A Recommendation System for Planning Software Releases

by

Jamshaid G. Mohebzada

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

CALGARY, ALBERTA

September, 2012

© Jamshaid G. Mohebzada 2012
Abstract

Strategic release planning is a critical step in iterative software development. It involves assignment of requirements to subsequent releases in consideration of constraints and stakeholder demands. Manually analyzing release planning projects is challenging since large volumes of data are involved, and release planning models are dependent on several input parameters and complex algorithms. A recommendation system, called SRP-Plugin 2.0, is presented in this thesis to assist product managers with better release decisions. First, literature is reviewed systematically for related recommendation systems (contribution 1). SRP-Plugin 2.0 is realized using four techniques. Simulation and sensitivity analysis are utilized to determine the important input parameters (contribution 2). Machine learning is used for predicting the impacts on the release plans due to project changes (contribution 3). A recommendation method assists with achieving certain release planning targets (contribution 4). Finally, SPR-Plugin 2.0 is implemented as a plug-in for an integrated development environment (contribution 5).
Publications

Some of the materials, ideas, tables, and figures in this thesis have appeared previously in the following publications.

Refereed Full Papers:


Technical Report:

Acknowledgements

I would like to thank my supervisors, Dr. Guenther Ruhe and Dr. Armin Eberlein, for their supervision, continuous support, and guidance throughout this work. I attribute much of the success of this work to their valuable feedback, positive criticism, and constructive advice.

I would also like to express my gratitude to Dr. Michael Richter for his guidance and valuable comments during the early phase of this work.

Many thanks go to all members of the Software Engineering Decision Support (SEDS) laboratory for their comments and insightful discussions during my research.

I am also grateful to the examining committee members, Dr. Victoria Mitchell, Dr. Mahmood Moussavi, and Dr. Behrouz Far, for reviewing my thesis and providing useful comments.

Finally, I would like to express my gratitude to my family for being a source of support, encouragement and motivation throughout my research.
This thesis is dedicated to my parents, who have always been there for me, and have supported me in every endeavor.
# Table of Contents

Abstract ............................................................................................................................... ii  
Publications ........................................................................................................................ iii  
Acknowledgements .......................................................................................................... iv  
Table of Contents ............................................................................................................... vi  
List of Tables ..................................................................................................................... ix  
List of Figures ...................................................................................................................... x  
List of Abbreviations ........................................................................................................ xii  

## CHAPTER ONE: INTRODUCTION

1.1 Motivation .................................................................................................................. 1  
1.2 Solution Approach and Background .......................................................................... 6  
1.3 Research Objectives ................................................................................................. 10  
1.4 Thesis Structure ....................................................................................................... 11  

## CHAPTER TWO: LITERATURE REVIEW

2.1 Introduction to Systematic Mapping of RSREs ....................................................... 14  
2.2 Methodology ............................................................................................................ 16  
   2.2.1 Search Strategy ................................................................................................ 16  
   2.2.2 Inclusion and Exclusion Criteria ..................................................................... 17  
   2.2.3 Search Results ................................................................................................. 18  
   2.2.4 Classification Scheme ..................................................................................... 19  
2.3 Mappings ................................................................................................................. 20  
   2.3.1 Map A: Overview of RSREs and Their State of Validation ........................... 22  
   2.3.2 Map B: Characteristics of the RSREs ............................................................. 24  
      2.3.2.1 Recommendation Technique ................................................................. 26  
      2.3.2.2 Type of Recommended Items ................................................................. 28  
      2.3.2.3 Output Form ........................................................................................... 30  
      2.3.2.4 Cross-Dimensional Features .................................................................. 31  
      2.3.2.5 Architecture ........................................................................................... 32  
2.4 Discussion ................................................................................................................ 32  
2.5 Summary .................................................................................................................. 35  

## CHAPTER THREE: RPSIM II – SIMULATION AND SENSITIVITY ANALYSIS

METHOD FOR RELEASE PLANNING ........................................................................... 37  
3.1 Objectives ................................................................................................................ 38  
3.2 Background .............................................................................................................. 39  
3.3 Overview .................................................................................................................. 40  
3.4 Input Parameters ..................................................................................................... 44  
3.5 Sampling Ranges ..................................................................................................... 45  
   3.5.1 Sampling Range of Efforts, and Capacities: .................................................... 45  
   3.5.2 Sampling Range of Stakeholder, Release, and Criteria Weights .................... 45  
3.6 Simulation Procedure .............................................................................................. 46
6.1 Introduction ..............................................................................................................96
6.2 Development Platform .............................................................................................98
6.3 System Architecture of SRP-Plugin 2.0 .................................................................100
  6.3.1 ReleasePlanner™ Interface ...........................................................................101
  6.3.2 Visual Studio Plugin Interface .......................................................................101
  6.3.3 RPSim II Simulation Module ........................................................................102
  6.3.4 Sensitivity Recommender Module .................................................................103
  6.3.5 Change Impact Recommender Module .........................................................104
  6.3.6 Target Achievement Recommender Module .................................................104
  6.3.7 Recommendation Engine ...............................................................................105
6.4 Summary ................................................................................................................105

CHAPTER SEVEN: CASE STUDY ...............................................................................107
  7.1 Case Study Design .................................................................................................107
  7.2 Case Description ....................................................................................................108
  7.3 Data Generation and Results ..................................................................................110
  7.4 Analysis Results .....................................................................................................112
    7.4.1 Rec. 1: Sensitivity of the Input parameters ...................................................112
    7.4.2 Rec. 2: Trends in Output Metrics Due to Input Parameter Changes ..........118
    7.4.3 Rec. 3: Predicted Impact of Release Project Changes .................................122
    7.4.4 Rec. 4: Achieving Release Plan Targets .......................................................128
  7.5 Threats to Validity .................................................................................................131
  7.6 Summary ................................................................................................................131

CHAPTER EIGHT: SUMMARY AND FUTURE RESEARCH .................................133
  8.1 Contributions and Summary ..................................................................................133
    8.1.1 C1: Comprehensive Systematic Review and Mapping of RSREs ................133
    8.1.2 C2: Simulation and Sensitivity Analysis Method for SRP .........................133
    8.1.3 C3: Recommendation Method for Predicting Impact of Changes on the Release Plans ..................................................................................................134
    8.1.4 C4: Recommendation Method for Supporting Release Plan Target Achievement ..................................................................................................135
    8.1.5 C5: Implementation and Evaluation of Recommendation Tool for Product Managers ..................................................................................................136
  8.2 Limitations and Applicability ................................................................................136
  8.3 Future Research .....................................................................................................137

REFERENCES ................................................................................................................139
List of Tables

Table 2-1: Databases searched for literature review ......................................................... 16
Table 2-2: Search terms used for systematic review of literature ..................................... 17
Table 2-3: Detailed inclusion and exclusion criteria of studies ........................................ 18
Table 2-4: Studies selected during the systematic review [15] ......................................... 21
Table 2-5: RSRE validation description [15] .................................................................... 23
Table 3-1. Definition of input parameters considered in RPSim II method ..................... 44
Table 3-2: Project properties for illustrative example ...................................................... 61
Table 3-3: Illustrative example of RPSim II simulation data ........................................... 64
Table 3-4: Input and output change metrics of illustrative simulation data ..................... 66
Table 4-1: Average accuracy of learning techniques used in pilot case study ................. 74
Table 4-2: Training data format ........................................................................................ 75
Table 4-3: Training data attributes .................................................................................... 76
Table 4-4: Training data labels ......................................................................................... 76
Table 5-1: Release planning targets .................................................................................. 86
Table 5-2: Ranking of scenarios based on their utility ..................................................... 92
Table 7-1. Selected input parameters and their sampling ranges .................................... 110
List of Figures

Figure 1-1: An example release planning problem solved by EVOLVE II in ReleasePlanner™. Five release alternatives are shown, and each assigns a requirement to either release 1, 2, or 3 (postponed)................................................................. 5

Figure 2-1: The study selection process [15]........................................................................ 19

Figure 2-2: Map A: Mapping of studies with respect to the RE activity addressed, year of publication, and validation type [15]. ........................................................................ 25

Figure 2-3: Map B: Combined mapping of the RE activity, recommendation technique, and type of items recommended [15]. ........................................................................ 25

Figure 2-4: RSRE validation types [15]................................................................................. 33

Figure 3-1: RPSim II Overview .......................................................................................... 43

Figure 3-2: Algorithm for the RPSim II's simulation procedure ........................................ 46

Figure 3-3: Procedure of simulation using OAT-SA approach ........................................... 48

Figure 3-4: Procedure of simulation using AT-SA approach in RPSim II ......................... 49

Figure 3-5: Stakeholder Excitement Evaluation from ReleasePlanner™............................ 57

Figure 3-6: Algorithm for calculating sensitivity indices ..................................................... 60

Figure 6-1: SRP-Plugin architecture in context of Visual Studio 2010 and ReleasePlanner™ ...................................................................................................................... 99

Figure 6-2: System architecture of SRP-Plugin 2.0 ............................................................ 102

Figure 7-1: Screen of the Planning & Re-Planning module .............................................. 109

Figure 7-2: Screen of the simulation tab of the tool ......................................................... 111

Figure 7-3: Screen of the sensitivity recommender tab of the tool ................................. 113

Figure 7-4: Plot of the 10 most and least sensitive input parameters ......................... 114

Figure 7-5: Sensitivity of the effort estimate parameters from different output perspectives ............................................................... 119

Figure 7-6: Screen of the sensitivity recommender tool showing trends ....................... 121
Figure 7-7: Scatter plot of change in PF metric versus change in resource capacity value ........................................................................................................................ 122

Figure 7-8: Prediction accuracy of SVM classifiers ........................................... 124

Figure 7-9: Screen of the change impact recommender tab - systematic ............ 125

Figure 7-10: Screen of the change impact recommender tab - manual ............... 126

Figure 7-11: Screen of the target achievement recommender tab ..................... 130
<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRP</td>
<td>Strategic Release Planning</td>
</tr>
<tr>
<td>RS</td>
<td>Recommendation System</td>
</tr>
<tr>
<td>RE</td>
<td>Requirements Engineering</td>
</tr>
<tr>
<td>RSRE</td>
<td>A Recommendation System for Requirements Engineering</td>
</tr>
<tr>
<td>RPSim II</td>
<td>Simulation and Sensitivity Analysis Method for Release Planning</td>
</tr>
<tr>
<td>OAT-SA</td>
<td>One-at-a-time Sensitivity Analysis</td>
</tr>
<tr>
<td>AT-SA</td>
<td>All Together Sensitivity Analysis</td>
</tr>
<tr>
<td>TDF</td>
<td>Triangular Distribution Function</td>
</tr>
<tr>
<td>PS</td>
<td>Release Plan Structure metric</td>
</tr>
<tr>
<td>LED</td>
<td>Levenshtein Edit Distance</td>
</tr>
<tr>
<td>SFP</td>
<td>Stakeholder Feature Points metric</td>
</tr>
<tr>
<td>PF</td>
<td>Number of postponed features metric</td>
</tr>
<tr>
<td>EA</td>
<td>Stakeholder Excitement Analysis metric</td>
</tr>
<tr>
<td>SI</td>
<td>Sensitivity index of an input parameter</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function</td>
</tr>
<tr>
<td>DSS</td>
<td>Decision Support System</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated Development Environment</td>
</tr>
<tr>
<td>VS</td>
<td>Microsoft Visual Studio</td>
</tr>
<tr>
<td>TFS</td>
<td>Team Foundation Server</td>
</tr>
<tr>
<td>VSTE</td>
<td>Visual Studio Team Explorer</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ALM</td>
<td>Application Lifecycle Management</td>
</tr>
<tr>
<td>CMMI</td>
<td>Capability Maturity Model Integration</td>
</tr>
</tbody>
</table>
1.1 Motivation

The size and complexity of software systems are increasing. Organizations are increasingly relying on software to add new or enhanced functionalities to their products to have a competitive edge. Systems need to be changed frequently to keep up with the competition and meet the end users’ needs [1]. Delivering the next release of such software systems on-time, with the right set of features is a challenging task. Proper planning starting from the initial phases of software engineering is important, and often determines the success or failure of a product. As a result, important decisions in product management involve features and their release planning. Having access to relevant project data helps with better release decision making. A survey conducted among 110 managers and developers at Microsoft to determine their information needs shows that the most influencing factor in their decision making is data and related metrics, and the top three important project artefacts to measure are features, components, and complete product artefacts [2].

Strategic Release Planning (SRP) involves the assignment of requirements or features to consecutive releases while considering hard and soft constraints, such as time, effort, quality or resources [3]. SRP usually occurs during the initial phases of the requirements engineering process (RE), and throughout the evolution of the software (re-planning). There are two main approaches to release planning according to Ruhe and Saliu [4]: manual (art of release planning) and hybrid approach (science of release planning). The manual approaches to release planning rely on human judgement alone to
decide which features go into which release, as in agile development. This is suitable when deciding which features from a small pool of candidate features should go into the next iteration, and few stakeholders make the decision informally through physical meetings and negotiations. But as the number of features and stakeholders grow as well as the planning scope (e.g. the next 5 releases), it becomes difficult to rely on manual approaches as the possible number of solutions to the release problem increases exponentially [5]. This issue is tackled in the hybrid release planning approach, which relies on both human as well as computational intelligence to systematically generate the best solutions to a release planning problem. Over the years, many formal models have been developed that approach the complex task of strategic release planning in a systematic manner as described by Svanhberg et al. in their survey paper [6]. A formal SRP model formulates the planning problem as an optimization problem with a large solution space, and then finds the best possible solution. The approach taken by formal SRP models is suitable for both large as well as small release planning problems [3].

The authors of the comprehensive review [6] of formal SRP models report that the EVOLVE family of SRP models [7], [4], [3] are the most prominent ones considering the depth to which the problem is addressed by these models. EVOLVE II [3] is latest of them and has been implemented in a web-based tool called ReleasePlanner™ 1. For better understanding of such a formal SRP model, the main steps of EVOLVE II model are summarized below. The detailed description is discussed in [3].

1 http://www.releaseplanner.com/rpApp/
Step 1 - Project Data Input: In this step key parameters of the planning problem are determined, by the product manager (PM) and stakeholders, based on the strategy and goals of the company. These parameters are entered into the tool implementing the EVOLVE II model. The parameters consist of the following.

