantifiers and corresponding $\alpha$ parameters**

<table>
<thead>
<tr>
<th>Decision strategies</th>
<th>$\alpha$</th>
<th>Quantifier (Q)</th>
<th>OWA weights ($w_i$)</th>
<th>Optimism /Orness($W$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Extremely optimistic</td>
<td>0</td>
<td>At least one</td>
<td>$w_1 = 1; w_i = 0$ for others</td>
<td>1</td>
</tr>
<tr>
<td>Very optimistic</td>
<td>0.1</td>
<td>A Few</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>Optimistic</td>
<td>0.5</td>
<td>Few</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Moderately Optimistic</td>
<td>0.3</td>
<td>Some</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Neutral</td>
<td>1</td>
<td>Half</td>
<td>$1/n$</td>
<td>0.5</td>
</tr>
<tr>
<td>Moderately pessimistic</td>
<td>2</td>
<td>Many</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Pessimistic</td>
<td>5</td>
<td>Most</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Very pessimistic</td>
<td>10</td>
<td>Almost all</td>
<td>*</td>
<td>*</td>
</tr>
<tr>
<td>Extremely pessimistic</td>
<td>$\infty$</td>
<td>All</td>
<td>$w_n = 1; w_i = 0$ for others</td>
<td>0</td>
</tr>
</tbody>
</table>

* these values are subject to the number of attributes.

3.1.5.2 Inclusion of Attributes Weights in Order Weights of the OWA Operator

As mentioned earlier, aggregation function should consider the relative importance or weights of the RR attributes. In the context of OWA operators, relative importance of the attributes needs to be considered in determining the order weights ($w_i$). For the linguistic quantifier based approach, Yager provided the following Eq. (3-3) to calculate the order weights ($w_i$) regarding the attributes weight vector $v_k$. 

54
In Eq. (3-3), attributes weights \( v_k \) is ordered in descending order of the attributes scores. Employing the RIM quantifiers in Eq. (3-3), Eq. (3-4) can be obtained which can be used directly to determine the order weights \( (w_i) \) based on the relative weights \( (v_k) \) and the decision strategy indicated by \( \alpha \).

\[
w_i = Q \left( \frac{\sum_{k=1}^{i} v_k}{\sum_{k=1}^{n} v_k} \right) - Q \left( \frac{\sum_{k=1}^{i-1} v_k}{\sum_{k=1}^{n} v_k} \right)
\]

(3-3)

Using Eq. (3-4), the order weight \( (w_i) \) can directly be calculated for any decision strategy.

### 3.1.6 Formulation of Release Readiness

Based on the above discussion, the RR of a software product is formulated as an integrated objective measure as follows,

i. for a given project \( P \), \( A = \{a_1, a_2, \ldots, a_n\} \) is a given set of RR attributes;

ii. at a given point of time \( t \), \( \mu_{a_i}(x_t) \) are the values of the membership functions (MF) of \( a_1, a_2, \ldots, a_n \) with \( \mu_{a_i}(x_t) \in [0,1] \) describing the degree of satisfaction of the stated attributes;

iii. \( u_1, u_2, \ldots, u_n \) are the attributes weights satisfying \( u_i \in [0,1], \sum_{i=1}^{n} u_i = 1 \);

iv. \( \alpha \) is the preferred decision strategy of the product manager;

v. \( w_1, w_2, \ldots, w_n \) are the ordered attributes weights satisfying \( w_i \in [0,1], \sum_{i=1}^{n} w_i = 1 \), ordered weights are determined with the consideration of attribute weights \( u_i \) and decision strategy \( \alpha \);

vi. \( b_i \) is the \( i^{th} \) largest element in the collection \( \mu_{a_1}(x_t), \mu_{a_2}(x_t), \ldots, \mu_{a_n}(x_t) \).
vii. then, the RR of the given project $P$ at time $t$ is defined with OWA aggregation operator as follows:

$$R(t) = \text{OWA}_{i=1...n} \left( \mu_{a_i}(x_t) \right) = \sum_{i=1}^{n} w_i \cdot b_i$$  \hspace{1cm} (3-5)

### 3.2 Workflow of the Proposed Release Readiness (RELREA) Approach

The above formulation of RR is called Release Readiness (RELREA) approach which is one of the main contributions of this thesis. The workflow of the proposed RELREA approach is illustrated in Figure 3-2. There are 8 main steps:

1. **Define Readiness attributes, metrics and their weights:** At the beginning of the project, based on the GQM [21] approach discussed in Section 3.1.2, the product manager will define the context specific readiness attributes and their corresponding objective metrics. By analyzing the readiness attributes, she will also define their relative weights.

2. **Define membership function:** Based on the proposed membership function elicitation techniques described in Section 3.1.4.1, the product manager will define the membership functions by selecting appropriate shapes and their parameters for the readiness attributes.

3. **Data collection:** At any point of time $t$ of the current release cycle, data related to pre-defined metrics will be collected (by automated tools) and values of the objective metrics defined in step 1 will be calculated.

4. **Computation of attributes satisfaction levels:** At this point, satisfaction levels of the readiness attributes will be determined by computing the degree of membership for the
values of the objective metrics at a specific point in time $t$.

5. **Define aggregation strategy:** At the same time, the product manager can specify the desired strategy (e.g. optimistic, neutral, pessimistic, etc.) for aggregating the degree of satisfaction of the RR attributes.

**Figure 3-2: Workflow of the proposed RELREA approach [8]**
6. **Computation of overall readiness:** The overall RR is determined by aggregating the satisfaction levels of the attributes based on Eq. (3-5).

7. **Compute projected readiness:** In this step, the projected RR at the given release date is made. Details about this step are discussed in Section 3.3.

8. **Analyze RR and gain insights about the limiting factors:** This step allows the product manager to further analyze current and projected RR of the product. She can also perform trend analysis of the monitored RR and drill down to the individual attribute satisfaction scores to gain insights about the limiting factors. Based on the insights about the current limiting factors she can adjust the project parameters (e.g. testing effort, coding effort, system capacity, etc.) to achieve the requested RR of the release.

### 3.3 Projection of Release Readiness

According to Eq. (3-5) overall RR scores can be calculated over a period of time that will eventually form the RR time series as follows,

\[
RR(t_0), RR(t_1), RR(t_2), \ldots \ldots \ldots RR(t)
\]  

In the context of RR analysis, measurement of \( RR(t) \) may be taken every day, week or any other irregular interval. Based on the previously evaluated RR time series, projected readiness \( PR(T) \) is studied for the futures time \( T = t + h \) where \( h \) is the step size of the projection. For instance, if RR is measured in weekly basis and the product manager is interested to know the projected RR score after three weeks, then the value of \( h \) will be 3. In this thesis, the univariate time series projection methods are used for the following two reasons.

- As RR is an aggregated value of the degree of satisfaction of several development processes, it will not show pure randomness over time. It will follow some rules and
patterns (e.g. trend-cycle pattern, seasonal patterns, etc.) along with some white noise. With the univariate projection methods future values of the series can be obtained by considering both trends and seasonal patterns.

- We assume that recent development activities (e.g. new feature implemented, bug fixed, new test case is written, etc.) have more influence on the current and future RR state of the software product. Therefore, significant linear correlation (autocorrelation) should exist in the RR time series. For such a time series, univariate projection methods allow to emphasize more on the recent evaluation points in determining the future values of the series.

In this thesis [80], Holt’s Linear Trend (HLT) method and Holt-Winter (HW) methods are used for studying $PR(T)$. These projection methods are successfully applied in economic time series analysis [81, 82]. Applicability and accuracy of these projection methods are shown in Chapter 5 through the RR time series obtained from the case study project. An overview of these projection methods is provided in Section 2.1.8.

### 3.3.1 Projection of RR with the Consideration of Trends

When significant trends are identified in the RR time series, HLT projection method can be applied to calculate $PR(T)$. This method involves a projection equation and two smoothing equations—one for level and one for trends. The equations are stated as follows,

**Projection equation:**

$$PR(T) = RR(t + h) = l_t + hb_t$$ \hspace{1cm} (3-7)

**Level equation:**

$$l_t = \alpha * RR(t) + (1 - \alpha)(l_{t-1} + b_{t-1})$$ \hspace{1cm} (3-8)

**Trend equation:**

$$b_t = \beta * RR(t)(l_t - l_{t-1}) + (1 - \beta)b_{t-1}$$ \hspace{1cm} (3-9)
where,

i. $l_t$ indicates an estimation of the level of the RR time series,

ii. $b_t$ indicates an estimation of the trend of the RR time series,

iii. $\alpha$ is the smoothing parameter for the level and $0 \leq \alpha \leq 1$,

iv. $\beta$ is the smoothing parameter for the trend, $0 \leq \beta \leq 1$.

v. $h$ is the step size. For instance, $h = 3$ indicates projection of RR after 3 weeks.

The level equation shows that $l_t$ is a weighted average of observation $RR(t)$ and $(l_{t-1} + b_{t-1})$. The trend equation shows that $b_t$ is a weighted average of the estimated trend of RR score at time $t$ based on the previous estimate of the trend which is $(l_t - l_{t-1})$. One issue of applying HLT is providing appropriate values for the smoothing parameter $\alpha$ and $\beta$. However, in sophisticated statistical tools these values are estimated automatically using advanced method such as ARIMA [83]. Product manager can also provide these values by identifying the patterns of the RR time series. Detail of the HLT method can be found in [23].

### 3.3.2 Projection of RR with the Consideration of Trends and Seasonal Patterns

Holt-Winters method is applied successfully in various fields as a robust, easy-to-use projection procedure [84]. It is an extended version of the HLT method to capture seasonality. Along with the level ($l_t$) and trend ($b_t$), a seasonal component ($s_t$) is used in the projection equation. Another smoothing parameter $\gamma$ is provided for the seasonal component. RR time series may have seasonal pattern due to the repetitive development activities through the iterations. There are two variations of this method based on the nature of the seasonal patterns. When seasonal component is constant through the time series an additive model of the method is preferred. A multiplicative method is preferred when the seasonal variations are changing proportional to the level of the series. Details about this projection method can be found in [81].
3.4 An Illustrative Example of RR Calculation

In order to facilitate deeper understanding of the fundamental concepts used in the proposed approach, an illustrative example is presented below with the real data from an open source project hosted in Github\(^1\).

3.4.1 Context of the Project

Publify\(^4\) is a powerful open source blogging engine hosted in Github\(^1\). It is one of the oldest Ruby on Rails project started back in 2004. Since then, 52 contributors have engaged themselves on this project and contributing continuously to its development. It uses Github for managing issues (e.g. features, bugs, tasks, etc.) and source code versioning. For the continuous integration (CI) it uses the Travis-CI\(^5\) which is a cloud hosted CI platform. The quality of the source code of the project is tracked with CodeClimate\(^6\) that performs real-time static analysis on the source code repository in order to provide a comprehensive quality report. The latest version (Publify 8.0) of the engine was released on March 01, 2014. In this illustrative example, release readiness calculation steps of RELREA are shown for \(t = 170^{th}\) day (March 01, 2014) of the development of Publify 8.0.

3.4.2 Definition of RR Attributes, Metrics and Relative Weights

Following the proposed workflow, in step 1 important readiness attributes and metrics were defined from the four dimensions of the Publify project. Definition of the RR attributes \((a_i)\), metrics and their relative weights \((u_i)\) are shown in Table 3-5.

---

\(^4\) https://github.com/publify/publify
\(^5\) https://travis-ci.org/
\(^6\) https://codeclimate.com/
Table 3-5: Definition of the selected RR attributes, metrics definitions and relative weights used in the illustrative example.