- Scope of planning in number of releases, and a weight assigned to each release based on their importance.
- The planning criteria, i.e. perspectives from which the planning is to be carried such as risk, value. Each criterion is also assigned a weight.
- The resource types, and the capacity of each resource available for each release. This is determined by the product manager based on which resources are available to be utilized for a project, and the resource types could be person-hours, dollar amounts, or any other resource need for the project.
- Planning objects (features or requirements), and their effort estimates in terms of the resources specified in previous step.
- Project constraints or dependencies between the planning objects.
- Stakeholders and weight assigned to each stakeholder.

Step 2 - Prioritization of planning objects: In this step the PM and each stakeholder assigns a score (from 0 to 9) to each planning object from the perspective of one or more planning criteria.

Step 3 - Generation of release plan alternatives: The release planning problem is formulated as an optimization problem, with the objective being assignment of features scored highly by the important stakeholders to as early release as possible. The problem
is solved by applying algorithms involving branch-and-bound and specialized integer programming procedures presented in [7]. The solution consists of up to five diverse release plan alternatives, each specifying the assignment (or postponement) of each of the planning objects to one of the releases, as demonstrated in Figure 1-1.

**Step 4 – Analysis of alternatives and selection of a plan:** Finally, the product manager analyzes the alternative release plans to select the one that is considered most useful for the situation. The analysis involves comparing the plans in terms of quality, resource consumption, and the degree by which each stakeholder is expected to be satisfied from the plan. These analyses are also part of the output of the EVOLVE II model in addition to the release plan alternatives.
EVOLVE II is a sound and rigorous formal strategic release planning model applicable for both small as well as large problems (with more than 1000 planning objects). However, we can see from the steps above that such a model is based on a large number of input parameters with uncertain and often inaccurate values, such as effort estimates, and weights assigned to stakeholders, releases, and criteria. This makes preparing the ‘right’ input for the model difficult since the uncertainty is unavoidable. Methods are needed to assist product managers to ensure the input parameter values fed to the SRP model reflect the reality as much as possible, and assist in mitigating the uncertain nature of project data.

![Figure 1-1: An example release planning problem solved by EVOLVE II in ReleasePlanner™. Five release alternatives are shown, and each assigns a requirement to either release 1, 2, or 3 (postponed).](image)
Also, due to the nature of the problem complex optimization algorithms need to be used. As a result, analyzing the input-output relationships is not straightforward task. A change to a certain input parameter might have different effects in different projects and the impact of each input parameter on the output is not clear from the model itself. Preparing the right input for the model and the subsequent analysis of the outputs involve large amounts of data. These issues affect the adoption of the models in industry, and acceptance of the generated releases by the product managers.

Managers often lack data analysis skills or do not have the time for it [2], making extraction of useful information from often large release planning data a challenge. In order to assist product managers with better release decision making, recommendation tools and methods are needed that complement the functionality of SRP models with automatic release data analysis.

1.2 Solution Approach and Background

To address these needs, this thesis presents a recommendation system (RS) for assisting release planners at the strategic level. The specific recommendations given are discussed later in this section. The proposed RS integrates with, and complements the functionalities of the EVOLVE II strategic release planning model. Simulation, sensitivity analysis, data mining, and machine learning techniques are used to extract useful recommendations from a release planning project’s data. In contrast to a manual project analysis, the system would be able to provide concise information to the product manager quickly and proactively.
A recommendation system builds upon a model of existing data and expert knowledge, from which potentially useful information is derived for a new situation [8]. RSs have been widely used in e-commerce to provide users with personalized product, content or service recommendations. Amazon’s² product recommendation is a popular example. As a result, recommendation technologies have been researched extensively within the context of e-commerce. A RS is typically classified based on its recommendation generation mechanism. A content-based RS provides recommendations based on a user’s past preferences [9]. A collaborative RS provides recommendations based on similar users’ past preferences [10]. A hybrid RS applies both content-based and collaborative techniques [11]. A matrix factorization-based RS is a new and efficient type [12]. These systems utilize various data mining, machine learning, and search techniques. Data mining is used to analyze large data sets to discover patterns from it. Machine learning techniques are used to learn and generalize rules by observing current data available, to be able to make predictions for future data.

A software simulation model represents a dynamic system or phenomenon of the software process. Simulation models are used to gain insight into various issues such as strategic management, process improvements, or software project management. It is a useful decision making aid, by providing a means of experimentation, prediction, and analysis of “what if” scenarios [13]. Recommendation systems analyze existing data to derive useful information for a given situation as mentioned earlier. Previous strategic

² http://www.amazon.com
release planning data usually does not exist. To overcome this, simulation is used in this work to generate the data that is input to the recommender models.

The EVOLVE II model formulates the strategic release planning process and uses algorithms to solve it as described earlier. Simulation here refers to the actual running of the EVOLVE II model, and a simulation run is an instance when the model is run to generate release plans for a given set of input parameter values. The simulation method treats the EVOLVE II model as a black-box and is only concerned with the inputs and outputs of the model. The simulation method proactively generates various release planning scenarios. A scenario refers to a variation of the original release project’s setting due to changes made to one or more of the parameter values which may result to a different release plan (or solution set) than the original plan, called the baseline plan. Systematic input parameter changes simulate the behaviour of a product manager when altering the model parameters to investigate a specific goal. Manual inspection of the impact of input parameter changes is tedious, and almost impossible in large projects. The recommendation generation methods used in this RS are based on simulation data of a particular project (provided as input), and also based on our domain knowledge of strategic release planning. Four types of recommendations are provided to the product manager as outlined below.

As discussed earlier, release planning models are based on uncertain estimates, such as estimated effort required to implement a feature of a software product. The more precise the estimates are, the better the release plans and decision making will be. Sensitivity analysis lets us identify those parameters of a model that have the most impact
on the model’s output [14]. If we determine the input parameters that are the most sensitive, we can take proactive steps to mitigate their uncertainty. For example; find more precise effort estimates for the most sensitive requirements, re-assess the capacity of most sensitive resource capacities, or re-evaluate the weights assigned to ‘sensitive’ stakeholders. In this RS, two types of recommendations are provided for this situation. First, the data generated from simulation is analyzed to recommend the sensitivity ranking of each input parameter of the release planning project. Second, the RS allows the PM to move on from concise sensitivity rankings, to drill-down into detailed sensitivity analysis data to observe how some pre-defined release plan metrics are changing (trending) due to a series of input variations. Such trend analysis helps in gaining more insight about the project, and detection of non-obvious input-output relations.

Release project data need to be changed frequently in light of new information, for instance the available resource capacities might need to be decreased. Changing the project data right away might have unforeseen consequences on the release plans. For instance, some stakeholders might get very disappointed, or more features would get delayed. It would be useful for the product manager to know what will happen to the release plans, if some inputs $x_1...x_n$ are changed. For this, we use machine learning to learn the impact of input changes from the simulation data, and provide a product manager with concise and quantitative release plan impact recommendations estimated from the trained prediction model. This is a quick, concise, and automatic method to help
product managers perform pro-active contingency planning, and better understand the effects of release changes.

The fourth recommendation type provided by the RS described in this thesis assists the product manager in achieving certain release targets. For example, the product manager would like to improve satisfaction of a stakeholder, or would like to reduce the number of features getting postponed according to the baseline plan. More than one target, often conflicting, can be specified at a time. The approach used for this method is to analyze the release planning scenarios generated during the sensitivity and simulation analysis, by ranking the scenarios based on how much it helps in achieving the specified targets or, in other words, how much utility each release scenario in the data has for the PM from the perspective of target achievement. The utility score also depends on the trade-offs and magnitude of changes that needs to be made in order to achieve the targets. The trade-offs could include adding more resource capacity or dissatisfaction of some stakeholders, or postponement of some features.

The recommendation system described here uses EVOLVE II release planning model since this model is the most comprehensive of the available SRP models [6], and comes with a tool support. However, the concepts are generic and any other SRP model can be plugged-in into the recommendation system with little modifications.

1.3 Research Objectives

The main goal of this thesis is to design and develop a recommendation system to assist product managers in the task of strategic release planning. To achieve this goal, the research objectives have been set to the following.
• **RO1**: Systematic review of literature for recommendation systems for requirements engineering, and analysis and mapping of the review results.
  
  - **RO1.1**: Identification of existing recommendation systems for requirements engineering and the RE activities they focus on.
  
  - **RO1.2**: Analysis of the state of validation of the recommendation systems found in RO1.1.
  
  - **RO1.3**: Analysis of the characteristics of the recommendation systems found in RO1.1.

• **RO2**: Design of a simulation and sensitivity analysis method for release planning. Utilizing the sensitivity analysis results for generation of two types of recommendations; 1) sensitivity rankings of the input parameters of the release planning project, and 2) trends in defined release plan metrics due to changes in the release project.

• **RO3**: Design of a recommendation method that predicts the impact of changes, made to the project, on the release plans.

• **RO4**: Design of a utility-based recommendation method which assists a product manager with achieving strategic release planning targets.

• **RO5**: Design, implementation, and evaluation of a recommendation tool which integrates the simulation and sensitivity analysis method from RO2 with the recommendation methods from RO3 and RO4.

1.4 Thesis Structure

The thesis is structured as follows.
- Chapter 1 introduces the thesis, discusses the motivation, explains the background, and provides an overview of the solution approach.

- Chapter 2 provides a review of the literature in the area of recommendation systems for requirements engineering. The review methodology is described, the studies are mapped from different perspectives, an analysis of the results is provided.

- Chapter 3 describes the design of the simulation and sensitivity analysis method for strategic release planning, including the metrics devised for measuring the input and output changes of the SRP Model, and the steps for sensitivity computations. Data generated with this method is further analyzed in Chapters 4 and 5 for recommendation generations.

- Chapter 4 describes the design of the method that recommends the impact of changing input(s) of the SRP model on the release plans. It also discusses the background of the learning technique.

- Chapter 5 describes the design of a method that informs a product manager on what changes need to be done to the baseline plan in order to achieve certain release planning targets, and what are the trade-offs of the changes recommended.

- Chapter 6 describes the development of an integrated and comprehensive recommendation tool that is based on the implementation of the recommendation methods described in Chapters 3, 4, and 5.
• Chapter 7 evaluates the developed recommender tool through a case study of a real life release planning project. Demonstrations of the tool for assisting the product manager in various release planning tasks are also provided in this chapter.

• Chapter 8 concludes the thesis, discusses the main contributions, applicability, and proposes future research to improve the work.
Chapter Two: Literature Review

In this chapter, we use systematic mapping to provide an overview of recommendation systems for the requirements engineering (RSRE) process, their characteristics, and state of validation. Since strategic release planning is done during the initial phases of requirements engineering, we also investigated the recommender systems for release planning. The resulting mappings of the literature are analyzed to provide conclusions and to identify the limitations of current studies, and to suggest future research areas. The results reaffirm the need for a recommendation system such as the one described in this thesis that helps product managers with better release decision making.

The literature review presented in this chapter has been adopted from our published work in [15], and addresses the first research objective of the thesis described in section 1.3.

2.1 Introduction to Systematic Mapping of RSREs

Requirements engineering is an initial and critical phase of software engineering. Requirements engineers have to deal with a large information space [16]. It is also important to note that the specification of requirements is not a one-time task but evolves over time to reflect the realities of the project. This makes the task even more challenging. Incomplete or incorrect requirements are one of the main causes of project failures [17]. Systems and methods that could support requirement engineers during this task are clearly needed.
Robillard, Walker, and Zimmermann [18] define a recommendation system for software engineering as:

"A software application that provides information items estimated to be valuable for a software engineering task in a given context."

In this review, we adopt this definition for a recommendation system for requirements engineering by focusing on the requirements engineering aspects of software engineering. Adomavicius and Tuzhilin provide an overview of recommendation approaches and techniques in their widely referenced paper [8].

Current research on recommendation systems in the area of software engineering [18], [19] focuses on the coding phase where recommendations are given to help the developers during various programming tasks - from suggesting code reuse opportunities [20] to finding expert consultants for a project [21]. RSREs have significant potential [22], [23], [24]. They can be used to provide likely useful information to requirements engineers in a certain context. The overall goal is to provide the right information, at the right time, to the right person. This would allow requirements engineers to spend their limited time on more important aspects of the project.

Requirements engineers and product managers need to know how recommendation systems can assist them in their tasks. The purpose of this review is to address this need by providing the results of a systematic mapping study of current RSREs and techniques, their main characteristics, and state of validation. In addition, since research on RSREs is relatively new, in this mapping study we determine the coverage of the research field, and identify research gaps in recommendation systems and
techniques suitable for requirements engineers. This review addresses the first research objective of this thesis.

2.2 Methodology

In this section, the methodology of the systematic mapping is discussed which includes search strategy, databases used, the selection criteria, and the classification scheme used for mapping the studies. The methodology is based on the software engineering systematic mapping guidelines by Petersen et al. [25]. The research questions investigated in this study addresses research objective RO1 from section 1.3.

2.2.1 Search Strategy

For this review, we used the six databases presented in Table 2-1, which are known as major sources of software engineering related studies. The search terms used to find relevant studies are presented in Table 2-2. Note that terms a and b were not used separately but in conjunction with other terms. This is to limit the search to the requirements engineering area. A search log was used to keep track of all the selected or rejected studies found during the course of the search for relevant studies. This is necessary to have a transparent and repeatable search process.

<table>
<thead>
<tr>
<th>Database Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACM Digital Library</td>
</tr>
<tr>
<td>IEEE Xplore</td>
</tr>
<tr>
<td>Science Direct</td>
</tr>
<tr>
<td>Springer Link</td>
</tr>
<tr>
<td>Scopus</td>
</tr>
<tr>
<td>Engineering Village (Compendex, and Inspec)</td>
</tr>
</tbody>
</table>

Table 2-1: Databases searched for literature review
### Table 2-2: Search terms used for systematic review of literature

<table>
<thead>
<tr>
<th>No.</th>
<th>Search Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>“requirements engineering”</td>
</tr>
<tr>
<td>b</td>
<td>“requirements management”</td>
</tr>
<tr>
<td>1</td>
<td>{a, b} AND recommender</td>
</tr>
<tr>
<td>2</td>
<td>{a, b} AND recommendation</td>
</tr>
<tr>
<td>3</td>
<td>requirements recommender</td>
</tr>
<tr>
<td>4</td>
<td>requirements recommendation</td>
</tr>
<tr>
<td>5</td>
<td>feature recommendation</td>
</tr>
<tr>
<td>6</td>
<td>recommend requirements</td>
</tr>
<tr>
<td>7</td>
<td>1 OR 2</td>
</tr>
<tr>
<td>8</td>
<td>3 OR 4 OR 5 OR 6</td>
</tr>
<tr>
<td>9</td>
<td>{1, 2} AND {system, framework, tool, plug-in, software, prototype, agent}</td>
</tr>
<tr>
<td>10</td>
<td>{1, 2} AND {model, technique, technology, approach, algorithm}</td>
</tr>
<tr>
<td>11</td>
<td>{1, 2} AND {mapping, review, analysis, validation}</td>
</tr>
</tbody>
</table>

#### 2.2.2 Inclusion and Exclusion Criteria

The selection of relevant studies was done in two phases.