<table>
<thead>
<tr>
<th>Readiness Dimensions</th>
<th>Release Readiness Attributes ( (a_i) )</th>
<th>Attributes Weights ( (u_i) )</th>
<th>Metrics Definitions</th>
<th>Values of the metric ( (x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Implementation</td>
<td>( a_1 ): Satisfaction of feature implementation</td>
<td>0.25</td>
<td>Feature implementation ratio</td>
<td>38.46</td>
</tr>
<tr>
<td></td>
<td>( a_2 ): Satisfaction of build/continuous integration trends</td>
<td>0.05</td>
<td>Percentage of successful builds/integration</td>
<td>62.34</td>
</tr>
<tr>
<td></td>
<td>( a_3 ): Satisfaction of codebase stability</td>
<td>0.15</td>
<td>Code Churn per contributor per day</td>
<td>150</td>
</tr>
<tr>
<td>Testing (Quality)</td>
<td>( a_4 ): Satisfaction of defect finding</td>
<td>0.2</td>
<td>Defect find rate (per day)</td>
<td>0.22</td>
</tr>
<tr>
<td></td>
<td>( a_5 ): Satisfaction of unit test</td>
<td>0.15</td>
<td>Covered LOC/ total LOC</td>
<td>84.58</td>
</tr>
<tr>
<td>Source Code Quality</td>
<td>( a_6 ): Satisfaction of codes smells</td>
<td>0.05</td>
<td>Number of code smells per class</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td>( a_7 ): Satisfaction of method complexity</td>
<td>0.05</td>
<td>Average method complexity</td>
<td>8</td>
</tr>
<tr>
<td>Documentation</td>
<td>( a_8 ): Satisfaction of readme file</td>
<td>0.1</td>
<td>Percentage of issues fixed related to documentation</td>
<td>100</td>
</tr>
</tbody>
</table>

3.4.3 Definition of the Membership Function of the Selected RR Attributes

Definitions of the membership functions and their parameters for the selected RR attributes are shown in Table 3-6. As shown in Table 3-6, two types of shapes are used to describe the membership functions of the selected RR attributes. *Piecewise decreasing* shapes are applied to characterize the membership function of the satisfaction of codebase stability and satisfaction of defect finding. Membership functions of the other RR attributes are defined with *piecewise increasing* shapes. Parameters \( (p, q) \) of the membership functions are defined by analyzing the previous releases of the project. As software development is dynamic and
unpredictable by nature, definition of the membership functions can be changed during a release cycle. For instance, at the middle of the release cycle, management may discover that most of reported bugs are not critical. In that case, the product manager may need to tune the membership function of the satisfaction of defect finding to capture the true readiness status of the software. With the earlier definition, she might be fully satisfied when the defect find rate is 0.1 (0.1 defects per day), but in the current context she might be fully satisfied if the defect find rate is 0.15 (0.15 defects per day). Thus, through the release cycle the product manager will learn more about the characteristics of the project and adjust the parameters of the membership functions of the RR attributes.

Table 3-6: Definition of membership functions of the RR attributes

<table>
<thead>
<tr>
<th>Release Readiness Attributes ($a_i$)</th>
<th>Metrics Definitions ($m_i$)</th>
<th>Selected Shapes</th>
<th>Parameters</th>
<th>Values of the metrics ($x_i$)</th>
<th>$\mu_{a_i}(x_i)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$ : Satisfaction of feature implementation</td>
<td>Feature implementation ratio</td>
<td><img src="image" alt="Graph" /></td>
<td>$p$ : 10, $q$ : 50</td>
<td>$x_i$ : 38.46</td>
<td>0.71</td>
</tr>
<tr>
<td>$a_2$ : Satisfaction of build/continuous integration trends</td>
<td>Percentage of successful builds/integration</td>
<td><img src="image" alt="Graph" /></td>
<td>$p$ : 30, $q$ : 80</td>
<td>$x_i$ : 62.34</td>
<td>0.65</td>
</tr>
<tr>
<td>$a_3$ : Satisfaction of codebase stability</td>
<td>Code Churn per contributor per day</td>
<td><img src="image" alt="Graph" /></td>
<td>$p$ : 50, $q$ : 200</td>
<td>$x_i$ : 150</td>
<td>0.67</td>
</tr>
<tr>
<td>Release Readiness Attributes ($a_i$)</td>
<td>Metrics Definitions ($m_i$)</td>
<td>Selected Shapes</td>
<td>Parameters</td>
<td>Values of the metrics ($x_i$)</td>
<td>$\mu_{a_i}(x_i)$</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------</td>
<td>-----------------</td>
<td>------------</td>
<td>-----------------------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>$a_4$ : Satisfaction of defect finding</td>
<td>Defect find rate (per day)</td>
<td><img src="image1" alt="Graph" /></td>
<td>$p$</td>
<td>$q$</td>
<td>0.22</td>
</tr>
<tr>
<td>$a_5$ : Satisfaction of unit test</td>
<td>Covered LOC/ total LOC</td>
<td><img src="image2" alt="Graph" /></td>
<td>60</td>
<td>90</td>
<td>84.58</td>
</tr>
<tr>
<td>$a_6$ : Satisfaction of codes smells</td>
<td>Number of code smells per class</td>
<td><img src="image3" alt="Graph" /></td>
<td>1</td>
<td>0.25</td>
<td>0.6</td>
</tr>
<tr>
<td>$a_7$ : Satisfaction of method complexity</td>
<td>Average method complexity</td>
<td><img src="image4" alt="Graph" /></td>
<td>20</td>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>$a_8$ : Satisfaction of readme file</td>
<td>Percentage of issues fixed related to documentation</td>
<td><img src="image5" alt="Graph" /></td>
<td>50</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>
3.4.4 Data Collection and Computing Attributes Satisfaction Levels

At \( t = 170^{th} \) development day of the Publify 8.0, required data was collected from the Github, Travis-CI and CodeClimate to calculate the values of the predefined metrics. Calculated values \( (x_t) \) of the metrics are shown in Table 3-5. According to the definition of their membership functions, attributes satisfaction levels \( \mu_{a_i}(x_t) \) were calculated which are also shown in Table 3-5.

3.4.5 Calculation of the Order Weights for the OWA Operators

For a specific decision strategy and the given attributes weights the order weights of the OWA operator is calculated. Let assume, the decision strategy is moderately optimistic which means some attributes need to be satisfied, not necessarily all of them. The term “some” is represented by the linguistic quantifier \( Q(p) = p^{0.5} \) with Eq. (3-1) where 0.5 is the value of \( \alpha \). Now, reordering of the degree of satisfaction of the RR attributes leads to:

\[
\begin{align*}
           b_1 & = 1 = \mu_{a_8}(x_t), \text{ hence } v_1 = u_8 = 0.1, \\
           b_2 & = 0.86 = \mu_{a_4}(x_t), \text{ hence } v_2 = u_4 = 0.2, \\
           b_3 & = 0.82 = \mu_{a_5}(x_t), \text{ hence } v_3 = u_5 = 0.15, \\
           b_4 & = 0.80 = \mu_{a_7}(x_t), \text{ hence } v_4 = u_7 = 0.05, \\
           b_5 & = 0.71 = \mu_{a_1}(x_t), \text{ hence } v_5 = u_1 = 0.25, \\
           b_6 & = 0.67 = \mu_{a_3}(x_t), \text{ hence } v_6 = u_3 = 0.15, \\
           b_7 & = 0.65 = \mu_{a_2}(x_t), \text{ hence } v_7 = u_2 = 0.05, \\
           b_8 & = 0.53 = \mu_{a_6}(x_t), \text{ hence } v_8 = u_6 = 0.05,
\end{align*}
\]

Eq. (3-4) is applied to calculate the order weights \( (w_i) \) according to the decision strategy \( (\alpha) \) and attributes weights \( (u_i) \):
Here,

\[ \sum_{k=1}^{8} v_k = 1, \]

Now according to Eq. (3-4) ordered weights are:

\[
\begin{align*}
    w_1 &= \left( \frac{\sum_{k=1}^{8} v_k}{\sum_{k=1}^{8} v_k} \right)^{0.5} - \left( \frac{\sum_{k=1}^{7} v_k}{\sum_{k=1}^{7} v_k} \right)^{0.5} = w_i = \left( \frac{0.1}{1} \right)^{0.5} - \left( \frac{0}{1} \right)^{0.5} = 0.32 \\
    w_2 &= \left( \frac{\sum_{k=1}^{8} v_k}{\sum_{k=1}^{8} v_k} \right)^{0.5} - \left( \frac{\sum_{k=1}^{7} v_k}{\sum_{k=1}^{7} v_k} \right)^{0.5} = w_i = \left( \frac{0.3}{1} \right)^{0.5} - \left( \frac{0.1}{1} \right)^{0.5} = 0.23 \\
    w_3 &= \left( \frac{\sum_{k=1}^{8} v_k}{\sum_{k=1}^{8} v_k} \right)^{0.5} - \left( \frac{\sum_{k=1}^{7} v_k}{\sum_{k=1}^{7} v_k} \right)^{0.5} = w_i = \left( \frac{0.45}{1} \right)^{0.5} - \left( \frac{0.3}{1} \right)^{0.5} = 0.12 \\
    w_4 &= \left( \frac{\sum_{k=1}^{8} v_k}{\sum_{k=1}^{8} v_k} \right)^{0.5} - \left( \frac{\sum_{k=1}^{7} v_k}{\sum_{k=1}^{7} v_k} \right)^{0.5} = w_i = \left( \frac{0.5}{1} \right)^{0.5} - \left( \frac{0.45}{1} \right)^{0.5} = 0.04 \\
    w_5 &= \left( \frac{\sum_{k=1}^{8} v_k}{\sum_{k=1}^{8} v_k} \right)^{0.5} - \left( \frac{\sum_{k=1}^{7} v_k}{\sum_{k=1}^{7} v_k} \right)^{0.5} = w_i = \left( \frac{0.75}{1} \right)^{0.5} - \left( \frac{0.5}{1} \right)^{0.5} = 0.16 \\
    w_6 &= \left( \frac{\sum_{k=1}^{8} v_k}{\sum_{k=1}^{8} v_k} \right)^{0.5} - \left( \frac{\sum_{k=1}^{7} v_k}{\sum_{k=1}^{7} v_k} \right)^{0.5} = w_i = \left( \frac{0.9}{1} \right)^{0.5} - \left( \frac{0.75}{1} \right)^{0.5} = 0.08 \\
    w_7 &= \left( \frac{\sum_{k=1}^{8} v_k}{\sum_{k=1}^{8} v_k} \right)^{0.5} - \left( \frac{\sum_{k=1}^{7} v_k}{\sum_{k=1}^{7} v_k} \right)^{0.5} = w_i = \left( \frac{0.95}{1} \right)^{0.5} - \left( \frac{0.9}{1} \right)^{0.5} = 0.03 \\
    w_8 &= \left( \frac{\sum_{k=1}^{8} v_k}{\sum_{k=1}^{8} v_k} \right)^{0.5} - \left( \frac{\sum_{k=1}^{7} v_k}{\sum_{k=1}^{7} v_k} \right)^{0.5} = w_i = \left( \frac{1}{1} \right)^{0.5} - \left( \frac{0.95}{1} \right)^{0.5} = 0.03 \\
\end{align*}
\]

Key characteristic of the above order weights is that more weights are given to the attributes which are important and highly satisfied than the others. According to Eq. (2-13), degree of optimism of the calculated ordered weights is 0.72.

**3.4.6 Computation of Overall Release Readiness**

Now the calculated ordered weights are used to compute the aggregated overall RR at \( t = 170 \) development days using the Eq. (3-5) as follows,

\[
RR(t = 170) = OW A_{i=1\ldots8} \left( \mu_{a_i}(x_t) \right) = \sum_{i=1}^{8} w_i \cdot b_i,
\]
\( RR(t = 170) = (0.32 \times 1) + (0.32 \times 1) + (0.32 \times 1) + (0.32 \times 1) + \)
\( (0.32 \times 1) + (0.32 \times 1) + (0.32 \times 1) = 0.84 \)

**Table 3-7: Correspondence among decision strategy, order weights, degree of optimism, and aggregated RR scores.**

<table>
<thead>
<tr>
<th>Decision Strategy names</th>
<th>Linguistic Quantifier (Q)</th>
<th>ordered weights ((w_i))</th>
<th>Optimism ((w_j))</th>
<th>(RR(t))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Optimistic</td>
<td>Few attributes</td>
<td>(0.63) 0.16 0.07 0.02 0.07 0.04 0.01 0.01</td>
<td>0.86</td>
<td>0.92</td>
</tr>
<tr>
<td>Moderately Optimistic</td>
<td>Some attributes</td>
<td>(0.32) 0.23 0.12 0.04 0.16 0.08 0.03 0.03</td>
<td>0.72</td>
<td>0.84</td>
</tr>
<tr>
<td>Neutral</td>
<td>Half of the attributes</td>
<td>1 0.10 0.20 0.15 0.05 0.25 0.15 0.05 0.05</td>
<td>0.5</td>
<td>0.77</td>
</tr>
<tr>
<td>Moderately Pessimistic</td>
<td>Many attributes</td>
<td>2 0.01 0.08 0.11 0.05 0.31 0.25 0.09 0.10</td>
<td>0.40</td>
<td>0.71</td>
</tr>
<tr>
<td>Pessimistic</td>
<td>Most attributes</td>
<td>5 0.00 0.00 0.02 0.01 0.21 0.35 0.18 0.23</td>
<td>0.24</td>
<td>0.65</td>
</tr>
</tbody>
</table>

The aggregated overall RR score 0.84 indicates that if the product is released at this point, then the product manager will be overall 84% satisfied. Using the above procedure, calculated RR \((t=170)\) scores and corresponding ordered weights for the other decision strategies, i.e., *optimistic, moderately optimistic, neutral, moderately pessimistic, and pessimistic* are shown in Table 3-7. Order weights emphasize certain position in the re-ordered levels of satisfaction of the RR attributes as indicated by bold font. With the *optimistic* decision strategy, overall degree of RR of *Publify 8.0* is the highest \((0.92)\) because order weights emphasize more on the outstanding attribute \(a_8\) and \(a_4\). In other words, these highly satisfied attributes are compensated with poorly satisfied attributes \(a_2\) and \(a_6\). Conversely, using *pessimistic* decision strategy overall degree of RR of the project is the lowest \((0.65)\). In this case, order weights
emphasize more on the poorest attributes compared to the outstanding attributes. As least risk is taken in this decision strategy, the product manager will be on the safe side when determining the overall degree of RR. The \textit{neutral} decision strategy assigns relative weights of attributes as order weights of OWA operator. The overall RR values decreases from \textit{optimistic} to \textit{pessimistic} decision strategy due to the nature of the OWA method. Investigation of RR with various decision strategies is useful when the product manager need to make trade-off between highly satisfied and poorly satisfied RR attributes. From Table 3-7, she can decide which distribution of order weights is most appropriate and how much risk she is taking in determining the overall degree of RR. More about the applicability and usefulness of the proposed approach is explored in Chapter 5 through a comprehensive case study.

\textbf{3.5 Summary}

In this chapter, the proposed continuous RR evaluation approach RELREA is introduced. In the approach RR is determined by aggregating the degree of satisfaction of the important RR attributes. Using this approach, the product manager can monitor RR at any point of time during the release cycle and study the projection of RR at the release time. Fundamental concepts and workflow of the approach are also described. An illustrative example was used to facilitate a deeper understanding of the fundamental concepts and RR calculation steps of the proposed approach. In the next chapter, a comprehensive RR analysis tool is described that implements the steps involved in the proposed approach. Tool support allows product managers truly evaluate the approach presented in this chapter.
Chapter Four: **Tool Support**

### 4.1 Introduction

In this chapter, design and implementation of the proposed RELREA approach as an integrated tool is described. The tool automates the processes such as data collection, defining RR attributes and membership functions, OWA weights calculation, RR computation and RR analysis. Tool support allows the users (product managers) to actually evaluate the proposed approach and methodologies in real life situations. In the context of software product management, providing tool support with the integration of the existing project management tools is more meaningful and effective. Existing popular development and project management tools such as Github\(^1\), TFS\(^3\), and Jira\(^2\) provides state-of-the-art facilities for managing, gathering, and tracking large amount of software project data. But most of the data analysis facilities are specific to the developers, not for product managers. As discussed in the state of the tool support in Section 2.3.3, release readiness investigation related features are very limited in the existing tools. By analyzing the existing tools following requirements are formulated for designing and developing a comprehensive RR analysis tool:

- **R1**: The tool should provide a set of pre-defined RR attributes and their corresponding metrics from important RR dimensions so that the product manager can select the most relevant one using GQM method.

- **R2**: The tool should integrate the existing project management tools so that required data (e.g. bug reports, code change history, state of the features, etc.) for calculating RR metrics can be collected automatically.
- R3: The tool should allow product manager’s evaluating (by setting targets or characterising the RR metrics) degree of satisfaction of the RR attributes based on the objective measures.

- R4: The tool should provide an interactive visual chart for monitoring the status of the degree of overall RR at any point of time in the release cycle.

- R5: The tool should have the drill-down capability so that product managers can investigate the individual level of satisfaction of the RR attributes for gaining insights about the limiting factors.

- R6: The tool should allow product managers studying projection of RR at the release time.

- R7: The tool should provide visual indicator so that product managers can understand the impact of the individual RR on overall RR.

The continuous release readiness analysis tool presented in this chapter addresses these requirements. The tool is designed and implemented as a web application on top of Github¹ and Jira² following the guideline presented in [31]. In the following sections, development platform, architecture and system modules of the tool are explained.

4.2 Architecture of the Tool

The tool was implemented as a web application on top of the existing project management and tracking tools which are Github¹ and Jira². Github is one of the most popular distributed project managements and source code control systems. Key features of Github include integrated issues tracking, collaborative code review, and dashboard for monitoring the development activities. Data related to key release readiness attributes such as source code stability, build status, source code quality can be retrieved from Github. On the other hand, Jira is a popular web
based project management tool which provides services such as bug tracking, project tracking, agile planning, test management, customer feedback, and others. RR attributes related to implementation status, defect status, testing scope and status can be evaluated with the data available in Jira. Both Github and Jira provides REST API to support integration with other tools and technologies. As shown in Figure 4-1, data tier of the tool access the live data from Github and Jira via API and store them in local database for evaluating RR attributes. The application tier of the tool contains the logic and services required to support continuous RR monitoring and analysis. It uses the data available in data tier to implement the services. Key services includes data collection, RR attributes evaluation, RR computation and monitoring, charts and graphs for RR analysis and visualization. The presentation tier of the tool contains various GUI components of the tool which invoke the services hosted in the application tier via HTTP. It also allows the product manager to interact with the tool via web client.

MVC based three tier architecture of the tool is shown in Figure 4-1. Three tier architecture provides benefits such as reusability, flexibility, manageability, maintainability, and scalability [85]. For instance, integration of another project management tool requires modification only in the data tier. Similarly, new services can be added easily in the application tier of the tool. Thus, the choice of three tier architecture was appropriate for developing the tool.
4.3 Development Platform

The tool is developed in Ruby on Rails (ROR) platform which is an open source full-stack web application framework written in Ruby Programming Language. The principal of ROR is “Convention over Configuration” which means a developer only needs to specify unconventional aspects of the application. The Model View Controller (MVC) architecture of the framework is integrated (see Figure 4-1) with the three tier architecture of the tool. The integrated Object Relational Model (ORM) component of the framework provides easy and flexible way of manipulating data stored in the database. An overview of the tools and technologies used for managing, developing and deployment of the tool are shown in Table 4-1. As described in Table
4-1, source code of the tool is hosted in Github repository\(^7\) as open source. The tool itself is hosted in heroku which is a cloud based software deployment framework. Using the url in [55], demo of the tool can be accessed.

**Table 4-1: Technologies used in the development of the tool**

<table>
<thead>
<tr>
<th>Tool and Technologies</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ruby on Rails (version 4.0)</td>
<td>Used as development platform</td>
</tr>
<tr>
<td>Github</td>
<td>For managing and hosting source code</td>
</tr>
<tr>
<td>Ruby</td>
<td>Used as application tier programming language</td>
</tr>
<tr>
<td>HTML, Javascript, JQery</td>
<td>Used to develop the view (web pages) of the tool</td>
</tr>
<tr>
<td>Highchart</td>
<td>Used to generating charts for RR analysis</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>Production database of the tool</td>
</tr>
<tr>
<td>Heroku</td>
<td>Tool deployment</td>
</tr>
</tbody>
</table>

### 4.4 Process Steps of the Tool

The tool implements the workflow of the RELREA approach described in Section 3.2. In order to monitor RR, the product manager needs to complete the following process steps of the tool.

#### 4.4.1 Creating a Project

To use the tool, at first the product manager needs to create a new project using the interface shown in Figure 4-2. In order to create a new project she needs to provide the unique Github *username* and *repository name*. If the project uses Jira for tracking its development activities, then option for Jira can be selected. Once the project is created, it will be added in the project list below. A “Red” (see Mongo DB project in Figure 4-2) indicator associated to a project indicates required process steps (settings) for monitoring RR is not completed yet. When

\(^7\) [https://github.com/shawniut/RELREA](https://github.com/shawniut/RELREA)
the processes are completed the indicator will change into “green” which means the project is ready for monitoring RR.

4.4.2 Completion of Required Settings

In this step, the product manager needs to define the project information (i.e. title, project start date, and release date), data sources, RR attributes, RR metrics, and relative weights of the RR attributes required for monitoring RR. A screenshot of the settings panel of the tool for managing these processes is shown in Figure 4-3. By clicking on the “Edit” link, the product manager can expand each section and complete the corresponding process.

![Figure 4-2: Tool interface for creating a new project](image)

![Figure 4-3: Settings interface of the tool for managing project information, RR attributes, membership functions, and relative weights](image)
4.4.2.1 Managing Project Information

Interface of the tool for managing project information is shown in Figure 4-4. Using this interface the product manager can provide the required credential for accessing data from the integrated tools. In the current implementation of the tool as data sources Github, Jira, or both can be selected. Other information such as project start date and next release date can be specified using this interface.

![Figure 4-4: Tool interface for managing the project info]

4.4.2.2 Managing RR Attributes and Metrics

The current implementation of the tool provides a pre-defined set of RR attributes and metrics which are shown in Figure 4-5. From those RR attributes, the product manager can select the important ones for the project to be evaluated.
### 4.4.2.3 Managing Relative Weights of the RR Attributes

Product manager also needs to provide the weights of the selected RR attributes using the interface shown in Figure 4-6. In addition, she can customize the data definition of the selected RR metrics by providing specific values of the associated variables. For instance, name of the “label/tag” (see Issue Label field in Figure 4-6) used for identifying an issue of type feature can be provided. After the completion of this step, the tool is ready for collecting data from the data sources which are used for calculating the values of the selected metrics.

<table>
<thead>
<tr>
<th>Select</th>
<th>Source Tool</th>
<th>Readiness Attributes</th>
<th>Acronym</th>
<th>Metrics</th>
<th>Readiness Dimension</th>
</tr>
</thead>
<tbody>
<tr>
<td>☑</td>
<td>Github</td>
<td>Satisfaction of codebase stabilization</td>
<td>CCR</td>
<td>Code churn rate</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>Github</td>
<td>Satisfaction of defect finding</td>
<td>DFR</td>
<td>Defect find rate</td>
<td>Testing</td>
</tr>
<tr>
<td>☑</td>
<td>Github</td>
<td>Satisfaction of bug fixing</td>
<td>BFR</td>
<td>Bug fix rate</td>
<td>Testing</td>
</tr>
<tr>
<td>☑</td>
<td>Github</td>
<td>Satisfaction of pull request completion</td>
<td>PCR</td>
<td>Pull-request Completion Rate</td>
<td>Testing</td>
</tr>
<tr>
<td>☑</td>
<td>Github</td>
<td>Satisfaction of coding accuracy</td>
<td>DD</td>
<td>Defect density</td>
<td>Testing</td>
</tr>
<tr>
<td>☑</td>
<td>Github</td>
<td>Satisfaction of feature completion</td>
<td>FCR</td>
<td>Feature completion rate</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>Github</td>
<td>Satisfaction of feature implementation</td>
<td>FI</td>
<td>Features Implemented</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>JIRA</td>
<td>Satisfaction of low priority feature implementation</td>
<td>LP-FIR</td>
<td>Low priority Features Implementation ratio</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>JIRA</td>
<td>Satisfaction of high priority improvement implementation</td>
<td>HP-IIR</td>
<td>High priority improvement implementation ratio</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>JIRA</td>
<td>Satisfaction of low priority improvement implementation</td>
<td>LP-IIR</td>
<td>Low priority improvement implementation ratio</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>JIRA</td>
<td>Satisfaction of improvement implementation</td>
<td>IIR</td>
<td>Improvement implementation ratio</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>JIRA</td>
<td>Satisfaction of bug fixing</td>
<td>BFR</td>
<td>Bug fix rate</td>
<td>Testing</td>
</tr>
<tr>
<td>☑</td>
<td>JIRA</td>
<td>Satisfaction of feature completion</td>
<td>FCR</td>
<td>Feature completion rate</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>JIRA</td>
<td>Satisfaction of improvement implementation</td>
<td>II</td>
<td>Improvement Implemented</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>JIRA</td>
<td>Satisfaction of feature implementation</td>
<td>FI</td>
<td>Features Implemented</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>JIRA</td>
<td>Satisfaction of high priority feature implementation</td>
<td>HP-FIR</td>
<td>High priority Features implementation ratio</td>
<td>Implementation</td>
</tr>
<tr>
<td>☑</td>
<td>Travis-CI</td>
<td>Satisfaction of build status</td>
<td>BSR</td>
<td>Build success rate</td>
<td>Implementation</td>
</tr>
</tbody>
</table>
4.4.2.4 Managing Values of the Metrics

Using the interface shown in Figure 4-8, values of the metrics can be managed by the product manager. As shown in the screenshot, she can specify the observation interval of the defined RR metrics which is used by the tool for the retrieval of the required data from the pre-defined data sources. The values of the metrics are calculated and stored instantly in the local database for further RR analysis.