In the first phase, we used a basic inclusion criterion; in the context of requirements engineering, we considered the title, keywords and abstract of the all studies found by searching the databases using the search terms. Those studies were selected that describe design and implementation of recommendation systems, methods, and studies providing an analysis of existing RS. Previous related systematic reviews and mappings studies were also to be selected in order to help us find the original work reviewed in them.
In the second phase, we narrowed down the number of selected studies by applying detailed inclusion and exclusion criteria on the studies selected in the first step. These criteria are outlined in Table 2-3. A selected paper must meet at least one inclusion criterion. We excluded papers not meeting any of the inclusion criteria, or meeting one or more of the exclusion criteria.

Table 2-3: Detailed inclusion and exclusion criteria of studies

<table>
<thead>
<tr>
<th>Detailed Inclusion Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Detailed Exclusion Criteria</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
</tbody>
</table>

2.2.3 Search Results

The study selection process involved four steps as shown in Figure 2-1 and described here: (1) Six databases were searched individually using the keywords established earlier. A total of 6,109 studies were found. (2) From the 6,109 studies, 372 were selected by reading the title only. (3) Duplicate studies were removed. The basic inclusion and exclusion criteria were applied (based on reading the title, abstract, and
keywords), resulting in 69 studies to be looked into further in the next step. (4) In this step, the detailed inclusion and exclusion criteria were applied.

This resulted in selecting the 23 peer reviewed papers listed in Table 2-4, from which two provide an overview of potentials of RSRE [22], [23], and 21 studies describe a recommendation system for requirements engineering which were considered for the systematic mapping study. The papers are published between 2004 and 2011 in journals (4 papers), conferences (10), and workshops (9).

2.2.4 Classification Scheme

The studies selected for systematic mapping were classified based on the RE activity addressed, and mapped with respect to four other aspects; 1) publication year, 2) validation type, 3) recommendation technique used, and 4) recommended item type.
The RE process activities considered are based on those presented by Kotonya and Sommerville [26], and they are; elicitation, analysis, documentation, and validation. The rest of the classification scheme was determined by looking at the identified studies and deciding on the categories for each mapping aspect. For instance, from the studies we found that the categories for validation of proposed recommendation systems or techniques are; case study, experiment, and simulation (in academic or industry setting).

2.3 Mappings

Two maps were developed in this study in the form of bubble plots. The first mapping (Map A), shown in Figure 2-2, positions each study with respect to the RE activity addressed and is further mapped with respect to the publication year and validation type. The size of each bubble reflects the number of studies at each intersection of the mapping aspects. The second mapping (Map B) is shown in Figure 2-3, where each bubble demonstrates the number of studies addressing an RE activity with respect to the recommendation analysis technique, and the type of items recommended. Some studies are mapped to multiple categories, which is the reason for the unequal number of studies at each side of the maps in Figure 2-2 and Figure 2-3.
<table>
<thead>
<tr>
<th>Id</th>
<th>Year</th>
<th>Authors</th>
<th>Title</th>
<th>Ref.</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>2004</td>
<td>M. Andric, W. Hall, L. Carr</td>
<td>Assisting artifact retrieval in software engineering projects</td>
<td>[53]</td>
</tr>
<tr>
<td>S2</td>
<td>2008</td>
<td>J. Cleland-Huang, B. Mobasher</td>
<td>Using data mining and recommender systems to scale up the requirements process</td>
<td>[31]</td>
</tr>
<tr>
<td>S3</td>
<td>2008</td>
<td>C. Castro-Herrera, C. Duan, J. Cleland-Huang, B. Mobasher</td>
<td>Using data mining and recommender systems to facilitate large-scale, open, and inclusive requirements elicitation processes</td>
<td>[32]</td>
</tr>
<tr>
<td>S4</td>
<td>2008</td>
<td>J. Romero-Mariona, H. Ziv, D. J. Richardson</td>
<td>SRRS: a recommendation system for security requirements</td>
<td>[56]</td>
</tr>
<tr>
<td>S6</td>
<td>2009</td>
<td>C. Castro-Herrera, C. Duan, J. Cleland-Huang, B. Mobasher</td>
<td>A recommender system for requirements elicitation in large-scale software projects</td>
<td>[33]</td>
</tr>
<tr>
<td>S7</td>
<td>2009</td>
<td>C. Castro-Herrera, J. Cleland-Huang, B. Mobasher</td>
<td>Enhancing stakeholder profiles to improve recommendations in online requirements elicitation</td>
<td>[34]</td>
</tr>
<tr>
<td>S9</td>
<td>2009</td>
<td>H. Yang, C. Wang</td>
<td>Recommender system for software project planning one application of revised CBR algorithm</td>
<td>[50]</td>
</tr>
<tr>
<td>S10</td>
<td>2009</td>
<td>D. Ning, R. Peng</td>
<td>A Requirements Recommendation Method Based on Service Description</td>
<td>[54]</td>
</tr>
<tr>
<td>S11</td>
<td>2010</td>
<td>S. L. Lim, D. Quercia, A. Finkelstein</td>
<td>StakeSource: harnessing the power of crowdsourcing and social networks in stakeholder analysis</td>
<td>[41]</td>
</tr>
<tr>
<td>S12</td>
<td>2010</td>
<td>S. L. Lim, D. Quercia, A. Finkelstein</td>
<td>StakeNet: using social networks to analyse the stakeholders of large-scale software projects</td>
<td>[43]</td>
</tr>
<tr>
<td>S13</td>
<td>2010</td>
<td>C. Castro-Herrera</td>
<td>A hybrid recommender system for finding relevant users in open source forums</td>
<td>[39]</td>
</tr>
<tr>
<td>S15</td>
<td>2010</td>
<td>A. Felfernig et al.</td>
<td>Recommendation and decision technologies for requirements engineering</td>
<td>[23]</td>
</tr>
<tr>
<td>S16</td>
<td>2010</td>
<td>W. Maalej, A. Sahm</td>
<td>Assisting engineers in switching artifacts by using task semantic and interaction history</td>
<td>[52]</td>
</tr>
<tr>
<td>S17</td>
<td>2010</td>
<td>B. Peischl et al.</td>
<td>Constraint-Based Recommendation for Software Project Effort Estimation</td>
<td>[27]</td>
</tr>
<tr>
<td>S20</td>
<td>2010</td>
<td>M. Kumar, N. Ajmeri, S. Ghaisas</td>
<td>Towards knowledge assisted agile requirements evolution</td>
<td>[29]</td>
</tr>
<tr>
<td>S21</td>
<td>2011</td>
<td>S. L. Lim, D. Damian, A. Finkelstein</td>
<td>StakeSource2.0: using social networks of stakeholders to identify and prioritise requirements</td>
<td>[36]</td>
</tr>
<tr>
<td>S22</td>
<td>2011</td>
<td>S. Lim, A. Finkelstein</td>
<td>StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation</td>
<td>[35]</td>
</tr>
</tbody>
</table>
The two mappings are analyzed individually in the following subsections.

2.3.1 Map A: Overview of RSREs and Their State of Validation

The 21 RSREs assist users in different aspects of RE as highlighted in Figure 2-2. As far as the RE activity aspect is concerned, the mapping shows that a majority of the studies address the elicitation and analysis activities of the requirements engineering process. Very few studies address the validation activity.

The map shows that the number of publications have increased from 2004 to 2010. Fewer studies were identified for the year 2011. This might be because our study was carried during the summer of 2011.

Each study's validation type is briefly described in Table 2-5, and the mapping in Figure 2-2 shows the overall validation state of the identified recommendation systems for requirements engineering with respect to the RE activities. The majority of the studies were evaluated in academia - almost all of them through experiments. Few studies were evaluated in industry - through case studies and through experiments.

The state of validation of the RSREs identified is alarming. Many studies have not been evaluated at all and even fewer studies have been evaluated in an industrial setting. Looking deeper into the studies, 67% of them have not reported any validity threats. Most evaluations of RSREs use only a single data set or project and do not provide details on the validation steps used. These shortcomings have a negative effect on the adoption of such recommendation systems by practitioners.
Among the studies identified in this paper, studies published between 2004 and 2009 have either been validated through experiments in an academic setting, or have not been validated at all. The only three studies [27], [28], [29] that have been validated in an industrial setting appeared in the year 2010.

Table 2-5: RSRE validation description [15]

<table>
<thead>
<tr>
<th>Id</th>
<th>Validation Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>Academic experiment (based on mined feature requests of the open source SugarCRM project (<a href="http://www.sugarcrm.com/crm/">http://www.sugarcrm.com/crm/</a>)</td>
</tr>
<tr>
<td>S2</td>
<td>Academic experiment (based on mined feature requests of the open source SugarCRM project )</td>
</tr>
<tr>
<td>S3</td>
<td>Academic experiment (based on mined feature requests of the open source SugarCRM project )</td>
</tr>
<tr>
<td>S4</td>
<td>Not Validated</td>
</tr>
<tr>
<td>S6</td>
<td>Academic experiments (a: based on feature requests generated for Amazon-like student web-portal by 36 student participants, b: mined feature requests of the open source SugarCRM project and c: feature requests data set of virtual world game SecondLife (<a href="http://secondlife.com/">http://secondlife.com/</a>)</td>
</tr>
<tr>
<td>S7</td>
<td>Academic experiments (a: based on feature requests generated for Amazon-like student web-portal by 36 student participants, b: mined feature requests of the open source SugarCRM project and c: feature requests data set of virtual world game SecondLife, d: specifications of two large scale railway systems)</td>
</tr>
<tr>
<td>S8</td>
<td>Academic experiment (based on feature requests generated for Amazon-like student web-portal by 36 student participants)</td>
</tr>
<tr>
<td>S9</td>
<td>Academic experiment (data from 43 real projects of software consultancy company)</td>
</tr>
<tr>
<td>S10</td>
<td>Not Validated</td>
</tr>
<tr>
<td>S11</td>
<td>Academic Experiment (survey and interviews, 68 stakeholders)</td>
</tr>
<tr>
<td>S12</td>
<td>Academic Experiment (survey and interviews, 68 stakeholders)</td>
</tr>
<tr>
<td>S13</td>
<td>Academic experiment (based on discussion forums of 7 open source projects)</td>
</tr>
<tr>
<td>S14</td>
<td>Not Validated</td>
</tr>
<tr>
<td>S16</td>
<td>Academic simulation (the concepts of the tool evaluated, 7 participants)</td>
</tr>
<tr>
<td>S17</td>
<td>Industry case study (Survey, 7 industry practitioners who gave their opinion on two versions of the recommender system a) intermediate b) final implementation</td>
</tr>
<tr>
<td>S18</td>
<td>Academic experiment (classify non-functional requirements from a total of 625 requirements of 15 different software projects)</td>
</tr>
<tr>
<td>S19</td>
<td>Industry case study (a product line at Hitachi with 123 optional features)</td>
</tr>
<tr>
<td>S20</td>
<td>Industrial experiment (10 features of a requirement specifications)</td>
</tr>
<tr>
<td>S21</td>
<td>Not Validated</td>
</tr>
<tr>
<td>S22</td>
<td>Academic case study (200 interviews, 123 requirements, 87 stakeholders)</td>
</tr>
<tr>
<td>S23</td>
<td>Academic experiment (product descriptions mined from 20 Softpedia.com software product categories)</td>
</tr>
</tbody>
</table>
Of the 21 recommendation systems, 14 are original and seven are extensions of other systems. Just two of the studies [28], [29] come from industry researchers while the remaining 21 studies are from academia. Only one recommendation system [29] has been used in industry.

During this study, we did not find any previous systematic review or mapping study of recommendation systems for RE. However, three studies discuss RSREs in general and highlight the possible usefulness of recommendation systems for RE. Maalej and Thurimella [22] describe use cases of recommendation systems for requirements engineering. Felfernig et al. [23] describe their envisioned recommender system for RE and application scenario where recommender and decision technologies can be used to assist stakeholders in their decision making. The authors provide a description of potentials of recommender systems to improve the quality of decision making in RE, e.g. decisions regarding the requirements to be considered for the next release (release planning), effort estimation, or quality decisions regarding the requirements (understandability, non-redundancy). Castro-Herrera and Cleland-Huang [24] have provided an overview of their research on the application of recommendation systems to assist stakeholders and product managers during the requirements elicitation phase using online discussion forums or wikis.

2.3.2 Map B: Characteristics of the RSREs

The characteristics of the recommendation systems mapped with respect to the requirements engineering activities are explained below, and shown in Figure 2-3.
Figure 2-2: Map A: Mapping of studies with respect to the RE activity addressed, year of publication, and validation type [15].

Figure 2-3: Map B: Combined mapping of the RE activity, recommendation technique, and type of items recommended [15].
2.3.2.1 Recommendation Technique

A recommendation system's engine or main algorithm predicts how interesting a target item is expected to be for a user [10], [30]. This involves generating a predictive model based on data analysis (context and other data) and considering user profiles.