Figure 4-7: Tool interface for managing the values of the metrics
4.4.2.5 Defining Membership Functions

In this step, the product manager needs to define the shapes and parameters of the membership functions of the selected RR attributes and metrics. Screenshot of the interface of the tool designed for managing membership functions is shown in Figure 4-8. As shown in Figure 4-8, two shapes are integrated in the tool which can be assigned to describe the characteristics of the RR attributes. For instance, as shown in the screenshot, Shape 2 is selected as membership function of the satisfaction of defect finding attribute. After selecting the appropriate shape she needs to provide the values of the parameters associated to the shape. To assist her, maximum and minimum values of the metrics found in the current and past releases are provided. Based on the personal judgement and context of the product, the product manager can define the values of the parameters of the membership functions. In the screenshot, the value of the first parameter is 0.07 which indicates the product manager is fully satisfied when defect find rate is 0.07 per day. The other parameter is defined as 1 which means the product manager is not satisfied at all when defect find rate is 1 per day. Defined shapes and parameters of the MFs can be tuned at any point of time using this interface. Once the membership functions are defined for all RR attributes, the tool is ready to evaluate and monitor RR.
4.5 High Level Use-Cases

To better illustrate the capabilities of the tool, three high-level use cases (see Figure 4-9) are described from the product manager’s perspective in the following sub-sections.
4.5.1 **Use Case 1: Monitoring and Analyzing RR Over Time**

In this use case, the product manager can monitor and analyze the RR over time using the interface of the tool shown in Figure 4-10. Based on the pre-defined settings discussed in Section 4.4, aggregated overall degree of RR is calculated over time. This allows the product manager to know how the RR scores have been evolved over time. To make the long term trends clearer, tool allows applying moving average smoothing technique. As shown in Figure 4-11, the product manager can select an appropriate moving average order to see RR trends clearly. More about analysis on trends and patterns of RR scores over time is discussed in Chapter 5 with a case study.
4.5.2 Use Case 2: Studying Projection of RR

In this use case, the product manager can study the projection of RR into future. The HLT projection method discussed in Section 3.3 is implemented in the tool. It allows the product manager to know the projected RR score at the next week or at the scheduled release week based on the past RR scores. Screenshot of the tool for studying projection of RR is shown in Figure
4-12. By looking at the trends and patterns of the RR time series, the product manager can specify the level and trend constant required for applying the projection method. For instance, as shown in Figure 4-12, projected RR after 5 weeks is 0.508. When projected RR score is below the requested RR, the product manager can start investigating the limiting factors or revise the pre-defined release date. The accuracy of the projection method is evaluated with a case study described in Chapter 5.

![Figure 4-12: Tool interface for studying projection of RR](image)

4.5.3 Use Case 3: Gaining Insights about Limiting Factors

The tool helps the product manager gaining insights about the factors which limit the RR with the visualization of the level of satisfaction of the individual RR attributes. Knowledge about limiting factors allows the product manager re-allocating or adding resources for improving RR. Screenshot of the tool where impact of the individual RR attributes on overall RR over time is shown in Figure 4-13. The green area shows to what extent and when the level of satisfaction of a RR attribute is above the overall RR. Conversely, red area indicates to what extend and when the degree of satisfaction of a RR attribute is below the overall RR. For
instance, by looking at the visualization shown in Figure 4-13, one product manager can easily notice that attributes such as IIR, LP-FIR, and HP-FIR are the factors which have continuous negative impact on the overall RR. Moreover, by looking at the RR attributes together she can gain insight about the limiting factors in achieving RR.

![Release Readiness Monitoring and Analysis Dashboard](image)

*Figure 4-13: Tool interface for gaining insights about the limiting factors.*

4.6 Summary

The tool is developed as a web application using Ruby on Rails framework. It can be used for evaluating and analyzing RR for the software projects which use Github\(^1\) as source code repository and Jira\(^2\) for managing and tracking development activities. In this chapter,
requirements of the tool, overall system architecture, process workflow and three high level use cases of the tool are presented. The tool addresses the limitation of the existing tools discussed in Section 2.3.3. Specifically, projection of RR and facility for identifying limiting factors with visualization technique are unique features of the tool. Overall the tool helps the product manager with automated, fast and continuous RR evaluation, monitoring and analysis. In the next chapter, the applicability of the tool is evaluated through a case study. More about the web interfaces of the tool are also explained in the next chapter.
Chapter Five: Case Study

A case study is a powerful empirical method which is primarily used for exploratory investigation, both prospectively and retrospectively that attempts to understand and explain phenomenon or to construct a theory [86]. The method has become popular in software engineering and is frequently used to understand, to explain or to demonstrate the capabilities of a new technique, methodology, tool, process, technology, or organizational structure. In this work, two case studies are performed from two different perspectives. The first case study is performed to evaluate the RR investigation capabilities of the RELREA approach and its implementation with a real life software project. The second case study is performed in the domain of OSS projects with the aim of demonstrating applicability of the proposed RR evaluation approach in monitoring frequencies and patterns of the bottleneck factors. An overview of the key characteristics of the case studies is presented in Table 5-1. Case studies are performed following the guidelines for conducting case studies in software engineering presented in [87].
Table 5-1: An overview of the key characteristics of the case studies

<table>
<thead>
<tr>
<th>Case Study Name</th>
<th>Key Characteristics</th>
</tr>
</thead>
</table>
| Case Study 1: Evaluation of the RELREA Approach and the Prototype Tool          | – A proprietary software project titled “Release Planner” in the domain of business intelligence solution for optimized release planning is used as case study project.  
|                                                                                | – Applicability of the RELREA approach and the prototype tool in evaluating, monitoring and investigating RR issues from the product manager’s perspective is evaluated.                              
|                                                                                | – Accuracy of the time series projection method for studying projection of RR at the release time is shown.                                                                                                           
|                                                                                | – Applicability of OWA aggregation operators i.e., the impact of various decision strategies in determining overall degree of RR is shown.                                                                               |
| Case Study 2: Monitoring Bottleneck Factors in Achieving RR                     | – Ten OSS projects hosted in Github¹ repository are used as case study projects.                                                                                                                                       
|                                                                                | – Applicability of the RELREA approach in monitoring bottleneck factors in the domain of OSS projects is evaluated.                                                                                                  
|                                                                                | – Occurrence frequency and patterns of the bottleneck factors in achieving RR are identified in the domain of OSS projects.                                                                                          |

In the following sections details of the case studies are discussed which include: objective of the case studies, context of the case studies, result analysis, and the threats to validities.

5.1 Case Study 1: Evaluation of the RELREA Approach and the Prototype Tool

This case study is a *descriptive* case study which involves details investigation of an attribute or phenomena with in-depth real life scenarios. In software engineering, a case may be anything that is a contemporary software engineering phenomenon in its real-life setting [88]. In this study, *release readiness* is the case which comes from an ongoing real life software project
titled “Release Planner 2.0” (RP2) developed for Expert Decision Inc\textsuperscript{8}, located in Calgary, Canada. As a single project is studied as a whole, this is a \textit{single case holistic} type of case study \cite{88}. Details of the case study design and result analysis are discussed in the following sub-sections.

5.1.1 Case Study Objective

The objective of the case study is to evaluate the proposed RELREA method and its implementation by demonstrating how it can assist a product manager in better investigating the RR at any point of time in the release cycle in a real-life software project. For this, the following case study questions are formulated:

- CQ 1: What is the degree of RR over a sequence of iterations? And how the monitored RR can be used to identify the indicator of RR difficulties at the end of the iterations and over the iterations?
- CQ 2: What is the impact of various decision strategies in evaluating overall degree of RR? And to what extent the current RR is below the requested RR with respect to various decision strategies?
- CQ 3: What is the projected RR for the next observation period? And how accurate is the projection method?
- CQ 4: What are the factors limiting the overall degree of RR?

The aim of the above case study questions is to provide various types of investigation facility to the product manager with respect to evaluating and improving the current RR of the

\footnote{http://www.expertdecisions.com/}
software. The case study results will demonstrate the applicability of RELREA and its capability in exploring these questions.

5.1.2 Context of the Case Study Project

The case study project RP2 is a business intelligence solution for optimizing portfolio planning, product development road-mapping, and human resource allocation planning. It is a result of 40 person years of R&D to date. Former versions of RP2 have proven successful with 100+ industrial clients globally. The development phase of the second version of the project was started back in July, 2013. The development team (five developers and testers) of the company has adapted iterative software development method for managing its development activities. At the time of writing this thesis, the team was working on the 4th iteration of the release cycle. In this study, RR of the past three iterations is studied retrospectively to explore the case study questions defined in section 5.1.1. Timeframe of the studied iterations are presented in Table 5-2. As shown in Table 5-2, length (12-14 weeks) of the iterations of RP2 is much long compared to the typical iteration length (2-6 weeks). According to the product manager, through the release cycle there were some unexpected events such as lead developer of the project had left the job at the middle of its development and testing resources were not always available which eventually prolonged the iterations.

Table 5-2: Duration of the studied iterations of RP2

<table>
<thead>
<tr>
<th>Name of the iteration</th>
<th>Start date – End date</th>
<th>Iteration Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iteration 1</td>
<td>July 15, 2013 – October 7, 2013</td>
<td>12 weeks</td>
</tr>
<tr>
<td>Iteration 2</td>
<td>October 14, 2013 – January 27, 2014</td>
<td>16 weeks</td>
</tr>
<tr>
<td>Iteration 3</td>
<td>February 3 – May 19, 2014</td>
<td>16 weeks</td>
</tr>
</tbody>
</table>
5.1.3 Data Sources of the Case Study

The availability and quality of the required data is important for performing a case study in software engineering [86]. In the context of RR evaluation, the case study project must use standard project management tools and technologies for managing and tracking its development activities. A brief description of the data sources of this case study is presented in Table 5-3. Based on the data available in Github\(^1\) and Jira\(^2\), RR of RP2 is evaluated for the implementation status and testing dimensions. As data required for source code quality and documentation dimensions were not tracked by the development team, these dimensions are not included in the RR evaluation process.

Table 5-3: Description and usage of the data sources of the case study project

<table>
<thead>
<tr>
<th>Data Sources</th>
<th>Description</th>
<th>Usage in RP2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Github(^1)</td>
<td>A popular web based environment for source code management.</td>
<td>Github(^1) was mainly used for versioning the source code. Commit history and addition/deletion on the source code files are also stored in Github. Data required for RR evaluation was accessed from Github using the tool described in Section 4.4.</td>
</tr>
<tr>
<td>Jira(^2)</td>
<td>A popular project management tool for planning, tracking and organizing issues.</td>
<td>RP2 management and development team used Jira for tracking development (i.e. new feature, improvement) and bug related issues. States (e.g. open, close, resolved, and re-opened) and priority (e.g. blocker, critical, major, minor, and trivial) of the issues were tracked using the web interface of the Jira(^2).</td>
</tr>
</tbody>
</table>
5.1.4 Instantiating RELREA in RP2

The tool described in previous chapter is used for instantiating RELREA in RP2 project. A project was created titled “Release Planner” in the tool for studying RR of RP2 retrospectively. In this section, instantiating process of the proposed RELREA approach is presented according to the workflow described in Section 3.2.

5.1.4.1 Defining the RR Attributes of RP2

As discussed in Section 3.1.2, GQM [21] method is applied to define the RR attributes and their corresponding metrics. Based on the data available in Github\textsuperscript{1} and Jira\textsuperscript{2}, a draft of the RR attributes and metrics was presented to the product manager of RP2. He reviewed the draft and adjusted the RR attributes according to the definition of business success of the product. For example, he suggested to define separate RR attribute for high priority and low priority feature implementation ratio. The revised final set of RR attributes and metric definition following the GQM [21] method is presented in Table 5-4. The approved RR attributes and metrics were entered in the tool following the steps described in Section 4.4.
Table 5-4: GQM based definition of RR attributes and their corresponding metrics of the case study project

<table>
<thead>
<tr>
<th>Goal</th>
<th>Refined Goal (RR attributes)</th>
<th>Questions</th>
<th>Metrics</th>
<th>Acronyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessing satisfaction of implementation</td>
<td>Satisfaction of high priority feature implementation</td>
<td>To what extent high priority features are implemented?</td>
<td>High priority (e.g. blocker, critical, major) feature implementation ratio</td>
<td>HP-FIR</td>
</tr>
<tr>
<td></td>
<td>Satisfaction of low priority feature implementation</td>
<td>To what extent low priority features are implemented?</td>
<td>Low priority (e.g. minor, trivial) feature implementation ratio</td>
<td>LP-FIR</td>
</tr>
<tr>
<td></td>
<td>Satisfaction of improvement implementation</td>
<td>To what extent approved improvements are implemented?</td>
<td>Improvement implementation ratio</td>
<td>IIR</td>
</tr>
<tr>
<td></td>
<td>Satisfaction of codebase stabilization</td>
<td>Is the code being stabilized over time?</td>
<td>Code churn rate (for the last 21 days)</td>
<td>CCR</td>
</tr>
<tr>
<td>Assessing satisfaction of testing</td>
<td>Satisfaction of defect finding</td>
<td>Are the testing activities reducing the defects?</td>
<td>Defect find rate (for the last 21 days)</td>
<td>DFR</td>
</tr>
<tr>
<td></td>
<td>Satisfaction of bug fixing</td>
<td>Is the bugs are being fixed regularly?</td>
<td>Bug fix ratio (for the last 21 days)</td>
<td>BFR</td>
</tr>
</tbody>
</table>

5.1.4.2 Collection of Data Using the Tool

As discussed in Chapter 4, the current implementation of the tool provides the facility to collect data from Github\(^1\) and Jira\(^2\) for calculating the values of the metrics over time. Using the web interface of the tool, the values of the metrics were collected for the period of around one year (44 weeks). As shown in Figure 5-1, values of the metrics were calculated with the interval of 7 days.
5.1.4.3 Defining Membership Functions

The RP2 product manager selected the appropriate shapes and parameters of the membership functions using the tool interface discussed in Section 4.4.2.5. The tool provided the maximum and minimum values of the metrics to help him in defining the parameters. For instance, the maximum and minimum values of DFR (defects found per day) were 1.62 and 0.0 respectively. By looking at these extreme values of DFR he might got some insights about the appropriate parameters of the selected membership function. Selected shapes and the values of their parameters for each metric are presented in Table 5-5. Given the values of the metrics at a particular week $t$, degree of satisfaction of the RR attributes are evaluated based on the definition of the membership functions.
<table>
<thead>
<tr>
<th>RR attributes</th>
<th>Metrics</th>
<th>Membership Function</th>
<th>MF Parameters</th>
</tr>
</thead>
</table>
| HP-FIR        | High priority (e.g. blocker, critical, major) feature implementation ratio | ![Shape 1](highcharts.com) | p = 0.3  
q = 1 |
|               |         | ![Shape 1](highcharts.com) | p = 0.3  
q = 0.8 |
| LP-FIR        | Low priority (e.g. minor, trivial) feature implementation ratio | ![Shape 1](highcharts.com) | p = 0.2  
q = 0.9 |
|               |         | ![Shape 1](highcharts.com) | p = 0.3  
q = 0.95 |
|               |         | ![Shape 1](highcharts.com) | p = 100  
q = 500 |
|               |         | ![Shape 2](highcharts.com) | p = 0.07  
q = 1 |
5.1.4.4 Defining Relative Weights of the RR Attributes

At this point, the RP2 product manager provided the relative weights of the RR attributes on a 0 to 1 scale. Weights were given based on his personal judgement about relative importance of the selected RR attributes. Provided relative weights of the RR attributes are shown in Table 5-6. The weights distribution scheme shows that more weights are given to the attributes related to feature implementation status compared to the attributes related to testing processes. In this case study, the same weights are used from evaluating RR through the observed duration of the release cycle.

Table 5-6: Definition of the relative weights of the RR attributes of the case study project

<table>
<thead>
<tr>
<th>Readiness Attributes</th>
<th>Readiness Dimension</th>
<th>Relative Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td>Satisfaction of high priority feature implementation</td>
<td>Implementation</td>
<td>0.25</td>
</tr>
<tr>
<td>Satisfaction of improvement implementation</td>
<td>Implementation</td>
<td>0.15</td>
</tr>
<tr>
<td>Satisfaction of low priority feature implementation</td>
<td>Implementation</td>
<td>0.15</td>
</tr>
<tr>
<td>Satisfaction of codebase stabilization</td>
<td>Implementation</td>
<td>0.1</td>
</tr>
<tr>
<td>Satisfaction of bug fixing</td>
<td>Testing</td>
<td>0.2</td>
</tr>
<tr>
<td>Satisfaction of defect finding</td>
<td>Testing</td>
<td>0.15</td>
</tr>
</tbody>
</table>

5.1.5 Analysis of the Results

In the following section, the results of the case study are discussed with respect to the case study questions described in Section 5.1.1.

5.1.5.1 CQ 1: Monitoring Degree of RR Over the Iterations

Monitoring the status of RR over the iterations is important for the product managers to plan and control the development. If any inconsistency in terms of RR is identified she can make corrective steps to improve the RR in the next iteration. In this case study, RR was monitored
over the iterations of the RP2 release cycle using the proposed RELREA approach and the tool described in Chapter 3 and Chapter 4 respectively. A screenshot of the RR monitoring interface of the tool is shown in Figure 5-3 which visualizes the overall degree of RR and individual degree of satisfaction of the RR attributes of RP2 in weekly basis. Two main use cases of monitoring RR are discussed below.

**Comparison of RR Status at the End of Each Iteration:** In the context of iterative development, it is expected that each iteration will progressively contribute to the delivery of the complete product. Iteration-to-iteration comparison of RR states allows the product manager to see how the iterations are contributing to the overall degree of RR. Comparison of the RR at the end of each iteration is shown in Figure 5-2. From Figure 5-2, it is evident that at the end of each iteration RR have improved by a certain degree. However, the improvement is not significant. For instance, at the end of iteration 2 and iteration 3, RR improvement is only 0.06 (from 0.39 to 0.45) and 0.05 (from 0.45 to 0.5) respectively. From the management perspective, this is definitely an indicator of RR issue which may cause delay in product shipment. To investigate more about the little contribution of the iterations, individual degree of satisfaction of the RR attributes are studied. The kiviat charts from the tool displaying the degree of satisfaction of the RR attributes at the end of each iteration are shown in Figure 5-4. From the charts following observation can be made:

- Degree of satisfactions of the testing related RR attributes such as BFR and DFR is as expected. Specifically, BFR was fully satisfied which indicates bug backlog is empty at the end of the iterations. However, the DFR is not fully satisfied which indicates that more testing effort is required to ensure high quality of the implemented functionality.
• Degree of satisfaction of the implementation status related RR attributes such as HP-FIR, LP-FIR and IIR is below the expected level. Only significant improvement of IIR is observed over the iterations. Little improvement in satisfaction of HP-FIR and LP-FIR indicates features are not being implemented as planned. Based on this objective evidence, the product manager can start investigating the root cause of the problem. For instance, she can check the state (e.g. blocked, resolved, postponed, etc.) of the features in the Jira or discuss this RR issue with the development team to know the actual fact.

• At the end of the iterations it is expected that the code base will be stabilized. However, the charts shows that CCR is not satisfied at all which indicates significant amount of addition/deletions was made in the last couple of weeks of the iterations which eventually acted as a limiting factor for being the RR score low.

![Figure 5-2: Comparison of RR of RP2 at the end of the iterations](image-url)
Figure 5-3: Screen of the tool showing the evolution of the overall RR and the degree of satisfaction of the individual RR attributes of RP2 over the iterations.

Figure 5-4: Screen of the tool visualizing individual attributes status at the end of the iterations.
**Evolution of Overall RR and Individual RR Attributes During the Iterations:** Though the comparison of RR status at the end of each iteration is important for planning for the next iterations, it is also essential to observe how the overall RR and individual RR attribute being evolved during the iterations. It helps product manager to identify the indicator of RR related problems at the early stage of the development. From the RR monitoring chart shown in Figure 5-3, the product manager can gain the following insights:

- During the last 8 weeks of the first iteration there is no change in degree of satisfaction of the RR attributes (LP-FIR, IIR, and HP-FIR) related to implementation of functionality. This indicates that no features or approved improvements have been fully implemented during this period. From the management perspective, it is an indicator of a RR problem which triggers the question “Is it actually possible to deliver the product within the resource allotted for implementation of the functionality?” Another issue is that after the week of August 13, 2013 overall degree of RR has gradually decreased until the week of August 26, 2013. However, this can be explained from the evolution of DFR during this period. In this time interval, a significant number of defects (from recently implemented features) were reported which lead to low satisfaction of DFR and eventually degraded the overall RR.

- Again during the second iteration there is no change in degree of satisfaction of the RR attributes (i.e. LP-FIR, IIR, and HP-FIR) related to implementation of functionality during the last 8-9 weeks. This confirms that change in plan for feature implementation is essential for improving the current RR status of the project. Satisfaction of CCR during the first 8 weeks of the second iteration is also an indicator of RR related problems. This indicates that very few (less than 500 addition/deletion per day) changes
are made by the development team in the code base during those weeks which is usually expected at the end of the iteration as evident in the first iteration. When this happens over a long period, management should investigate the root cause and take necessary actions to ensure proper implementation effort until the completions of the features planned for the iteration.

- In the third iteration, no significant change in overall degree of RR is noticed until the last few weeks. However, degree of satisfaction of the implementation related RR attributes have improved at the late stage of this iteration which is as expected. The main concern from the management perspective is that CCR is not satisfied at all during the last few weeks. This indicates that code base is not stable since developers are making lots of changes in the source code to implement features or to fix bugs. When this inconsistency is identified, the product manager can communicate this fact to the developers or monitor what changes are being made to manage the situation.

Overall, the above charts and discussion reveal that by applying the proposed RR monitoring approach and the prototype tool, it is possible to identify RR difficulties at the early stage of the release cycle. For instance, in the context of RP2 it was evident that management needs to revise the current plan (adding or re-allocating resources) for implementation effort to improve its RR status.

5.1.5.2 CQ 2: Impact of Various Decision Strategies in Evaluating Overall RR

As discussed in Section 3.1.5, aggregated RR is calculated by considering the product managers desired decision strategy and relative importance of the RR attributes. The key characteristic of the decision strategies is that risk associated in evaluating overall degree of RR gradually reduces from optimistic strategy to pessimistic strategy. Using the tool, the product
manager can see the overall RR evaluation with respect to five specific decisions strategies—optimistic, moderately optimistic, neutral, moderately pessimistic, and pessimistic as defined in Section 3.1.5.1. A set of RR evaluation based on various decision strategies helps the product manager to make more informed product release decision, specifically when time-to-market is crucial for business success of the product. It also provides an analytical ground for understanding the level of risk of the release decision.