The majority of the RS identified use a collaborative filtering technique. The studies presented in [31], [32], [33], and [34] apply collaborative filtering for predicting how interesting a particular discussion forum might be to a user. Neighborhoods of users with similar interests are found using unsupervised clustering and K-Nearest Neighbor (KNN) [10], which is then used to predict the level of interest of a user with regards to a new forum. StakeRare [35] also uses a collaborative filtering algorithm to predict ‘interesting’ requirements for a stakeholder. For a given stakeholder, KNN is used to find similar stakeholders based on their profiles and rating histories. Unrated requirements are then recommended to the stakeholder that maybe of interest to her. The recommendation feature of StakeSource2.0 [36] applies an item-to-item collaborative filtering algorithm to find unrated requirements that are similar to the requirements that were rated previously by the same stakeholder. A collaborative filtering technique is also used by Castro-Herera and Cleland-Huang [37] in their method to identify expert stakeholders from discussion forums. The same authors report in [24] that during experimentations the KNN collaborative recommender systems approach performed better than the matrix factorization approach - the method used in the winning system for the Netflix movie recommendations prize [38].
Castro-Herrera [39] presents a hybrid RS to identify potential users who could respond to unanswered posts in online forums. A content-based recommender technique analyzes the text of unanswered posts, and compares them to previous posts. Spherical K-Means (SPK) [40] clustering is used to produce groups of similar posts. The users of these posts are then recommended to reply to the unanswered post. It is assumed that these users would likely be interested in replying since they have replied to similar posts in the past.

StakeSource [41] sends emails to an initial list of stakeholders identified by experts, asking them to recommend other stakeholders. New stakeholders are asked to do the same, i.e., the number of stakeholders identified builds up (snowballing effect). A social network of stakeholders is then drawn using the private judgment of each stakeholder about the importance of other stakeholders. Social network measures [42] are then used to prioritize all the project stakeholders. StakeNet [43] also analyzes stakeholders of large-scale projects using social networks.

Various data mining and machine learning techniques are used in the recommendation systems identified in this study. SPK clustering [40] is used in [24] to classify the sections of a project’s vision documents. The same technique is then used to cluster the requirements and feature requests. Cosine similarity is used to compare each topic from the project plan with each topic in the set of feature requests. Unexplored topics would be those topics in the project plan with no matches in the feature requests clusters. Such topics are then recommended to the product manager for further consideration.
Incremental diffusive clustering [44] is used in [45] to create a domain model of the common and variant product features from the mined product descriptions. When a domain analyst provides a preliminary description, the system parses it and uses association rule learning [46], i.e., it finds relations among features within products to provide an initial recommendation. Data mining and association rule learning are also used in [28] to determine new feature constraints from a database of previous feature selections in derived products of a product line.

In [47], a user provides a relatively small number of requirements already classified. The system uses semi-supervised learning (Naive Bayes algorithm [48]) to classify the rest of the requirements into functional or non-functional. A variant of case-based reasoning (CBR) [49] is used in [50] to provide a project manager with similar cases based on a description of the project requirements. Association mining and statistical analysis are used to analyze these cases to provide more fine grained estimates, such as average cost, average revenue rate, resources needed to complete the similar projects.

2.3.2.2 Type of Recommended Items

The authors of [31], [32], and [33] have introduced a recommender system to recommend discussion forums to stakeholders as part of the requirements elicitation phase of large open source projects. The recommendation system has been further improved in [34]. Cleland-Huang et al. [51] applied clustering techniques to automatically allocate feature requests to specific threads in a discussion forum, whereas the recommender system from [33] can be used to introduce stakeholders into one or
more of these discussion threads. The hybrid recommender system proposed in [39] identifies users who might be able to reply to such unanswered forum posts. This work is based on refinement of [37] to identify expert stakeholders of a project. Castro-Herrera and Cleland-Huang [24] proposed a recommender system to inform project managers about forum topics that need additional focus.

Yang and Wang [50] described a recommender system for the task of software project planning - to assist project managers in dealing with the uncertainty at the start of project planning. Peischl et al. [27] presents a recommender system to assist in determining suitable methods for estimating effort, size or project schedule.

Maalej and Sahm [52] presents a recommendation tool for assisting software engineers in switching between artifacts during their tasks. For instance, a developer can refer to related requirements specifications when working with the code. Andric, Hall, and Carr [53] presents a knowledge-based recommender system for assisting in artifact retrieval (such as requirements specifications, design diagrams, etc.) in a software development project.

The types of recommendations among the studies identified in this study take several other forms as well. Study 18 [47] presents a method for automatic identification and classification of non-functional requirements from textual requirements specifications. RS in [54] matches the initial user requirements with existing services which are based on the Service-Oriented Architecture (SOA). A RS method for detection of new feature constraints in an evolving product line by mining previous feature selections in derived products is provided in [28]. Dumitru et al. [45] present a RS that
recommends features similar to a given product description during domain analysis [55]. RS in study 4 helps in estimating an appropriate approach for security requirements specifications [56]. Agile requirements evolution issues are addressed by RS named K-gileRE [29].

2.3.2.3 Output Form

The output of the recommendation system could be pulled by the user, or the system could proactively push information predicted to be useful to the user. The presentation of the recommendation could be textual or visual.

The users of 15 (71%) RS identified in the study request for recommendation (pull). For example, in the RS presented in [27], a software project manager or engineer can request recommendations regarding effort estimation methods suitable for the context of the current project. The output is presented in the form of a web-page which lists all available methods and their descriptions.

Six (29%) RS identified push potentially useful information to the user. StakeSource2.0 [36] pushes the recommended requirements to the stakeholders through e-mails. The output of the recommendation system proposed in [39] uses the push mechanism to inform potential users who might be able to answer unanswered posts in online discussion forums of open source projects. Switch! [52] implicitly tries to predicate the next artifact needed by the engineer, and the recommendations are provide to the user visually.
2.3.2.4 Cross-Dimensional Features

Cross-dimensional features of a RS involve different components of the system. For instance, user feedback about past recommendations could be taken into consideration for future recommendations. The RS could also provide explanations, such as rationale behind the recommendations. Only six (29%) of the RS identified in the review have at least one such feature.

Four RS identified consider user feedback. The recommender system described by Yang and Wang [50] reuses the user’s success experience of project plan construction for future recommendation generations. Accepted recommendations in [28] (feature constraints) are fed back to the recommender system’s input. RS in [45] and [47] consider user feedback from the initial recommendation given, for refining future recommendation generations.

Two RS explain the rationale behind the recommendation provided, such as StakeRare [35]. The web-based recommender system presented by Peischl et al. [27] provides an explanation as to why some other software effort estimation method is not suitable in the current context. This explanation feature of the recommendation system is achieved by utilizing a predicate-based finite state machine (PBFSM) [57]. The second cross-dimensional feature of RS is automatic computation of model-based repair actions [58]. It searches the solution space and looks for the next best recommendable alternative using a software effort estimation method.
2.3.2.5 Architecture

Only ten (48%) RS identified in the study have stated that they have been implemented. The authors of only one RS, StakeSource2.0 [36], mentioned how the tool\(^3\) can be accessed.

Fourteen (67%) RS propose a web-based architecture. RS in [31], [32], [33] have been described as one of the components of a proposed web-based tool. StakeSource [41], StakeSource 2.0 [36], and StakeRare [35] are web-based tools that can easily be made available for stakeholders so that they can submit the requirements and the ratings.

Casamayor, Godoy, and Campo [47] present a recommender system which is a standalone desktop application implemented in Java. The recommender system in [52] is implemented using TeamWeaver [59] that allows users to switch between their pre-defined tasks.

2.4 Discussion

Our analysis of the studies shows that there is none or only limited evaluation of recommendation systems or methods. In most cases, a RS is evaluated on a single data set (e.g. in [35]), and 86% of the systems presented are not evaluated in an industry setting as demonstrated in Figure 2-4. Previous research has been done on the evaluation of recommender systems [60], [61], and on the development of metrics for measuring the effectiveness of recommender systems [60], [62], [63]. These techniques should be utilized and applied in a proper and rigorous manner to evaluate the recommender systems and methods for requirements engineering. Evaluating recommender systems in

\(^3\) http://www.cs.ucl.ac.uk/research/StakeSource/
general is expensive and time-consuming [8], but necessary to truly understand the effectiveness of the system and help with the adoption of the systems in real life projects.

Our results also indicate that there is very limited tool support, in fact only one study [36] had provided a clear link to access a demo of the tool. Having tool support would allow users to truly evaluate the system and see if it is suitable for them. It is also useful to integrate recommendation tools and techniques into the existing work environments of requirement engineers (e.g. as plug-ins). This would ensure fewer context switches and a smoother and light-weight usage experience [22].

RSREs can be improved in other areas as well. Explainability features [64] would be useful in increasing recommendation acceptance by providing explanations to the user regarding the rationale behind the recommendations, e.g. as in [65]. More work needs to be done to address the privacy issues [66]. It is also beneficial to help the user in refining a recommendation request, and address the ambiguity in keyword-based queries [22].

Figure 2-4: RSRE validation types [15]
71% of the RS identified in this study uses a pull mechanism where a user explicitly requests for recommendations. However, the RS would be more effective if estimated useful information is proactively pushed to the user. But then a RS needs to make sure the user is not overloaded with too much information.

Application of recommendation systems and techniques is a relatively new research area according to our mapping results. The first study identified has been published in 2004 [53]. The majority of the studies have been published either in a conference or workshop, and only four studies are a journal publication which indicates low maturity of the research field.

A lot of research has been done in assisting requirements engineers in their task, but not within the context of recommendation systems. These ideas can be utilized and incorporated in RSREs. One of such areas is decision support tools and methodology in RE. Ngo-The and Ruhe [67] have provided an analysis of decision support in RE. Jiang et al. [68] provides a decision support methodology for selecting RE techniques. A method for automatic classification of informal requirements is presented in [69], which is particularly useful in large-scale projects. Data mining and machine learning is used in [70] for a partially automatic technique for requirements prioritization and triage. These are just a few examples of the concepts and techniques that could be used in RSREs.

As far as the limitations of this study are concerned, it is possible that we have not covered all relevant studies. However, we kept the search terms (Table 2-2) as general as possible to make sure all related studies are found in the databases searched, and we
searched through 6,109 studies to select the final 23 studies. The second limitation is that the study selection process was conducted mainly by just one (the first) author.

2.5 Summary

Requirement engineers and analysts deal with a large information space, particularly in complex projects with substantial requirements specifications and a large number of stakeholders. Technologies, such as recommendation systems, that assist requirements engineers are certainly needed. Industry practitioners need to make informed decisions about using recommendation systems for requirements engineering. For us in academia, we need to know the state of the art, possible areas of improvement, and future research areas. In this review, we have addressed these needs by providing the results of a systematic mapping of current recommendation systems for requirements engineering.

Twenty three studies related to recommendation systems or techniques for requirements engineering were identified, from which 21 studies describe a RS and two of them provide an overview of potentials of RSREs. From the 21 recommendation systems, 14 are original and seven are extensions of previous work. The majority of the studies address requirements elicitation and analysis activities, and few consider requirements validation. The number of published RSREs has been increasing over the years. The type of recommended items includes discussion forums, stakeholders and their priority, similar requirements. Collaborative filtering technique is mostly used to generate recommendations. Few systems have cross-dimensional features, such as consideration of
user feedback or providing explanations. Most of the studies are evaluated primarily in academia. Much needed industry validation is rare among the identified studies.
Chapter Three: **RPSim II – Simulation and Sensitivity Analysis Method for Release Planning**

The literature survey presented in the Chapter 2 highlighted various areas of requirements engineering in which recommendation systems have been applied, and discussed their usefulness. For instance, Maalej and Thrimella [22] discuss some use cases of recommendation systems for RE and the benefits of such systems. The authors of [23] envision applicability of recommender systems and decision technologies in RE aimed towards improving the quality of RE decision making. Some example RE areas discussed in [23] include; effort estimation, quality decisions regarding the requirements (understandability, non-redundancy), or decisions regarding the requirements to be considered for the next release (release planning). This thesis focuses on the former example: design and implementation of a RS to assist product managers with better decision making during strategic release planning. According to the literature survey and to the best of our knowledge, the RS presented in this thesis is the first RS for SRP.

The RS described in this thesis is based on three main methods described in Chapters 3, 4, and 5. This chapter provides the details and steps involved in RPSim II - the first of these methods.

RPSim II is an extension to our previous work presented in [71], where we presented a method called *RPSim* for the application of sensitivity analysis when weak constraints in optimization problems have to be changed. The method estimates the impact of changing input parameters of an optimization model on its output. The application area was also strategic release planning, and aimed for the EVOLVE II SRP model. A prototype tool was developed that performs sensitivity analysis, and
investigates the consequences of the changes. The tool uses a simulation model consisting of three steps; data generation, classification of the changes based on their impact level, presentation of the results to the product manager as recommendations. Quality Threshold clustering [72] algorithm was used for classification, and Levenshtein Edit Distance [73] was used for the similarity measure.

The simulation method implemented in \textit{RPSim} considers only two input parameters of the model, and only one parameter is changed at a time. \textit{RPSim} ignores the hard constraints of the optimization problem, these needs to be considered to increase the accuracy of the results. The ReleasePlanner$^{\text{TM}}$ tool generates five diverse release plan solutions; however \textit{RPSim} considered only the first solution for simplicity. In this work, we improved the method by addressing the above limitations, and extended it by adding new features as explained in the following sections.

\subsection*{3.1 Objectives}

Strategic release planning models, such as the EVOLVE II model [3] used in this thesis, are based on numerous input parameters. Before being able to apply such models in practice, managers need to invest time and effort collecting data, such as feature effort estimates, stakeholders and their priority. The more precise the estimates, the better the release plans. If we focus on the most important input parameters, we can likely gather better estimates for them. Simulation based sensitivity analysis allows to achieve the task of ranking the input parameters based on their sensitivity [74], which is the first objective of the \textit{RPSim II} method.

38
Data generated from simulation and sensitivity analysis is a useful source of information for the generation of recommendations in the context of release planning. For some situations, data generated in this manner might be the only way due to lack of historical release data that could be used for analysis. In Chapter 4 we present a method that uses machine learning to analyze such data to learn the impact of input parameter changes of the EVOLVE II SRP model, and estimating the impact of any future input parameter changes. This data is also analyzed by a method presented in Chapter 5 that assists a product manager to achieve certain release planning targets. Data generation and its analysis for recommendation generation is the second objective of the RPSim II method.

3.2 Background

Sensitivity analysis [14] studies the influence of input parameters on the model output, by repeatedly changing input parameters of a model and investigating the consequences of such changes on the output.