For the case study project RR scores with various decision strategies is discussed for $t = 44$ week of the case study project. The results for the latest evaluation point ($t = 44$ weeks) is given in Figure 5-6 and Figure 5-5. The requested RR for releasing RP2 is defined 1 by the product manager which indicates the product will be released when all the attributes are fully satisfied. With an optimistic decision strategy the actual RR (0.79) is the highest because more weights (around 70%) are given to the outstanding (high priority and fully satisfied) attributes (BFR and DFR) compared to the poorest (least satisfied) attributes (LP-FIR and CCR). This means low satisfied attributes such as CCR (0% satisfied) and LP-FIR (20% satisfied) is compensated with highly satisfied attributes BFR (100%) and DFR (82%). When using a moderately optimistic decision strategy, the difference between actual RR (0.64) and requested RR is 0.36 because weight distribution is more balanced. In this case, only one low satisfied attribute CCR is compensated with fully satisfied BFR. The neutral decision strategy assigns relative weights of attributes as order weights of OWA operator and leads to RR score of 0.5 which is 50% below from the requested RR. The use of moderately pessimistic strategy shows actual RR (0.31) is much less (69 % less) compared to the requested RR because less risk is considered by giving more weights (around 50 %) among the low satisfied attributes LP-FIR and CCR. When using a pessimistic strategy the actual RR is lowest (0.18) which is 82% below.
the requested RR. In this strategy, least risk is taken by giving 80% weight to the poorest (least satisfied) attributes. Here, essentially the product manager will be on the safe side when determining the actual RR of the software.

Figure 5-5: Screenshot of the tool demonstrating the impact of various decision strategies at $t = 44$ weeks

<table>
<thead>
<tr>
<th>RR Attributes</th>
<th>BFR</th>
<th>DFR</th>
<th>IIR</th>
<th>HP-FIR</th>
<th>LP-FIR</th>
<th>CCR</th>
<th>Req. RR</th>
<th>Act. RR</th>
<th>Diff.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sat. deg</td>
<td>1.0</td>
<td>0.82</td>
<td>0.53</td>
<td>0.26</td>
<td>0.2</td>
<td>0.0</td>
<td>1.0</td>
<td>0.79</td>
<td>0.21</td>
</tr>
<tr>
<td>Rel W</td>
<td>0.2</td>
<td>0.15</td>
<td>0.15</td>
<td>0.3</td>
<td>0.1</td>
<td>0.1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OWA W</td>
<td>0.617</td>
<td>0.113</td>
<td>0.082</td>
<td>0.123</td>
<td>0.034</td>
<td>0.031</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 5-6: Screenshot of the tool presenting the distribution of OWA weights for various decision strategies for the RR at $t = 44$ weeks of the case study project.
The actual RR values decreases from optimistic to pessimistic decision strategy due to the nature of the OWA method. The selection of the aggregated degree of RR depends on the context of the project. RR scores obtained with optimistic strategy can be used when fulfillment of some of the important RR attributes are enough for making release decision. On the other hand, if it is required to fulfill all of the RR attributes, then RR score obtained with pessimistic decision strategy can be chosen. In the context of this case study, actual RR is close to the requested RR with the optimistic decision strategy. It is also known to the product manager that trade-off is made between fully satisfied attribute BFR with least satisfied attribute CCR. In contrast, according to the other decision strategies requested RR is not fulfilled at weeks which indicate significant amount of implementation effort is required to achieve the requested RR. Knowing the RR deficit objectively with the consideration of decision strategy can help the product managers to take corrective steps for improving the RR in near future.

5.1.5.3 CQ 3: Projected RR Score

As mentioned in Chapter 1, one of the main objectives of this study is to enable projection of RR at the pre-defined release date (or any point of time before the release date) so that the product manager can act proactively if any sign of problems regarding release window of the product is identified. In particular, if the projected RR score is below the requested RR she can think about making changes, i.e., process change or trade-off between quality and time-to-market of the release in the current plan. Projection of RR is also important for setting the customer expectations and marketing strategy. In this thesis, projected RR score is determined based on the monitored RR score with the time series projection method described in Section 3.3.
For the case study project, RR time series was generated using the tool which is shown in Figure 5-7. In order to identify an appropriate time series projection method it is pre-requisite to characterize the RR time series with respect to trends and seasonal patterns. Below the analysis of trends and seasonal patterns in RR time series of the case study project is discussed.

**Figure 5-7: Screenshot of the RR time series of the case study project**

**Trend Analysis:** As the actual RR time series of RP2 shown in Figure 5-7 shows some random fluctuation over time, we cannot directly determine whether or not a trend exists. Therefore, the moving average smoothing technique [89] was applied to make the long term trends clearer. The resultant RR time series with moving average order 3 is shown in Figure 5-8. From Figure 5-8, one can easily see that there is a long term RR growing trends in the time series. Particularly, at the earlier stage (between the weeks of August 12, 2013 and December 30, 2013) there is a clear RR growing trend. Later RR seems to stabilize and then, again until last evaluation week RR has improved over time. To validate this growing tendency, the linear trend line is fitted which is shown in Figure 5-9. The trend equation tells us that the predicted RR increase in each week is 0.006. The \( p \) value of the residuals of the trend line was 0.00003 which is less than the significance level 0.001. Therefore, it is clear that there is a significant long term growing trend in RR over time and this information can be used to predict the RR into future for this case study.
Figure 5-8: Moving average smoothing for visualizing the long term trends of the RR time series of the case study project

Figure 5-9: Linear trend analysis of RR of RP2

**Seasonality Analysis:** The seasonality pattern in RR time series was analyzed with the Auto Correlation Function (ACF) [80] which represents the linear correlation coefficient for consecutive lags in a time series. In time series analysis, the *lag operator* operates on a time series to produce the previous element. Similarity between observations as a function of time lag
between them is studied with ACF plot. ACF plots for 16 time lags of the RR time series is shown in Figure 5-10 with 5% significance limit. The ACF plot shows an exponentially decreasing pattern up to lag 7 where only the first four lags are statistically significant. In addition, no strong repetitive patterns are evident which suggests the absence of a seasonal pattern in the time series. Based on this evidence, it was confirmed that the RR time series has no significant seasonal patterns. At the same time, ACF plot also revealed that RR at a particular point of time is largely correlated with the RR of the most recent evaluation points. This indicates that most recent development activities (e.g. new feature implemented, bug fixed, new test case is executed, etc.) have more influence on the future RR of a software.

Figure 5-10: Autocorrelation plot of RR time series up to 16 lags
Based on the evidence of significant trends and autocorrelation in the RR time series of RP2, the HLT [90] method was applied to calculate the projected RR for the next observation period. As discussed in Section 3.3.1, HLT method allows us to emphasize more on the latest evaluation points by defining the smoothing constants for the level parameter. The trend of the past RR evaluations is also considered by providing the constants for the trend parameter. Plots of the projected RR and actual RR over time are shown in Figure 5-11. From Figure 5-11 we can see that the projected RR and actual RR are very close in most of the observation points. The forecasting accuracy of these methods is measured in Mean Absolute Percentage Error (MAPE) as stated in Eq. (5-1).

$$MAPE = \frac{1}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|$$  \hspace{1cm} (5-1)

Where $A_t$ is the actual value and $F_t$ is the projected value. The MAPE of the HLT forecasting method was 10%. According to the industry standard MAPE less than 20% is considered as reliable forecasting method [91].
According to the HLT method, the projected RR for \( T = 45 \) weeks (next observation point) and \( T = 76 \) (approximate release week of RP2) is studied using the tool. Screenshot of the projected RR for these weeks are shown in Figure 5-12 and Figure 5-13. Figure 5-12 shows that \( PR (T = 45) \) is 0.5 which is same as the current RR. This indicates that without any change in the current development and testing schedule RR of the software will not improve. As shown
in Figure 5-13, $PR(T = 76)$ is 0.56 which indicates that there is very less chance that the software will be ready to release by January 1st, 2015.

Overall, the above discussion shows that it is meaningful to project the RR into future and it allows the project manager to make proactive decision if the projected RR is below the requested RR. In the next section, how the product manager can gain insights about the factors which limit the RR is discussed. Based on the identified limiting factors some guidelines for improving the RR are also provided.

5.1.5.4 CQ 4: Providing Insights to the Product Manager about the Limiting Factors

One of the main use cases of the proposed RELREA approach and the tool from product management perspective is gaining insights about the factors (RR attributes) which are limiting the overall RR. In particular, when it is identified that the projected RR is below the requested RR, knowledge about the limiting factors can help the product manager to make changes in the process so that the requested RR is achieved within the pre-defined release date. Using the tool, to gain insights about the limiting factors, the product manager can see how the individual RR attributes impacted (i.e. positively, negatively) the overall RR over time. Degree of impact of a RR attribute $a_i$ is defined as follows:

$$Impact \ (a_i) = (\mu_{a_i}(x_t) - R(t)) \times u_i$$ \hspace{2cm} (5-2)$$

In Eq. (5-2), $\mu_{a_i}(x_t)$ is the degree of satisfaction of RR attribute $a_i$ at a particular point of time $t$, $R(t)$ is the aggregated degree of RR, and $u_i$ is the relative weight of the attribute. The tool interface for visualizing the impact of the RR attributes on overall RR for the case study project is shown in Figure 5-14. The “green” area (positive impact) of the visualization shows when and to what extent the degree of $Impact \ (a_i)$ was positive. On the other hand, the “red”
area indicates when and to what the extent degree of \( \text{Impact} \ (a_i) \) was negative. Along with the visualization, occurrence frequency (number of weeks a RR attribute negatively impacted the overall RR) of the negative impact of the RR attributes is also shown in Figure 5-15.

![Release Readiness Over Time](image)

**Figure 5-14:** Screen of the tool showing the impact of the individual RR attributes on overall RR over time
Following observation can be made in the context of the case study project from Figure 5-14 and Figure 5-15.

- From the beginning of the project (43 weeks out of 44 weeks) HP-FIR negatively impacted the overall RR and it has the highest degree of negative impact compared to the other RR attributes. Thus, we can consider the low satisfaction of HP-FIR as a potential limiting factor for being the overall RR below the requested RR. Armed with this information, the product manager can consider that focusing more on the resources and processes corresponding to this attribute would increase the overall RR.

- Continuous negative impacts of the other implementation related attributes LP-FIR (43 weeks out of 44 weeks), and IIR (39 weeks out of 44 weeks) also indicate that they are limiting the overall RR. However, their degree of negative impact is not significant due to their less relative weights. In particular, negative impact of IIR is decreasing over the last few weeks. From this, the product manager can gain insight that adding more resources on IIR would not increase the overall RR significantly. However, the product
manager can consider adding or re-allocating resources for LP-FIR with the hope of improving the overall RR since it is not showing any decreasing trend for long period.

- Significant negative impact of CCR (32 weeks out of 44 weeks) on overall RR is also observed during the last few weeks and the degree of its impact is not decreasing over time. It indicates that development team is continuously making large amount of changes in the code base. Hence, unstable code base is also a limiting factor in achieving RR. Stability of the code base can be improved by communicating the CCR goal (100 addition / deletion per day) with the development team. The idea is that it will create awareness among developers that requested RR is not possible to achieve until they stop making significant amount of changes in the source code files. As suggested in [2], management can also monitor the code check-ins and set pre-check-in authorization in the source code repository to improve satisfaction of CCR.

- Other testing related attributes such as BFR and DFR always positively impacted the overall RR during the early stage and late stage of the development; therefore, they should not be considered as limiting factor.

Overall, the above discussion shows that the proposed RELREA approach and the tool is useful in evaluating, monitoring, and identifying RR issues proactively at the early stage of the release cycle. It also helps product managers in knowing the factors which might be considered for improving the RR status.

5.1.6 Threats to Validity

The discussion of the threats to validity is important to judge the strengths and limitations of the case study results. The key threats to validity of the case study are related to the selection of the RR attributes and metrics, relative importance of the RR attributes, applied membership
functions, the aggregation operator applied, and context of the project. According to the standard classification scheme of validity threats suggested in [92], discussion is made on four types of threats to validity: construct validity, conclusion validity, internal validity, and external validity.

5.1.6.1 Construct Validity

One key threat to construct validity is that the approach only considers the RR attributes which can be evaluated objectively. Subjective RR attributes (e.g. testing depth, user experience, amount of possible enhancements, etc.) are not considered in the method and as well as in the case study. However, inclusion of subjective attributes requires both human experts and more effort to support continuous evaluation of RR. Another threat to validity specific to the case study project is that it does not follow all kind of standard software engineering processes. For instance, iteration length is too long compared to the traditional concept of iteration. As a result, it was not always possible to interpret the results according to the industry standards. However, the project was enough standard (i.e. has a solid release plan, tracked the progress of the issues, and used test driven development) to demonstrate the applicability of the proposed method and the tool.

5.1.6.2 Conclusion Validity

Conclusion validity is the degree to which conclusions we reach about the relationships about the data (or variable) are correct or reasonable. One conclusion validity of this study is the selection of RR attributes and metrics. As discussed in Section 5.1.4.1, RR attributes and metrics are defined in a goal oriented way by analyzing data available in Github¹ and Jira². In addition, the RR attributes were verified by the product manager of RP2. Around 50% of the RR attributes
found in the literature is included in the study. Therefore, threats to validity regarding selection process of RR attributes and metrics are minimized.

The accuracy of the calculated RR largely depends on the proper definition of the membership functions and relative weights of the RR attributes. As the parameters of the membership functions and relative weights are provided by the product manager, we can consider them reliable. Selection of proper aggregation operator for combining the satisfaction levels of attributes is also a threat to the conclusion validity. As a flexible aggregation operator is selected which can be adjusted by the product manager, this threat to validity is at least reduced.

5.1.6.3 Internal Validity

Internal validity issue arises when the results are drawn incorrectly. In this study, the presented results and conclusions are transparent and based on analysis of real world software project data. Therefore, there is little threat to internal validity in this case.

5.1.6.4 External Validity

Results discussed above are specific to the case study project, therefore, the external validity cannot be ensured. More industry evaluation would increase validity of the results as well as the method and tool. However, answer to the case study questions indicate that the proposed RR evaluation method and its implementation are helpful in investigating RR issues at the early stage of the release cycle.

5.2 Case Study 2: Monitoring Bottleneck Factors in Achieving RR

With this case study, the applicability of the fundamental concepts used in RELREA approach is shown for monitoring bottleneck factors (BF) in OSS domain. Motivation behind monitoring bottleneck factor is that the product management team can mitigate BFs by re-
allocating or adding resources. Necessary definitions, case study objectives, context of the case study projects, results analysis, and threats to validity are discussed below.

5.2.1 Definitions

A bottleneck in general is a factor that limits the performance of an entire system. Resource bottlenecks in project management are a well-understood phenomenon. The definitions provided below are taken from our published work in [56]:

**Definition 5-1 (Bottleneck Factor):** For a given project \( P \) and a given week \( t \), the bottleneck factor \( BF(t) \) is a RR attribute \( a_i \) that has the lowest satisfaction value \( \mu_{a_i}(x_t) \) among all RR attributes and thus limits overall RR (Eq. 3-7) value at week \( t \) [56].

\[
BF(t) = \arg\min_i \left( \mu_{a_1}(x_t), \mu_{a_2}(x_t), \ldots, \mu_{a_n}(x_t) \right)
\]  

(Eq. 5-1) returns the index of the BFs. According to the equation, multiple RR attributes can be identified as BF in a single observation. In this study, to avoid biasness, equal relative weights for RR attributes are applied.

**Definition 5-2 (Bottleneck Frequency):** For a given project \( P \) and a given time interval \([0,T]\) the bottleneck frequency of a RR attribute \( a_i \) (denoted by \( BNF(a_i,T) \)) is defined as the weekly frequency of becoming bottleneck in achieving RR within \( T \) [56].

**Definition 5-3 (Bottleneck Transition Frequency):** For a given project \( P \) and a given time interval \([0,T]\), the bottleneck transition frequency \( BNTF(a_i,a_j,T) \) is defined as the relative frequency of transitions between bottleneck factors \( a_i \) to \( a_j \) within interval \( T \) [56].
5.2.2 Case Study Objectives

The goal of this explorative case study is to understand frequencies and pattern of occurrence of bottleneck factors affecting project success by restricting the status of release readiness. To investigate these, following case study questions are defined.

- CQ 1: What are the most and what are the least frequent BFs limiting RR?
- CQ 2: Do release phases (early versus late) significantly influence the frequency of BFs?
- CQ 3: Are there any significant patterns in transition between consecutive BFs over time?

Answer to the above case study questions, helps understanding of the most frequent bottleneck factors in achieving release readiness and their likelihood of subsequent occurrence. This can be used to guide the effort required for improving release readiness.

5.2.3 Context of the Case Study Projects

In this explorative case study, data was collected for a period of seven months (or 28 weeks) from 10 OSS projects hosted in Github repository. A summary of the selected projects are shown in Table 5-7.

Table 5-7: A summary of the selected open source case study projects [56]

<table>
<thead>
<tr>
<th>Project name</th>
<th>Project domain</th>
<th>Recent major release</th>
<th>Release date</th>
<th>Number of releases observed</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1: Adobe/Brackets</td>
<td>Web</td>
<td>Sprint 30</td>
<td>28-08-13</td>
<td>8</td>
</tr>
<tr>
<td>P2: Celluloid/Celluloid</td>
<td>Desktop</td>
<td>V 0.15.0</td>
<td>06-09-13</td>
<td>2</td>
</tr>
<tr>
<td>P3: GitHub/Android</td>
<td>Mobile</td>
<td>1.0</td>
<td>09-07-12</td>
<td>6.5</td>
</tr>
<tr>
<td>P4: Imathis/Octopress</td>
<td>Web</td>
<td>V 2.0</td>
<td>29-04-12</td>
<td>0.25</td>
</tr>
<tr>
<td>P5: Intridea/Grape</td>
<td>Desktop</td>
<td>V 0.6.0</td>
<td>16-09-13</td>
<td>2</td>
</tr>
<tr>
<td>P6: Locomotive/Engine</td>
<td>Web</td>
<td>V 2.2.0</td>
<td>07-07-13</td>
<td>3.5</td>
</tr>
<tr>
<td>P7: Loopj/Android</td>
<td>Mobile</td>
<td>1.4.3</td>
<td>29-01-13</td>
<td>2</td>
</tr>
<tr>
<td>P8: Publify/Publify</td>
<td>Web</td>
<td>Publify7.0</td>
<td>15-09-13</td>
<td>2</td>
</tr>
<tr>
<td>P9: Resque/Resque</td>
<td>Desktop</td>
<td>V 1.24.0</td>
<td>21-03-13</td>
<td>3.5</td>
</tr>
<tr>
<td>P10: Ryanb/Cancan</td>
<td>Desktop</td>
<td>1.6.3</td>
<td>25-03-11</td>
<td>0.75</td>
</tr>
</tbody>
</table>
5.2.4 Defining RR attributes, Metrics and Membership Function

According to the RELREA approach, eights RR attributes and corresponding objective metrics were defined according to the Goal-Question-Metric (GQM) [21] paradigm from the two important RR dimensions (implementation of functionality, and testing scope and status). In absence of domain experts, *piecewise linear membership function* was used to describe the degree of satisfaction of the RR attributes. Parameters of the *membership functions* were defined with respect to *minimum* and *maximum* values in corresponding projects within the observation period. Details regarding these RR metrics along with corresponding RR attributes, goals, questions and acronyms are listed in Table 5-8.

**Table 5-8: Determining RR metrics using GQM approach, corresponding goals, RR attributes, questions and acronyms [56].**

<table>
<thead>
<tr>
<th>Goals</th>
<th>Refined Goals (RR Attributes)</th>
<th>Questions</th>
<th>RR Metric Definitions</th>
<th>Acronyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Assessing satisfaction of Implementation</td>
<td>Satisfaction of feature completion</td>
<td>To what extent requested features are completed?</td>
<td>Feature completion ratio</td>
<td>FCR</td>
</tr>
<tr>
<td></td>
<td>Satisfaction of features implemented</td>
<td>Is the implementation activity increasing implemented features?</td>
<td>Number of features implemented</td>
<td>FI</td>
</tr>
<tr>
<td></td>
<td>Satisfaction of build trends</td>
<td>To what extent performed builds are successful?</td>
<td>Percentage of successful builds</td>
<td>BSR</td>
</tr>
<tr>
<td></td>
<td>Satisfaction of implementation effort</td>
<td>Is the implementation activity increasing code churn per day?</td>
<td>Number of LOC per day</td>
<td>CCR</td>
</tr>
<tr>
<td></td>
<td>Satisfaction of change completion</td>
<td>To what extent requested changes are completed?</td>
<td>Change request implementation ratio</td>
<td>CR</td>
</tr>
</tbody>
</table>

Assessing satisfaction of Testing

| Satisfaction of defect finding | Is the testing activity reducing the defects? | Defect find rate | DFR |
| Satisfaction of bug fixing | To what extent detected bugs are | Bug fixed ratio | BFR |
Goals | Refined Goals (RR Attributes) | Questions | RR Metric Definitions | Acronyms
--- | --- | --- | --- | ---
Satisfaction of pull request completion | To what extent pull-requests are completed? | Pull request completion ratio | PCR

5.2.5 Analysis of the Results

Results of the case study are discussed in the following sub-sections according to the questions defined in Section 5.2.2.

5.2.5.1 CQ 1: Frequency of Occurrence of BF

A Pareto chart showing RR attributes on the X-axis and total Bottleneck Frequency (BNF) values and their cumulative percentage respectively on the primary and secondary Y-axis is shown in Figure 5-16.

![Pareto chart](image)

**Figure 5-16**: A Pareto chart illustrating BNF of the studied RR attributes [56]
Key findings from the chart are:

- FCR, BFR, and FI are the three most frequent BFs.
- CCR is the least frequent BF.
- The three most frequent BFs are responsible for 60% of total observed bottleneck occurrences.

5.2.5.2 CQ 2: Do Release Phases Significantly Influence the Frequency of BFs?

Going beyond just the frequencies of occurrence, we investigated whether certain BFs are more typical at certain stages of the process. For that, all releases were subdivided into two equal phases (called First and Second phase) with respect to release duration. Table 5-9 shows the mean BNF for weeks in first and second phases of ten projects with respect to RR attributes.

<table>
<thead>
<tr>
<th></th>
<th>FCR</th>
<th>BFR</th>
<th>FI</th>
<th>PCR</th>
<th>DFR</th>
<th>BSR</th>
<th>CR</th>
<th>CCR</th>
</tr>
</thead>
<tbody>
<tr>
<td>First</td>
<td>4.2</td>
<td>3.9</td>
<td>4.2</td>
<td>4.4</td>
<td>1.8</td>
<td>1.2</td>
<td>1.3</td>
<td>0.8</td>
</tr>
<tr>
<td>Second</td>
<td>5.3</td>
<td>4.5</td>
<td>3.6</td>
<td>1.6</td>
<td>1.8</td>
<td>2.3</td>
<td>1.9</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Due to unknown BNF’s distribution in First and Second phases, we applied two-tailed Mann-Whitney U-test to test the following null hypothesis.