There are two main types of sensitivity analysis [74]; local and global sensitivity analysis. In local sensitivity analysis, also called one-at-a-time (OAT-SA), only one input parameter is changed at a time and its impact on the output is studied. The other parameters' values are kept fixed to their original values, and hence any change in the output will clearly be due to the changed input parameter. In practice, an input parameter $x$'s value is changed repeatedly, either randomly or systematically, to varying degrees and each time the effect of the change on the output $y$ is observed. OAT-SA is simple and practical, but this approach does not capture the possible interactions between input
parameters since it does not consider the simultaneous changes of input parameters. For this, global or all-together sensitivity analysis (AT-SA) could be applied, which determines the impact on model output of changing all the input parameter values simultaneously. AT-SA is also explorative of the input space, given that the number of model runs is large.

Both OAT-SA and AT-SA have their advantages and disadvantages as discussed, and the approach to use depends on the problem context. RPSim II method presented in this chapter uses both the approaches. OAT-SA is used for determining the impact of changing the individual input parameters on the output, and consequently rank the input parameters based on their sensitivity. The EVOLVE II SRP model used here is deterministic, i.e. it always generates the same output for a given set of inputs. Therefore, OAT-SA is applicable in this situation since we can be certain that any impact observed on the output is due to the change in a particular input parameter. RPSim II method uses the AT-SA approach for capturing any interactions among the input parameters. The data from this approach cannot be used to determine the sensitivity of the individual parameters, but is a useful data input, along with OAT-SA data, for the recommendation method for predicting the impact of input changes as described in Chapter 4.

3.3 Overview

The main steps of RPSim II are highlighted below, and illustrated with a flowchart in Figure 3-1.

Step 1: The following simulation preparation data are input from a product manager:
a. Input parameters: Selection of the input parameters that are to be considered for sensitivity analysis. The manager uses personal judgement to select the most relevant ones, or all the input parameters for an inclusive sensitivity analysis.

b. Sampling Ranges: RPSim II is a sample-based sensitivity analysis method, and in this step the sampling range for each input parameter type is input by the product manager. This could be based on a probability distribution (e.g. triangular distribution) or the sampling range could be defined by lower and upper bound values.

c. Selection of sensitivity approach: One-at-a-time, All-together, or both.

**Step 2:** RPSim II simulation process starts, which involves the following:

a. Run a SRP model (e.g. EVOLVE II) to generate a release plan with the original input parameter values, this is called the baseline release scenario:

\[ S_0(\text{inputs }<x_1...x_m>, \text{output }<y_1...y_j>). \]

b. In one-at-a-time approach, select an input parameter and vary its value with a sampled value from the sampling range defined earlier. In the all-together approach, a set of input parameters are selected and simultaneously varied. In this manner all the release scenario variations are generated.

c. For each release scenario from b), run the SRP model to determine their release plans (output): \( S_i(\text{inputs }"<x_1...x_m>\), \text{output }"<y_1...y_j>). \)

**Step 3:** Generation of recommendations from the simulation data:
a. Determining sensitivity of the input parameters, and recommending the most important input parameters. This method is detailed in this chapter, and addresses RO2 of the thesis.

b. Training a prediction model on the simulation data, for estimating the impact of future project changes on the release plans. This method is described in Chapter 4, and addresses RO3.

c. Analysis of the simulation data by a utility-based recommendation method for recommending actions for achieving certain release plan targets. This is further explained in Chapter 5, and addresses RO4 of the thesis.
Figure 3-1: RPSim II Overview
3.4 Input Parameters

The input parameters of EVOLVE II model that can be considered in the RPSim II method and the description of these parameters are presented in Table 3-1 below.

Table 3-1. Definition of input parameters considered in RPSim II method

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Efforts</td>
<td>Efforts = {effort(n, r): n=1 to N, r=1 to R }. The set of feature effort estimates, where effort(n, r) is the estimated effort in resource r, to implement feature n. N is the total number of features to be considered during the simulation. R is the total number of resource types.</td>
</tr>
<tr>
<td>Capacities</td>
<td>Capacities = {capacity(r, k): r=1 to R, k=1 to K}. The set of resource capacities, where capacity(r) indicates the amount of resource of type r available for release k. The total number of releases is denoted by K.</td>
</tr>
<tr>
<td>Stakeholder Weights (Stake_w)</td>
<td>Stake_w = {stake_w(p): p=1 to P}. The set of weights of the stakeholders in the prioritization process, where stake_w(p) is the weighting factor for stakeholder p, and the total number of stakeholders is denoted by P.</td>
</tr>
<tr>
<td>Release Weights (Release_w)</td>
<td>Release_w = {release_w(k): k=1 to K}. The set of weights assigned to the release considered for planning, where release_w(k) is the weighting factor for release k.</td>
</tr>
<tr>
<td>Criterion Weights (Criterion_w)</td>
<td>Criterion_w = {criterion_w(q): q=1 to Q}. The set of independent planning criteria, where criterion_w(q) is the weighting factor for criterion q. The number of criteria in the project is denoted by Q.</td>
</tr>
</tbody>
</table>

There are other input parameters of the model that are not being considered by the RPSim II method. They are: number of releases, number of planning criteria, stakeholder votes, and constraints. These parameters are typically not frequently changed during the release planning process. For instance, the number of releases and criteria are decided at
the initial phase of the planning. Similarly, the stakeholders are asked to cast their votes for each feature once at the start of the planning process. The stakeholder votes are not considered in RPSim II since we want to simulate changing those parameters that are altered by the product manager throughout the planning process due to the project realities, such as updated effort estimates, or new capacities for a resource type.

3.5 Sampling Ranges

The sampling range for the input parameters are defined in the manner described in the following sub-sections.

3.5.1 Sampling Range of Efforts, and Capacities:

Random sampling from a defined triangular distribution function TDF (lower, mode, upper) is used to vary the efforts and capacities parameters. For each type of parameter, an expert (e.g. product manager) uses personal judgement to provide the lower, upper, and most likely (mode) percentage values by which the inputs could be changed. Specifically, the sampling ranges for the set efforts and capacities are: $Reffort (\text{lower, mode, upper})$ and $Rcapacities (\text{lower, mode, upper})$.

3.5.2 Sampling Range of Stakeholder, Release, and Criteria Weights

In the EVOLVE II model, a stakeholder, release, or criteria weight is given an integer value between 0 and 9. We will use systematic sampling to generate samples for these parameters within a user given lower and upper bounds. The ranges could be defined as: $Rstake\_w[\text{lower, upper}]$, $Rrelease\_w[\text{lower, upper}]$, and $Rcriteria\_w[\text{lower, upper}]$. 
3.6 Simulation Procedure

Once the input parameters for simulation are selected and the sampling ranges are defined, the automated simulation step of RPSim II can be executed. The overall simulation procedure is detailed in Algorithm 1 shown in Figure 3-2. In addition to the set of selected input parameters and their respective sampling ranges, the algorithm’s inputs include; how many times to vary the effort estimate parameters during OAT-SA (A), number of times to vary the capacity parameters during OAT-SA (B), and number of simulation runs for each level of AT-SA (C).

As shown in Figure 3-2, the algorithm (line 3) first determines the output of the

```
Algorithm 1 RPSim II’s simulation procedure

Require:
    Selected parameters: Efforts, Capacities, Stake.w, Release.w, Criterion.w
    Sampling ranges: Reffort, Rcapacities, Rstake.w, Rrelease.w, Rcriteria.w
    No. of runs for efforts: A, No. runs for capacities: B, No. of AT runs: C

Ensure: A set S of release scenarios have been generated

1: ▷ Find the output of the baseline scenario
2: S ← Ø
3: S₀.Output ← SRPMo del(S₀.Inputs)
4: ▷ Generate scenarios in one-at-a-time approach
5: if OAT-SA is selected then
6:     S₀AT ← Scenarios from Algorithm 2
7:     S ← S ∪ S₀AT
8: end if
9: ▷ Generate scenarios in all-together approach
10: if AT-SA is selected then
11:     S₁AT ← Scenarios from Algorithm 3
12:     S ← S ∪ S₁AT
13: end if
14: for all Sᵢ ∈ S do
15:     Sᵢ.Output ← SRPM odel(Sᵢ.Inputs)
16:     Sᵢ.OutputMetrics ← Compare(S₀, Sᵢ)
17:     Process and save Sᵢ for further analysis
18: end for
19: return S
```

Figure 3-2: Algorithm for the RPSim II's simulation procedure
baseline release scenario, \( S_0 \), as a reference by running a SRP model (e.g. EVOLVE-II). Other scenarios will be compared with the baseline scenario. Then a number of scenarios are generated depending on the type of selected simulation types. Scenario generation involves creating a clone of the baseline scenario’s input settings, and then changing the value of one or more input parameters. Algorithm 1 calls a sub-procedure in line 6 to get a set of scenarios generated using the OAT-SA approach, as explained in Algorithm 2 shown in Figure 3-3. Similarly, Algorithm 1 calls another sub-procedure in line 11 to get a set of scenarios generated in the AT-SA approach, as explained in Algorithm 3 shown in Figure 3-4. The scenarios received from the both the sub-procedures are added to the set of all the scenarios \( \mathcal{S} \).

The final steps of Algorithm 1 (lines 14 to 18) involve determining output of each release scenario by running a SRP model, and computing the output metrics. The output metrics quantifies the difference between our reference baseline scenario \( S_0 \) and a scenario \( S_i \) in the set of scenarios \( \mathcal{S} \). The metrics are detailed in Section 3.7.2. The generated scenarios are saved for further analysis as explained in Section 3.8, Chapter 4, and Chapter 5.

As explained earlier, during OAT-SA (Figure 3-3) only one input parameter is changed at a time, the model is run, and the output is recorded. However, in the AT-SA approach (Figure 3-4), similar parameter types are changed first (lines 3 to 25) and then all the input parameters are changed all together (lines 28 to 44). Changing different types of input parameters at the same time reflects the situations when a product manager tinkers with similar parameters to observe the effects of the changes. And as a result, this
will help us to mine better analysis results from the data in later steps during the mining and learning stages for the purpose of recommendation generation for a product manager.

**Figure 3-3: Procedure of simulation using OAT-SA approach**
Algorithm 3 RPSim II's sub-procedure for AT-SA simulation

Require: Inputs from Algorithm 1
Ensure: A set $S_{AT}$ of release scenarios generated using AT-SA approach

1: $S_{AT} \leftarrow \emptyset$
2: for $i = 0$ to $C$ step 1 do
3:   ▷ 1. Change all effort estimate parameters
4:       $S' \leftarrow \text{Clone}(S_0, \text{Inputs})$
5:       for all effort$(n, r) \in \text{Efforts}$ do
6:           $S'.\text{effort}(n, r).\text{delta} \leftarrow \text{SampleFrom}(\text{Reffort})$
7:       end for
8:       $S_{AT} \leftarrow S_{AT} \cup S'$
9:   ▷ 2. Change all resource capacity parameters
10:      $S' \leftarrow \text{Clone}(S_0, \text{Inputs})$
11:      for all capacity$(r, k) \in \text{Capacities}$ do
12:          $S'.\text{capacity}(r, k).\text{delta} \leftarrow \text{SampleFrom}(\text{Rcapacities})$
13:      end for
14:      $S_{AT} \leftarrow S_{AT} \cup S'$
15:   ▷ 3. Change all stakeholder, release, and criteria weights
16:      $S' \leftarrow \text{Clone}(S_0, \text{Inputs})$
17:      for all stake$_w$(p) $\in \text{Stake}_w$ do
18:          $S'.\text{stake}_w(p).\text{delta} \leftarrow \text{SampleFrom}(\text{Rstake}_w)$
19:      end for
20:      for all release$_w$(k) $\in \text{Release}_w$ do
21:          $S'.\text{release}_w(k).\text{delta} \leftarrow \text{SampleFrom}(\text{Rrelease}_w)$
22:      end for
23:      for all criterion$_w$(q) $\in \text{Criterion}_w$ do
24:          $S'.\text{criterion}_w(q).\text{delta} \leftarrow \text{SampleFrom}(\text{Rcriteria}_w)$
25:      end for
26:      $S_{AT} \leftarrow S_{AT} \cup S'$
27:   ▷ 4. Change all input parameters
28:      $S' \leftarrow \text{Clone}(S_0, \text{Inputs})$
29:      for all effort$(n, r) \in \text{Efforts}$ do
30:          $S'.\text{effort}(n, r).\text{delta} \leftarrow \text{SampleFrom}(\text{Reffort})$
31:      end for
32:      for all capacity$(r, k) \in \text{Capacities}$ do
33:          $S'.\text{capacity}(r, k).\text{delta} \leftarrow \text{SampleFrom}(\text{Rcapacities})$
34:      end for
35:      for all stake$_w$(p) $\in \text{Stake}_w$ do
36:          $S'.\text{stake}_w(p).\text{delta} \leftarrow \text{SampleFrom}(\text{Rstake}_w)$
37:      end for
38:      for all release$_w$(k) $\in \text{Release}_w$ do
39:          $S'.\text{release}_w(k).\text{delta} \leftarrow \text{SampleFrom}(\text{Rrelease}_w)$
40:      end for
41:      for all criterion$_w$(q) $\in \text{Criterion}_w$ do
42:          $S'.\text{criterion}_w(q).\text{delta} \leftarrow \text{SampleFrom}(\text{Rcriteria}_w)$
43:      end for
44:      $S_{AT} \leftarrow S_{AT} \cup S'$
45: end for
46: return $S_{AT}$

Figure 3-4: Procedure of simulation using AT-SA approach in RPSim II
3.7 Metrics

Defining the right metrics is important in quantifying the changes under study during sensitivity analysis. In this section, Goal Question Metric (GQM) [75] approach is used to define the metrics needed for the RPSim II method. The top-down approach in GQM is suitable in our situation since the overall objectives (Section 3.1) are clear, and only those aspects of release planning need to be measured that contribute to the goal.

**Goal:** Measure the impact of changing the value of one or more input parameter of a baseline release scenario on the output (release plan), and rank the input parameters based on their sensitivity.

**Questions:** The underlying task in order to fulfil our goal is to quantify the difference between the baseline release scenario $S_0$(inputs $x_1...x_m$, output $y_1...y_j$) and a new release scenario $S_i$(inputs $"x_1...x_m", output"y_1...y_j$) from sensitivity analysis point of view. Release $S_i$ is generated by altering one or more inputs of the baseline release plan $S_0$, and as a result $S_i$ might have a different output. The measurement goal is further refined by the following questions.

Q1: By how much the values of input parameters $(x_1...x_m)$ have changed relative to their original values in the baseline release scenario?