**Null hypothesis:** There are no significant differences in BNFs between first and second phase.

Table 5-10 shows the initial test results. Columns in Table 4 respectively presents the RR attributes, the degree of freedom (df), the U-value of the study, the critical value (at 5% significance level) that the U-value must not exceed to be statistically significant, and the associated p-value.
Table 5-10: Results of two-tailed Mann-Whitney U-test in analyzing differences in BNF between first and second phase

<table>
<thead>
<tr>
<th>RR Att.</th>
<th>df</th>
<th>U-Value</th>
<th>Critical Value</th>
<th>p-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FI</td>
<td>1</td>
<td>46.0</td>
<td>23.0</td>
<td>.759</td>
</tr>
<tr>
<td>FCR</td>
<td>1</td>
<td>44.5</td>
<td>23.0</td>
<td>.675</td>
</tr>
<tr>
<td>BFR</td>
<td>1</td>
<td>47.5</td>
<td>23.0</td>
<td>.849</td>
</tr>
<tr>
<td>PCR</td>
<td>1</td>
<td>22.0</td>
<td>23.0</td>
<td>.032</td>
</tr>
</tbody>
</table>

Examining columns *U-value, Critical value and p-value* of Table 5-10 shows that the effect of release phases was significant for RR attribute *PCR* at significance level 0.05. Average BNF of *PCR* was significantly higher in *First* phase (Mean=4.4, SD=3.31) than in *Second* phase (Mean=1.6, SD=1.65). The results for *FI, FCR, and BFR* show a difference without statistical significance. Overall, following conclusion is made:

- The null hypothesis is rejected for Pull request Completion Rate (PCR).
- There is not enough evidence to reject the null hypothesis for all other RR attributes.

5.2.5.3 CQ 3: Are there any Significant Patterns in Transition between Consecutive BFs?

Figure 5-17 illustrates transitions between consecutive BFs for $T = 28$ weeks. Any transition $BNTF(a_i, a_j, T)$ between RR attributes $a_i, a_j$ is presented as directed arc. Solid line edges represent BNTF above a specified threshold of relative frequency. Considering 0.1 as the highest observed BNTF, we applied 0.05 as a threshold. The node “Others” represents the aggregation of the four least frequently occurring RR attributes (i.e. *DFR, BSR, CR, CCR*).
Key findings from the figure include:

- Transition from PCR to BFR has the highest BNTF (i.e. 0.055) between any two individual RR attributes.
- Three most frequent BFs (i.e. $FI$, $FCR$, $BFR$) are responsible for 50% of observed BF transitions.
- BFR has higher chances (i.e. higher aggregated BNTF) to become a bottleneck after transition from all other BFs.

5.2.6 Threats to Validity

This study is exploratory in nature. It is seen as the first step of an ongoing effort to detect, analyze and mitigate BFs in achieving RR. Since we considered only ten OSS projects, representativeness of these projects is a threat to the external validity of our observations. However, to mitigate this threat, we selected projects that conform to the four propositions on case selection by Verner et al. [93]: i) we can measure RR attributes at any time, ii) we can automatically measure the overall RR, iii) collected metrics and their collection process is clearly defined, and iv) collected metrics are relevant for answering the RQs.
In absence of domain experts, selection of RR attributes, relative weights and membership function parameters are threats to validity. To mitigate these threats, we applied the GQM approach in RR attributes selection. We considered key dimensions of RR (e.g. implementation, testing) at the goal level. The choice of the RR attributes was further determined by the availability in the Github\(^1\) repository. Selected RR attributes represent 50% of RR attributes known from the most comprehensive industry guidelines available. To reduce further bias in identifying BFs, we applied equal relative weights for all RR attributes and selected membership function parameters based on observed data.

5.3 Summary

In this chapter, applicability of the proposed RELREA approach was evaluated through two case studies. In the first case study, web based prototype tool developed for the RELREA approach was used for monitoring and analyzing RR with a real life proprietary software project. Analysis of the results showed that the tool is capable to assist the product manager in evaluating, monitoring, and investigating RR. Four types of investigations (i.e. contribution of the iterations in RR, impact of various decision strategies, projection of RR, and impact of individual RR attributes into overall RR) are provided to help product managers in making better decisions about project schedule and effort allocation (or re-allocation). The case study also showed that the classical time series forecasting methods are effective for making projection of RR.

The second case study presented in this chapter showed the applicability of the proposed RR approach with monitoring bottleneck factors in achieving RR. Analysis of occurrence of bottleneck factors across ten OSS projects revealed that certain RR attributes (e.g. FCR, BFR, and FI) are more frequently to be bottleneck compared to others. The explorative study also
showed that common patterns exist in their occurrences (e.g. eventually changing PCR to BFR) and they are influenced by certain project attributes (e.g. release phases). Knowledge about bottleneck factors helps product managers to mitigate them by re-allocating or adding resources.

Overall, the RELREA approach is helpful to product managers and project teams, be it proprietary or open-source team, in that product managers are empowered to make better decisions in term of resource allocation, management, etc. based on the transparency of data, and potential prediction of future trends and BFs.
Chapter Six: **Summary and Future Research**

In this chapter, a summary of the contributions made in this thesis towards achieving the research objec\-tives are provided. Applicability and limitations of the research facets, i.e. approach, tool, and case studies are also discussed. The chapter is concluded with a discussion of possible future research.

### 6.1 Summary of the Contributions

#### 6.1.1 RELREA Approach for Continuous Release Readiness Evaluation

This thesis presents an analytical approach called RELREA to support continuous evaluation, monitoring and projection of release readiness across the release cycle of a software product. The approach is generic and can be applied for any kind of software product management. In the approach RR attributes and their corresponding metrics are selected using the systematic GQM paradigm from the four important RR dimensions. The RR dimensions were identified by reviewing and categorizing the existing literature in the domain of RR. Defined objective metrics are used to evaluate the degree of satisfaction of the selected RR attributes with the concept of fuzzy membership functions. Overall RR is evaluated by aggregating the degree of satisfaction of the individual RR attributes with the nonlinear OWA operators. OWA operator allows the product manager to consider both the relative importance of the RR attributes and the desired decision strategy in the aggregation process. An illustrative example with real life software project data is presented to provide a deeper understanding of the proposed approach. As the approach provides a formal definition of RR, it can be applied for many kinds of analysis such as studying trade-off among the RR attributes, monitoring frequencies and patterns of bottleneck factors in achieving RR, projection of RR at the release date, evaluating the degree of impact of individual attributes on the overall RR, release to release
comparison of a software product, etc. With these analyses the product manager can identify the release related issues at the early stage of the release cycle. Therefore, corrective steps can be taken to avoid delays in releasing software in the competitive business world.

6.1.2 Studying Projection of RR with Time Series Projection Method

Based on the developed objective definition of RR, the projection of RR with the classical time series projection methods is studied. Robust and easy-to-use Holts Liner Trend (HLT) method and Holt-Winter (HW) method are used for making projection of RR at the predefined release date. After identifying the pattern of the RR time series, either HLT or HW method can be applied. These methods consider the current level, trend, and seasonality of the RR time series to determine the future RR states. The accuracy of the HLT projection method is evaluated with the RR time series (calculated using proposed approach) of a real life proprietary software project. The high MAPE value of the studied RR time series indicated that the HLT projection method is effective in RR analysis. Projection of RR allows the product manager making proactive decisions regarding RR improvement. Product manager can also set customer expectation and release window of the software products based on the projected RR score.

6.1.3 Implementation and Evaluation of RR analysis Tool for the Product Managers

A comprehensive RR analysis tool is designed and implemented for the product managers. The tool automates the processes such as data collection, defining RR attributes, defining membership functions, OWA weights calculation, RR computation and analysis. The tool is developed as a web application using Ruby on Rails framework. It can be used for evaluating, projecting and analyzing RR for the software projects which use Github¹ as source code repository and Jira² for managing and tracking development activities. The tool is
developed to help the product manager with automated, fast and continuous release readiness evaluation and analysis.

The tool was evaluated through a case study of a real life software project. Through the case study we demonstrated the capability of the RR approach described in this thesis to assist the product manager with four different types of investigations (i.e. contribution of the iterations in RR, impact of various decision strategies, projection of RR, and impact of individual RR attributes into overall RR). The tool allows the product manager to make data centric proactive decisions regarding releasing software product. Informal and intuition based product release decision may lead to the failure of the complete product.

6.1.4 Empirical Evaluation of Bottleneck Factors across Ten OSS Projects

Case study was performed on ten open source projects hosted in Github to provide empirical evidence of occurrence frequencies and patterns of bottleneck factors in achieving release readiness. Results of the explorative study revealed that certain RR attributes (i.e. FCR, BFR, and FI) are more prone to be bottleneck compared to others. The study also showed that common patterns exist in their occurrences (i.e. eventually changing BF to BFR) and they are influenced by certain project attributes (i.e. release phases). Knowledge about bottleneck factors helps the product manager to mitigate them by re-allocating or adding resources.

6.2 Limitations and Applicability

The RR evaluation approach presented in this thesis helps the product manager to better understand the current and future state of readiness of the software products as well as the limiting factors. The approach and the case studies described are explorative in nature. How to improve the current RR is not studied in this research. As discussed in the literature review, this is one of the least studied areas of software engineering and little is known about its industry
validation. Therefore, the RR approach presented here is meant to be a good starting point in this direction.

Some of the practical issues of applying the proposed approach are identifying the right set of RR attributes, defining relative weights of the RR attributes, defining and tuning the parameters of the membership functions. A bad selection of these parameters may lead to wrong readiness projection, which in turn lead to bad release decisions. However, these parameters are very much context specific and cannot be generalized for software products. The goal oriented definition of RR attributes and selection of parameters of the membership function based on the past releases or similar past projects data would reduce this risk. In the implemented tool, observed minimum and maximum values of the metrics of the past and current releases are provided as meant to assist the product manager defining parameters of the membership functions.

Another issue is that the proposed approach is suitable for the projects, which use state-of-the-art project management tools for managing and tracking the development activities. And finally, the approach is more applicable in the context of iterative development where contribution of the iterations in achieving RR can be studied for better planning the next iteration.

6.3 Future Research Scope

The continuous RR evaluation approach and the tool described in this thesis can be further improved. Some of the future research directions are:

1. Evaluation of the proposed approach with more real world industry and OSS projects would give us better understanding of the capability of the approach towards assisting product managers.
2. Analysis of the robustness of the results in dependence on the varying weights of RR attributes and the defined membership functions. Investigation of the impact of these parameters on RR at different stages (i.e. early, late) of the release cycle is also an important future research direction.

3. The tool described can be improved by integrating more existing tools in the domain of continuous integration, source code analysis, testing, etc. In addition, inclusion of capabilities such as analysis on likelihood of achieving RR on the release date, release to release comparison, product to product comparison, drill-down to the artifacts related to RR attributes would increase its acceptability to the product managers.

4. The empirical study of monitoring and identifying BF described in this thesis can be extended across project domains (i.e. desktop vs. web application) and with regards to project-specific characteristics such as project size, number of contributors, release development phase, etc. Results of that study would help software organizations systematically learn about the most critical RR attributes from related project experiences. Definition of the BF can also be improved by including the past behaviour of the RR attributes.

5. Development of a recommendation system that can re-allocate effort to reduce the gap between expected and actual RR. Such a recommendation system should include identification of RR improvement factor and budget allocation strategy for maximum improvement of RR for the upcoming time interval. Proactive analysis on possible RR improvement can help product manager’s avoiding unexpected delays in releasing software.
References


Appendices A: Clarification of the Contributions in the Second Case Study

The second case study titled “Monitoring Bottleneck Factors in Achieving RR” (Section 5.2) is a collaborative work which was performed as a part of the evaluation of the methodology presented in this thesis. The cases study results were published as a paper [56] where I am the second author. My contributions in the paper are as follows:

- The methodology (RELREA, Section 3.1.6) and the formulation of the release readiness used in the case study is designed and developed by me. As described in Section 6.1.1, the proposed methodology is one of the key contributions of this thesis.

- One of the important aspects of the study is selecting the case study projects and defining their release readiness attributes. I participated in the searching and selection of the case study projects by analyzing the large set of open source projects hosted in Github. The guideline used for defining the release readiness attributes from the important readiness dimensions (i.e. implementation status, and testing scope and status) was also developed by me. The guideline is described in Section 2.3.4 of the thesis.

- I also partially contributed in the data collection, results analysis, writing, and reviewing of the paper.

The undersigned certify that the first author of the paper have read and recommend the above contributions of the second author of the paper.

S M Didar Al Alam
Dept. of CPSC
University of Calgary
Calgary, Alberta, Canada
smdalam@ucalgary.ca