Q2: By how much the baseline plan (output) has changed due to the alteration of the input parameter(s)?

Q3: What is the sensitivity level of an input parameter?

**Metrics:** The metric defined for Q1 is detailed in section 3.7.1. Q2 essentially involves quantifying the difference between two release plans, and this could be done in
several ways. As a result, we first provide 5 metrics in section 3.7.2 for Q2 with consideration of their usefulness for both sensitivity analysis as well as recommendations for a product manager (as further explained in Chapters 4 and 5). In section 3.8.1 another metric is defined that integrates the former 5 metrics into one. The metric for measuring the sensitivity of an input parameter (Q3) is presented in section 3.8.2, and this metric is dependent on all the other metrics defined earlier.

### 3.7.1 Metric for Input Parameter Changes

The change in release, criteria, and stakeholder weight input parameters, $\Delta X_{Si}$, for a simulation run $S_i$ is calculated as:

$$
\Delta X_{Si} = \frac{X_{Si} - X_{S0}}{X_{S0}} \quad \text{for} \quad X_{Si} \geq 0, X_{S0} > 0
$$

$$
\Delta X_{S0} = X_{Si} \quad \text{for} \quad X_{S0} = 0
$$

Where $X_{S0}$ is the original value and $X_{Si}$ is the sampled value for the input parameter $X$ in the simulation run $S_i$. This formula is applied for calculating the change in any input parameter $X$ that can be $\Delta_{\text{stake}_w(p)}$, $\Delta_{\text{release}_w(k)}$, and $\Delta_{\text{criterion}_w(q)}$ as defined in Table 3-1.

The change in efforts and resource capacities ($\Delta_{\text{effort}(n, r)}$, $\Delta_{\text{capacity}(r, k)}$) are the values sampled from their respective sampling ranges and therefore we do not need to calculate the change.

### 3.7.2 Metrics for Output Changes

The metric defined in this section quantifies the difference between the output of baseline scenario $S_0$, and output of a scenario $S_i$. The difference is measured by the metrics;
release plan structure (PS), stakeholder feature points (SFP), number of features postponed (PF), resource consumption (RC), and stakeholder excitement analysis (EA).

3.7.2.1 Release Plan Structure ($Ay_1$)
EVOLVE II’s output consists of a set of release solutions, called alternatives. The main component of each alternative solution is the release plan, which shows the assignment of each feature to a release or if it will be postponed. A maximum of five diverse alternatives are provided, the number of alternatives could be less than five. We use Hamming distance (HDist) [76] to measure the difference between the plan structure of the original release solution set, and the plan structure of the solution set generated during a simulation run. Hamming distance is defined as the minimum number of substitutions needed to change string A into another string B (both are of the same length).

For instance, consider two alternative release plans: ($A_{1o}$: 1 2 1 1 1 2 3) and ($A_{1s}$: 2 1 1 1 1 2 1). $A_{1o}$ is the first release alternative of the baseline scenario, and $A_{1s}$ is the first release alternative of a release scenario $s$. The numbers in the sequences represent each feature’s assignment to a release (1, 2, or 3). For example, in $A_{1o}$ the first number (1) means that feature #1 is assigned to release-1 in this plan, while the last number (3) in the sequence means that feature #8 is assigned to release-3. We can see that the assignment of features differs in each alternative release plans. The structural difference between the two release alternatives $\text{HDist}(A_{1o}, A_{1s})$ would be 3 – the distance value of $A_{1s}$ with respect to $A_{1o}$. In order to get the ratio of structure differences with respect to all the features of the release plan, the Hamming distance value is divided by
the total number of features $N$. In this case it would be $3/8$. A high ratio value would indicate a release plan which is very different compared to the original release plan.

We need to have a single value that indicates the structural difference between the original scenario and another scenario. Therefore, we use the following formula to combine the structural difference of the all the alternatives into a single difference value.

$$
\Delta y_1 = HDist(S_0, S_i) = \frac{\sum_{l=1}^{L} HDist(A_{S0,l}, A_{Si,l})}{N} \leq N
$$

$$
L = \text{maximum}(L_{S0}, L_{S_i})
$$

where $A_{S0,l}$ is the $l$th alternative of the original release plan, $A_{Si,l}$ is the $l$th alternative of the release plan release scenario $S_i$, and $N$ is the total number of features of the release planning project. $L_{S0}$ is the number of release plan alternatives the original scenario. $L_{S_i}$ is the number of release plan alternatives a scenario $S_i$. $L_{S0}$ and $L_{S_i}$ are not always equal. In order to factor in this difference, the maximum of the two are considered in the formula above. For instance, if $L_{S0} = 5$ and $L_{S_i} = 4$, then $A_{S_i,5}$ (the missing alternative in scenario $S_i$) is considered as an empty string when the Hamming distance $HDist$ is calculated.

3.7.2.2 Stakeholder Feature Points ($\Delta y_2$)

Stakeholder feature points is an absolute integer value representing the optimality of a release plan alternative provided by EVOLVE II. The level of optimality depends on how well a release plan solution utilizes the available resources. SFP helps a PM to evaluate the quality of a release plan. The average value of SFP from $L$ alternative release plans is
calculated, and the difference with the original plan’s average SFP value is found, and reflected in metric $\Delta y_2$:

$$\Delta y_2 = \frac{\sum_{l=1}^{L} SFP_{S_i,l} - SFP_{S0,l}}{L} \quad f o r \quad SFP_{S_i,l} \geq 0, SFP_{S0,l} > 0$$

(3-3)

where $SFP_{S0,l}$ is the stakeholder feature points of alternative $l$ of baseline scenario $S_0$, and $SFP_{S_i,l}$ is the stakeholder feature points of alternative $l$ in a release scenario $S_i$, and $L$ is the number of alternative release plans in $S_i$.

3.7.2.3 Postponed Features ($\Delta y_3$)

Number of postponed features is the total number of features than could not be allocated in one of the releases and therefore had to be delayed, due to limited resources or other constraints between the features, e.g. feature 1 cannot be released before feature 2. These constraints are determined by a product manager during project setup. The average value of PF from $L$ alternative release plans is calculated, and the difference with the original plan’s average PF value is found. This difference is reflected in metric $\Delta y_3$:

$$\Delta y_3 = \frac{\sum_{l=1}^{L} \Delta PF}{L} \quad f o r \quad PF_{S_i,l} \geq 0$$

(3-4)

$$\Delta PF = \begin{cases} \frac{PF_{S0,l} - PF_{S_i,l}}{PF_{S0,l}} & \text{if } PF_{S0,l} > 0 \\ -PF_{S_i,l} & \text{if } PF_{S0,l} = 0 \end{cases}$$

where $PF_{S0,l}$ is the number of features postponed in alternative $l$ of baseline scenario $S_0$, and $PF_{S_i,l}$ is the number of features postponed in alternative $l$ of a release scenario $S_i$, and $L$ is the number of alternative release plans in $S_i$. 

54
If $\Delta y_3$ is 0, there is no change (neutral). If it is greater than 0, it means the number of postponed features have decreased. If it is less than 0, it means that the number of postponed features has increased overall.

3.7.2.4 Resource Consumption ($\Delta y_{4r}$)

This metric reflects the difference in resource consumption for a resource type $r$ between the baseline release scenario $S_0$ and a scenario $S_i$. Resource consumption is taken from a SRP model, for example EVOLVE II provides a summary of the resource consumption report for each resource in each release plan alternative.

$$
\Delta y_{4r} = \begin{cases} 
\frac{(C_{S0,r} - C_{S_i,r})}{C_{S0,r}} & \text{if } C_{S0,r} > 0 \\
C_{S_i,r} & \text{if } C_{S0,r} = 0
\end{cases}
$$

where $C_{S0,r}$ and $C_{S_i,r}$ are the weighted sums of the resource consumption for all the releases of the original release scenario $S_0$ and a scenario $S_i$ respectively. Either one is given by:

$$
C_r = \sum_{k=1}^{K} C_{rk} \cdot w_k
$$

where $w_k$ is the weight of a release $k$, and $C_{rk}$ is average resource consumption rate of the resource $r$, in all the alternatives of a release $k$, in a release scenario. It is calculated by:

$$
C_{rk} = \frac{\sum_{l=1}^{L} RC_{l,r,k}}{L}
$$

$L$ is the number of alternatives in the solution set in the release scenario, and $RC_{l,r,k}$ the consumption rate of resource $r$ (in %) for release $k$ in alternative $l$. A positive $\Delta y_{5r}$ reflects increased resource consumption for resource type $r$. 

55
3.7.2.5 Stakeholder Excitement Analysis Metric ($\Delta y^3_p$)

Stakeholder excitement analysis is a functionality provided by the EVOLVE II model, which provides a quantitative measure of the projected satisfaction or dissatisfaction of a stakeholder with a proposed release plan. The analysis reflects the agreement between the assignment of individual features in a particular release plan, and the wishes of the stakeholder for those features (expressed by her votes for the features over the planning criteria). This agreement is expressed through seven categories as shown in example stakeholder excitement analysis in Figure 3-5, where the table shows satisfaction level of each stakeholder for $x$ number of features in a release plan (e.g. stakeholder with username ‘arash@email.com’ is expected to be ‘very excited’ due to assignment of 5 features, and ‘very disappointed’ due to assignment of 2 features). The satisfaction level is determined in this manner: consider a project with a planning scope of two releases ($K=2$). Very Excited (VE) would refer to the situation when a feature is considered very important (scored 8 or 9 out of 9) and the feature is assigned to the first of the two releases of the project. The stakeholder would be Very Disappointed (VD) if this important feature is postponed, and Disappointed (D) if the feature is assigned to Release 2. However, if the stakeholder had assigned a medium level score (5, 6, 7) and the feature is assigned to Release 1, the stakeholder's excitement category in this situation would be projected to be Excited (E). The other excitement categories are neutral, surprised, and very surprised which are detailed in [3]. These are not considered here since the purpose is to quantify the excitement (or disappointed) of the stakeholders for a given release plan.
Excitement analysis helps a product manager to compare different release plan alternatives in a solution set from the perspective of stakeholder satisfaction. However, our goal is to compare two different solution sets; the alternative release plans in baseline scenario $S_0$ and release plans in a scenario $S_i$. We also need to quantify this comparison in a single value. For this, the unaltered stakeholder excitement analysis is processed further by combining multiple numerical values of the analysis to quantify the satisfaction level of stakeholder $p$ (with a weight $w_p$) for a solution set in either $S_0$ or $S_i$, according to the following:

$$\begin{align*}
EA_{S_0,p} \text{ or } EA_{S_i,p} &= w_p \sum_{l=1}^{L} (w_{ve} \cdot VE_l + w_e \cdot E_l) - (w_{vd} \cdot VD_l + w_d \cdot D_l)
\end{align*}$$

(3-8)

Where $w_{ve}$, $w_e$, $w_{vd}$, and $w_d$ are weights assigned for each factor of the stakeholder excitement formula above. $L$ is the number of alternative release plans. Using above we can then calculate the relative difference of these values between the baseline scenario and any other scenario $S_i$:

---

**Figure 3-5. Stakeholder Excitement Evaluation from ReleasePlanner™**
\[ \Delta y_{5p} = \frac{EAS_{i,p} - EAS_{0,p}}{|EAS_{i,p}|} \quad \text{for } EAS_{i,p} \neq 0 \]  

\[ \Delta y_{5p} = EAS_{i,p} \quad \text{for } EAS_{i,p} = 0 \]  

A positive \( \Delta y_{5p} \) value indicates increased satisfaction of stakeholder \( p \) compared to the baseline release plan.

3.8 Sensitivity Computations

The data generated during the OAT-SA approach is used to determine the sensitivity index of the input parameters. The calculations involved are explained in the following sections.

3.8.1 Combining the Output Metrics

In order to present the overall output changes in a release scenario \( S_i \) with respect to the baseline scenario \( S_0 \), the changes in all the output metric (\( \Delta y_j \) for \( j = 1 \) to 5) need to be aggregated into a single output metric \( \Delta Y_{Si} \). The overall change is then calculated as the following:

\[ \Delta Y_{Si} = \frac{\sum_{j=1}^{5} w_j \cdot |\Delta y_j| + (w_4 \cdot \sum_{r=1}^{R} |\Delta y_{4r}|) + (w_5 \cdot \sum_{p=1}^{P} |\Delta y_{5p}|)}{\sum_{i=1}^{5} w_j} \]  

(3-10)

where \( w_j \) is a weight (0 <= \( w_j <= 5 \)) given to an output metric \( j \) by the product manager.

This gives a product manager the option to select the output parameters and their importance when determining the sensitivity indices of the input parameters as explained in the next section.

3.8.2 Calculating Sensitivity Indices

Sensitivity index (SI) of an input parameter \( x \) determines how much impact it has on the release plans. The algorithm used to calculate the sensitivity index of each input
parameter selected during a simulation is shown in Figure 3-6. The set of input parameters $X$ could contain parameters of type effort($n$, $r$), capacity($r,k$), stake$_w(p)$, release$_w(k)$, and criterion$_w(q)$ which are explained in detail in Table 3-1. Only the scenarios generated during OAT-SA approach is considered here, and is denoted by the set $S_{OAT}$.

The overall approach to find sensitivity of each input $x$ in $X$, is to iterate over each scenario $S_i \in S_{OAT}$, and only consider $S_i$ if the input $x$ had been changed in this release scenario during the simulation. If so, then the absolute value of change in input $x$ is set to $\Delta x$ (line 13). The input change metrics were explained in Section 3.7.1. Then the metric from Section 3.8.1 is used to aggregate the overall change in the outputs of $S_i$ compared to the baseline scenario $S_o$, and this is set to $\Delta Y$ (line 16). The ratio of the change in input ($\Delta x$) and change in the output ($\Delta Y$) is found. The ratios are added and their average (from all scenarios where input $x$ have been changed) defines the overall sensitivity of the input parameter $x$. 

59
The algorithm returns the sensitivity of the input parameters based on the given perspectives. Sensitivity index of each input parameter is provided as recommendation to the product manager (visually or as a list). The higher the value is, the more sensitive the input parameter. The product manager can take further action based on this information, such as re-considering the effort estimation of the most sensitive (important) features. Applicability of such a recommendation is further demonstrated in the Section 7.4.1 of

![Algorithm 4 RPSim II’s procedure for calculating sensitivity indices](image)

**Figure 3-6: Algorithm for calculating sensitivity indices**
the case study in Chapter 7.

3.9 Visualization of Sensitivity Analysis Data

The sensitivity analysis data can also be visually presented to the product manager using scatterplots, to show the relationship between input parameter changes (x-axis) and output parameter changes (y-axis). Personal judgement could be used to detect any project specific trends and patterns. Typically, if the points in the plot have a clear curve-like shape, then this is an indication of a strong relation between the input parameter changes and the output. On the other hand, a scatter plot with no clear pattern would indicate that other input parameters are more influential on the output. Applicability of such analysis is further discussed in Section 7.4.2 of the case study in Chapter 7.

3.10 Illustrative Example

An artificial and very simple release planning project is used as an example in this section to illustrate application of RPSim II method, and to demonstrate how the metrics and sensitivity indices are calculated.

3.10.1 Application of RPSim II

Example project’s properties are shown in Table 3-2. To apply RPSim II, we follow the

<table>
<thead>
<tr>
<th>Table 3-2: Project properties for illustrative example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Project Property</strong></td>
</tr>
<tr>
<td>-----------------------</td>
</tr>
<tr>
<td>N = 10 Requirements</td>
</tr>
<tr>
<td>R = 1 Resource</td>
</tr>
<tr>
<td>P = 2 Stakeholders</td>
</tr>
<tr>
<td>K = 2 Releases</td>
</tr>
<tr>
<td>Q = 2 Criteria</td>
</tr>
</tbody>
</table>
steps explained in Section 3.3. **Step 1:** Two input parameters of the project are selected for simulation namely capacity(Res1, R1) and capacity(Res1, R2). And the sampling range for capacity parameters is Rcapacities(5%, 20%, 50%) – a triangular distribution function as explained in Section 3.5.1. For simplicity, we select only the one-at-a-time simulation and sensitivity approach. **Step 2:** We assume that the baseline scenario \((S_0)\) and four other release scenarios \((S_1, S_2, S_3, \text{ and } S_4)\) have been generated in this step. The scenarios are shown in Table 3-3. Each scenario’s output is made up and not determined by running a SRP model here for simplicity. **Step 3:** The scenarios are analyzed to extract different types of recommendations; the sensitivity analysis is explained next, and two other types of recommendations extracted from the simulation data is provided in Chapter 4 and 5.

**3.10.2 Calculations of Metrics**

Metrics defined in Section 3.7 play a central role in the analysis of the simulation data. The goal of the metrics is to compare the changes in input parameters and output of a release scenario with respect to the baseline scenario \(S_0\). Using the example simulation data from Table 3-3, we have calculated the metrics for the scenarios as shown in Table 3-4. To further explain the calculations, we detail the metric calculations for one of the scenarios \(S_1\), i.e. quantity the difference between \(S_0\) and \(S_1\).

Metric from Section 3.7.1 is used to determine the values of input change metrics, e.g. \(\Delta capacity(Res1, R1) = (110 - 100)/100 = 0.1\). This denotes the difference between capacity(Res1, R1)’s original value in \(S_0\) (100) and its new value in \(S_1\) (110) - an increase of 10%.
Metrics for output changes explained in Section 3.7.2 are used next to quantify the output differences of $S_0$ and $S_1$. In each case $L$ (the number of alternatives) is 2.

First, consider the release plan structure metric ($\Delta y_1$). The release plan alternatives of baseline scenario $S_0$ from Table 3-3 are: $A_{S0,1}$ (2,1,3,1,3,1,1,1,2,3) and $A_{S0,2}$ (2,1,3,1,3,1,1,1,2,3). Similarly, the release plan alternatives of $S_1$ are denoted as $A_{S1,1}$ and $A_{S1,2}$. Equation 3-2 is used to find the $\Delta y_1$, with $N=10$:

$$
\Delta y_1 = \left( \frac{HDist(A_{S0,1}, A_{S1,1})}{10} + \frac{HDist(A_{S0,2}, A_{S1,2})}{10} \right) \times \frac{1}{2} = \frac{5}{10} + \frac{7}{10} = 0.6
$$

The standard Hamming distance algorithm [76] is used to determine HDist values. For instance, $HDist(A_{S0,1}, A_{S1,1}) = 5$, since five changes are needed to convert string $A_{S0,1}$ to $A_{S1,1}$.

Stakeholder feature points ($\Delta y_2$) is determined using Eq. (3-3):

$$
\Delta y_2 = \left( \frac{SF_{P_{S1,A2}} - SF_{P_{S0,A2}}}{SF_{P_{S0,A1}}} + \frac{SF_{P_{S1,A1}} - SF_{P_{S0,A2}}}{SF_{P_{S0,A2}}} \right) \times \frac{1}{2}
$$

$$
\Delta y_2 = \left( \frac{(22900 - 21712)}{21712} + \frac{(22890 - 21700)}{21700} \right) \times \frac{1}{2} = 0.0548
$$
Table 3-3: Illustrative example of RPSim II simulation data

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>$S_0$</th>
<th>$S_1$</th>
<th>$S_2$</th>
<th>$S_3$</th>
<th>$S_4$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input parameters selected for simulation:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>capacity(Res1, R1)</td>
<td>100</td>
<td>110</td>
<td>120</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>capacity(Res1, R2)</td>
<td>50</td>
<td>50</td>
<td>50</td>
<td>60</td>
<td>70</td>
</tr>
<tr>
<td><strong>Output:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release Alternative 1 (A1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S. Feature Points</td>
<td>21712</td>
<td>22900</td>
<td>22690</td>
<td>22180</td>
<td>23920</td>
</tr>
<tr>
<td>Postponed Features</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>RC- Res1, R1</td>
<td>98%</td>
<td>99%</td>
<td>100%</td>
<td>99%</td>
<td>100%</td>
</tr>
<tr>
<td>RC- Res1, R2</td>
<td>75%</td>
<td>80%</td>
<td>84%</td>
<td>95%</td>
<td>96%</td>
</tr>
<tr>
<td>Stake1 EA - VE</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>5</td>
<td>6</td>
</tr>
<tr>
<td>Stake1 EA - E</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>Stake1 EA - VD</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Stake1 EA - D</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Stake2 EA - VE</td>
<td>3</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Stake2 EA - E</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Stake2 EA - VD</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Stake2 EA - D</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Release Alternative 2 (A2)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S. Feature Points</td>
<td>21700</td>
<td>22890</td>
<td>22800</td>
<td>22970</td>
<td>23840</td>
</tr>
<tr>
<td>Postponed Features</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>RC- Res1, R1</td>
<td>98%</td>
<td>99%</td>
<td>99%</td>
<td>98%</td>
<td>98%</td>
</tr>
<tr>
<td>RC- Res1, R2</td>
<td>79%</td>
<td>82%</td>
<td>89%</td>
<td>92%</td>
<td>97%</td>
</tr>
<tr>
<td>Stake1 EA - VE</td>
<td>5</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Stake1 EA - E</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td>Stake1 EA - VD</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Stake1 EA - D</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Stake2 EA - VE</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Stake2 EA - E</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Stake2 EA - VD</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Stake2 EA - D</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Postponed features metric ($\Delta y_3$) for scenario $S_1$ is found using Eq. (3-4):

64
\[ \Delta y_2 = \left( \frac{PF_{S0,A1} - PF_{S1,A1}}{SFP_{S0,A1}} + \frac{PF_{S0,A2} - PF_{S1,A2}}{SFP_{S0,A2}} \right) \cdot \frac{1}{2} \]

\[ \Delta y_2 = \left( \frac{(3 - 1)}{3} + \frac{(3 - 1)}{3} \right) \cdot \frac{1}{2} = 0.6667 \]

The metric for resource consumption (\( \Delta y_{4,Res1} \)), for \( S_I \) is determined using the equations explained in Section 3.7.2.4. First, the average consumption of Res1 in both alternatives for each release in \( S_0 \) is determined using Eq. (3-7):

\[
C_{S0,Res1,R1} = \frac{RC_{A1,Res1,R1} + RC_{A2,Res1,R1}}{2} = \frac{98 + 98}{2} = 98
\]

\[
C_{S0,Res1,R2} = \frac{RC_{A1,Res1,R2} + RC_{A2,Res1,R2}}{2} = \frac{75 + 79}{2} = 77
\]

Then the weighted sum of Res1 consumption in both releases for \( S_0 \) is determined using Eq. (3-6):

\[ C_{S0,Res1} = (C_{S0,Res1,R1} \cdot w_{R1}) + (C_{S0,Res1,R2} \cdot w_{R2}) = (98 \cdot 9) + (77 \cdot 6) = 1344 \]

Eq. (3-7) is applied to determine the average consumption of Res1 in \( S_I \) as well (not shown here), and then Eq. (3-6) applied to find weighted sum of Res1 consumption for \( S_I \):

\[ C_{S1,Res1} = (C_{S1,Res1,R1} \cdot w_{R1}) + (C_{S1,Res1,R2} \cdot w_{R2}) = (99 \cdot 9) + (81 \cdot 6) = 1377 \]

Finally, Eq. (3-5) is used to find value of metric \( \Delta y_{4,Res1} \):

\[ \Delta y_{4,Res1} = \frac{C_{S1,Res1} - C_{S0,Res1}}{C_{S0,Res1}} = \frac{1377 - 1344}{1344} = 0.0246 \]

Excitement analysis metric of project stakeholders Stake1 and Stake2 are determined following steps from Section 3.7.2.5. Here we show the metric calculations for Stake1 only, and the same steps are applied for Stake2 as well. First, \( EA \) of Stake1 in
baseline scenario $S_0$ is found by Eq. (3-8), where the weights $w_{ve}, w_e, w_{vd},$ and $w_d$ are 1 for simplicity:

$$EA_{S0,Stake1} = w_{Stake1} \times [(VE_{A1} + EA_{A1} - VD_{A1} - D_{A1}) + (VE_{A2} + EA_{A2} - VD_{A2} - D_{A2})]$$

$$EA_{S0,Stake1} = 9 \times [(6 + 2 - 2 - 0) + (5 + 3 - 1 - 1)] = 108$$

Similarly, $EA$ of Stake1 in scenario $S_I$ is determined by:

$$EA_{S1,Stake1} = 9 \times [(7 + 2 - 1 - 0) + (5 + 4 - 0 - 1)] = 144$$

Finally, difference in $EA$ of Stake1 in $S_0$ and $S_I$ is determined by metric $\Delta y_{5,Stake1}$ using Eq. (3-9):

$$\Delta y_{5,Stake1} = \frac{EA_{S1,Stake1} - EA_{S0,Stake1}}{|EA_{S0,Stake1}|} = \frac{144 - 108}{108} = 0.3333$$

Here the metric calculations for $S_I$ were only showed, and in the same manner the metrics for the other scenarios $S_2, S_3,$ and $S_4$ are calculated. The final metric values of all the scenarios are shown Table 3-4. Next, we demonstrate how the sensitivity of the input parameters are determined using the calculated metrics.

### Table 3-4: Input and output change metrics of illustrative simulation data

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Input Change Metrics</th>
<th>Output Change Metrics</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\Delta$capacity ($Res1,R1$)</td>
<td>$\Delta$capacity ($Res1,R2$)</td>
</tr>
<tr>
<td>$S_0$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$S_1$</td>
<td>0.1</td>
<td>0</td>
</tr>
<tr>
<td>$S_2$</td>
<td>0.2</td>
<td>0</td>
</tr>
<tr>
<td>$S_3$</td>
<td>0</td>
<td>0.2</td>
</tr>
<tr>
<td>$S_4$</td>
<td>0</td>
<td>0.4</td>
</tr>
</tbody>
</table>
3.10.3 Computing Sensitivity Indices

The procedure discussed in Section 3.8.2 is applied to metric data in Table 3-4 to determine the sensitivity index of the input parameters selected for simulation in this illustrative example. Equation 3-10 in Section 3.8.2 let us combine the output metrics into one (needed for sensitivity calculations), where each output metric is given a weight. In this example: \( w_1 = 4, \ w_2 = 1, \ w_3 = 2, \ w_4 = 0, \) and \( w_5 = 5. \) These weights are usually assigned by the product manager, but here we use these values for illustration purpose only. Note that \( w_4 = 0 \) implies that the metric \( \Delta y_{4,Res1} \) would not contribute during sensitivity calculations, while \( w_5 = 5 \) implies that stakeholders’ excitement metric will contribute more.

First, we consider the input parameter capacity(Res1, R1). From simulation data we see that capacity(Res1, R1)’s value has been changed only in scenarios \( S_1 \) and \( S_2. \) Therefore, only these two will be analyzed to determine SI of capacity(Res1, R1). In \( S_1, \)

\[
\Delta y_{S1} = \frac{(1 \cdot 0.6) + (1 \cdot 0.0548) + (2 \cdot 0.6667) + (0 \cdot 0.0246) + 5 \cdot (0.3333 + |−0.2|)}{12} = 0.5379
\]

Then SI of capacity(Res1, R1) for \( S_1 \) is:

\[
SI(S_1) = \frac{\Delta y_{S1}}{\Delta capacity(Res1, R1)} = \frac{0.5379}{0.1} = 5.379
\]

We can observe that the SI value of scenario \( S_1 \) is directly proportional to the amount of difference between the output of scenario \( S_1 \) and the baseline scenario \( S_0 \) (quantified by the output metrics), and is indirectly proportional to the change of the input parameter (capacity(Res1, R1) in the above example).
Now, we consider $S_2$ where $\Delta \text{capacity}(\text{Res1,R1}) = 0.2$. Similar to above, the SI of $\text{capacity}(\text{Res1,R1})$ is found to be:

$$SI(S_2) = \frac{\Delta Y_{S_2}}{\Delta \text{capacity}(\text{Res1,R1})} = \frac{0.7789}{0.2} = 3.8945$$

Then, the overall sensitivity index of $\text{capacity}(\text{Res1,R1})$ is the average of sensitivities of this input in $S_1$ and $S_2$:

$$SI \text{ of } \text{capacity}(\text{Res1,R1}) = \frac{5.379 + 3.8945}{2} = 4.6368$$

Following the above steps, the SI of the other input parameter $\Delta \text{capacity}(\text{Res1,R2})$ is found by considering the scenarios where its values were changed: $S_3$ and $S_4$:

$$SI \text{ of } \text{capacity}(\text{Res1,R2}) = \frac{1.2944 + 1.9448}{2} = 3.2392$$

We can see that input parameter $\text{capacity}(\text{Res1,R1})$ is more sensitive to changes in this release planning project.

In this illustrative example we demonstrated how RPSim II method can be applied for a very simplistic case and using artificial data. Nevertheless, we can see that large volumes of data and a lot of computations are involved in such analysis. This re-iterates our belief that recommendation systems for strategic release planning are needed to extract useful information form large volumes of release planning data. To this end, two other recommendation methods are explained in Chapters 4 and 5 that utilize the data generated during the simulation and sensitivity analysis of RPSim II method. To further demonstrate the applicability of the recommendation system presented in this thesis, a large real-life project is used in a case study in presented in Chapter 7.
3.11 Summary

In this chapter we introduced the RPSim II method – a simulation and sensitivity analysis method for strategic release planning. The objectives of the method with respect to the final recommendation system were explained. The simulation process and the calculations involved in determining the sensitivity of the input parameters are presented. An illustrative example was used to demonstrate application of RPSim II method, and the metric calculations involved.

In the next chapter, we present a method that further utilizes the sensitivity and simulation data to train a prediction model that could estimate the impact project changes on the release plans.
Chapter Four: **Predicting the Impact of Release Plan Changes**

4.1 **Introduction**

A product manager often changes the release planning parameters either proactively with the goal of improving the release plans, or reactively in light of new or revised information. It would be useful to know the impact of such changes on the current release plans partly to avoid any unforeseen consequences, such as disappointment of a stakeholder or delay of an important feature.

In what-if analysis some input parameters are manually changed in order to compare resulting release plans with the current plan. However, there are many possible input parameter combinations that influence the final release plan. For instance, the effort estimates of a feature for every resource type are inputs for the planning model. Similarly, there are several output attributes for a certain release plan (e.g., plan structure, stakeholder satisfaction, etc.). Assume a large release planning project with about 60 features and 20 stakeholders. Product manager would like to find out the effect of changing the feature effort estimates on the output, and determine the impact of stakeholder weight changes on the output. As in what-if-analysis, the product manager would have to manually change the effort estimates of each feature with various values, and change the stakeholder weights to various levels, and after each change generate a new solution set. The new solution set would have to be compared with the attributes of the baseline release plan. This means making changes and looking at the effects on the outputs hundreds of times, in order to get the overall picture of the impact of the changes. This is possible in theory, but not from the practical point of view. One would have to
spend a few days for such an analysis task. Clearly, the manual what-if analysis approach is unpractical, tedious, and extremely time consuming and hence costly. In addition, the whole process might not be repeatable. Manual what-if analysis would be suitable only for situations when a few changes need to be analyzed. But for large scale analysis such as in the example above, an automated and faster method is needed.

4.2 Problem Statement

For a given release project with a baseline \( S_0 < inputs(x_1...x_m), output < y_1...y_j > \), a prediction function \( F \) is needed that estimates the impact of changing one or more inputs \( x_1...x_m \) on the outputs.

The changes are represented by a set \( X \) with its members as the amount of change made to each of the input parameters relative to the baseline scenario. The input parameters are defined in Table 3-1, and the metric for measuring the relative changes is defined in Section 3.7.1.

\[
X = \{ \Delta \text{Efforts}, \Delta \text{Capacities}, \Delta \text{Stake}_w, \Delta \text{Release}_w, \Delta \text{Criterion}_w \} \quad (4-1)
\]

The impacts on the baseline scenario’s output are denoted by a set \( Y \) consisting of the metrics defined in Section 3.7.2 for measuring the difference between two release scenarios.

\[
Y = \{ \Delta y_1, \Delta y_2, \Delta y_3, \{ \Delta y_{4r} : r = 1 \text{ to } R \}, \{ \Delta y_{5p} : p = 1 \text{ to } P \} \} \quad (4-2)
\]

Then the prediction function \( F \) determines the values of each member of set \( Y \):

\[
Y = F(S_0, X) \quad (4-3)
\]
The function $F$ can be determined using a machine learning technique. Data generated during simulation and sensitivity analysis of a release project by the RPSim II method presented in Chapter 3, can be viewed as a series of release planning scenarios, i.e., input parameter changes and their impact on the output parameters. An illustrative example of such data is shown in Table 3-4. Such data is used to train the prediction function $F$– which generalizes and represents extracted information from the simulation data. From the recommendation point of view, concise and quantitative change impact analysis information can be provided for the product manager. In the next section, the learning technique used to determine the function $F$ is discussed.

### 4.3 Selection of Learning Technique

Since the training data’s label values (release plan output parameter values) are known in advance, the learning task is supervised classification. The classifier maps the attributes (input parameter changes) to the labels that could predict the output of unseen scenarios. A review of supervised classification techniques is provided in [77], and data mining methods used in recommender systems are discussed in [78]. In order to select a suitable classification technique, we considered the following well-known techniques: k-Nearest Neighbours (kNN), Decision Trees, Rule induction, Support Vector Machines (SVM), and Artificial Neural Networks (ANN). The relevant literature was reviewed and a pilot case study conducted to compare the prediction accuracy and training time of each technique for an example training dataset.

Cross-validation [79] is a method for assessing how accurately a prediction model performs with previously unseen data, i.e. data not used for training the model. In this
case this is a situation when attribute values of a release planning scenarios are known, then the model would estimate the values for the labels. This involves partitioning the data into two parts: a training set and a testing set. The training set is used to generate the prediction model and the testing data is used to evaluate the accuracy of the learned prediction model. Multiple rounds of cross-validation could be done, and in each round a different portion of the data is sampled for the testing set. The accuracy of the model is determined as the average of the validation results.

An example release planning project was used for the pilot case study which consists of 15 planning objects, 4 resource types, 3 stakeholders, 2 planning criteria, and the planning scope was for the next 2 releases. First, sensitivity analysis (OAT-SA and AT-SA) and simulation method described in Chapter 3 was applied, and all the input parameters were considered. A total of 46 scenarios were generated, and the output metrics were calculated. The data was recorded in a spreadsheet and imported into a machine learning and data mining tool called RapidMiner⁴. Five machine learning methods mentioned earlier were trained on the imported data. For k-NN a k value of 8 was used, and the default learning parameter values were used in the other cases. Cross-validation method of RapidMiner was used to evaluate the accuracy of each learning method. Stratified sampling was used to randomly choose 20% of data for testing, and the remaining 80% for training. This is repeated 10 times, and the average accuracy of the learning methods, for all output metrics (labels), was recorded as presented in Table 4-1. From the practical point of view, the test data represents future changes to the baseline

⁴ http://rapid-i.com/content/view/181/196/
release planning project. Therefore, the cross-validation results help us determine the
learning method most suitable to predict future changes by the product manager.

From accuracy point of view, the candidate methods for our case are SVM and
ANN since they both perform well compared to the rest. Since the accuracy of the
methods is very similar, other criteria were used to determine the suitable method. We
selected SVM since training it was faster than the ANN. The efficiency of SVM trainers
is more noticeable for larger data sets and this was true in the large case study presented
in Chapter 7 later on. Accuracy of a SVM can be further improved with an optimal
choice of training parameter settings, as discussed in Section 4.5.2.

**4.4 Support Vector Machines**

SVM, first introduced in the early 1970s [80], is a widely used supervised data
classification technique. A detailed description of SVM is given in [81]. A SVM
classifier finds a hyper-plane separating the data with as much ‘margin’ as possible. The
margin refers to the side of hyper-planes that separate the data classes.

<table>
<thead>
<tr>
<th>Learning Technique</th>
<th>Average Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Vector Machine</td>
<td>60.6</td>
</tr>
<tr>
<td>Artificial Neural Network</td>
<td>58.0</td>
</tr>
<tr>
<td>Decision Tree</td>
<td>56.6</td>
</tr>
<tr>
<td>K-NN (k=8)</td>
<td>56</td>
</tr>
<tr>
<td>Rule Induction</td>
<td>55.3</td>
</tr>
</tbody>
</table>
For a given training data with attributes \( \{x_i \ldots x_n\} \) and labels \( \{y_i \ldots y_n\} \), the learning problem is formulated as an optimization problem to be solved by SVM [80]:

\[
\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i , \quad C > 0
\]

subject to \( y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i, \quad \xi_i \geq 0 \) \hfill (4-4)

\( C \) is the penalty of the error, and is defined by the user. The function \( \phi \) maps the input space into a higher dimension feature space in which a separating hyperplane is found with maximum margin, while minimizing the margin error (i.e. minimizing the points with \( \xi_i \neq 0 \)). \( K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j) \) is the kernel function that could be linear or non-linear. In this case, we use the non-linear radial basis function (RBF) with a kernel parameter \( \gamma \):

\[
K(x_i, x_j) = e^{\frac{-||x_i - x_j||^2}{\gamma^2}} \hfill (4-5)
\]

The relation between the attributes and labels in our data is non-linear, and therefore, the RBF can handle this.

<table>
<thead>
<tr>
<th>Scenario (Example)</th>
<th>Attributes</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_1 )</td>
<td>0.2111</td>
<td>0.1758</td>
</tr>
<tr>
<td>( S_2 )</td>
<td>-0.0051</td>
<td>0.3673</td>
</tr>
<tr>
<td>( \ldots )</td>
<td>( \ldots )</td>
<td>( \ldots )</td>
</tr>
<tr>
<td>( S_i )</td>
<td>0.2395</td>
<td>0.1094</td>
</tr>
</tbody>
</table>

Table 4-2: Training data format
Some benefits of the SVM are discussed in [77] and [78]. SVM is suitable for this learning task due to its ability to consider high-dimensional input data (e.g. release planning projects with a large number of features). Also, unlike ANN and decision trees, SVM does not have the local minima problem: the solution to an SVM is unique and global. Global refers to the fact that the objective function does not take a lower value at any other point in the possible solution space of the optimization problem, Eq. (4-4). The SVM solution is global since training a SVM involves solving a convex quadratic programming problem, which has the property that its solution is global. Also, when the objective function is strictly convex then solution is guaranteed to be unique [82]. As SVM needs fewer learning parameters to be selected, it is very attractive for us. A

<table>
<thead>
<tr>
<th>Attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta$effort(1) … $\Delta$effort(N)</td>
</tr>
<tr>
<td>$\Delta$capacity(1, 1) … $\Delta$capacity(R, K)</td>
</tr>
<tr>
<td>$\Delta$stake_w(1) … $\Delta$stake_w(P)</td>
</tr>
<tr>
<td>$\Delta$release_w(1) … $\Delta$release_w(K)</td>
</tr>
<tr>
<td>$\Delta$criterion_w(1) … $\Delta$criterion_w(Q)</td>
</tr>
</tbody>
</table>

**Table 4-3: Training data attributes**

<table>
<thead>
<tr>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta$Release Plan Structure: $\Delta y_1$</td>
</tr>
<tr>
<td>$\Delta$Stakeholder Feature Points: $\Delta y_2$</td>
</tr>
<tr>
<td>$\Delta$Postponed Features: $\Delta y_3$</td>
</tr>
<tr>
<td>$\Delta$Resource Consumption: $\Delta y_{4_1} \ldots \Delta y_{4r}$</td>
</tr>
<tr>
<td>$\Delta$Stakeholder Excitement: $\Delta y_{5_1} \ldots \Delta y_{5p}$</td>
</tr>
<tr>
<td>$\Delta$Combined Output Impact: $\Delta Y$</td>
</tr>
</tbody>
</table>

**Table 4-4: Training data labels**
method for selecting the right parameters is discussed later on. SVM results are also stable and reproducible.

Next, we discuss in detail how SVM is applied to generate change impact recommendations for the product manager from the RPSim II's simulation data.

4.5 Procedure

In this section, steps taken to apply SVM for the prediction task are discussed.

4.5.1 Step 1: Data Preparation and Label Discretization

As mentioned before, the data to be used for learning is from the simulation and change computations produced earlier from the RPSim II method. The training data consists of a large number of stored simulation scenarios (examples), where each scenario consists of the changes in the input parameters (attributes) with respect to the baseline release plan, and the corresponding change in the output parameters (labels). The change computations for input (Section 3.7.1) and output parameters (Section 3.7.2 and 3.8) were discussed in detail earlier. Both the attributes and the labels are real-valued. The original data format is illustrated in Table 4-2. The possible attributes of the data are described in Table 4-3. The labels of the data are shown in Table 4-4.

Before the data is used for training the classifiers, the label values in data are discretized, i.e. their continuous values are transformed to discrete ones. Although SVM can handle continuous data, discretization significantly improves the classification performance of the trained SVM classifiers [83].

The discretization method employed here is described below.

1. Among all the label values, find the minimum (MIN) and maximum (MAX) value.
2. Take the floor of the min value, and the ceiling of the max value.

3. Generate the discretization classes as follows. Each class has a lower and an upper bound value. A special class of zero is always added. The class interval (WIDTH) is specified by the user.

   Let Classes be a list of classes, and i be a real variable.

   Loop until $i > \text{MIN}$
   \[
   \text{Lower} = i = i - \text{WIDTH} \\
   \text{Upper} = i \\
   \text{Add to Classes -> new class(lower, upper)}
   \]

   Loop until $i < \text{MAX}$
   \[
   \text{Lower} = i \\
   \text{Upper} = i + \text{WIDTH} \\
   \text{Add to Classes -> new class(lower, upper)}
   \]

4. Assign a discrete class to each value, of each label type in the simulation data.

4.5.2 Step 2: Kernel Parameters Selection

The radial basis function kernel chosen for SVM has two parameters (C and $\gamma$) as discussed earlier. The values of these parameters are given by the user and affect the accuracy of the SVM classifier. However, we do not know which parameter values are suitable for our data in advance. We follow the guidelines in [84] to perform a grid search for a good pair of $C$ and $\gamma$ values, and use 10 fold cross-validation to measure the prediction accuracy of each classifier. The search involves looking for pairs of values for $C$ and $\gamma$ that give the best cross-validation accuracy. Prediction accuracy measurement of a classifier was explained in Section 4.3.

A kernel parameter selection feature is implemented in the SVM library called LIBSVM [85] and is used here.
4.5.3 Step 3: Model Training and Recommendations

Once a good pair of kernel parameters is found, those $C$ and $\gamma$ kernel parameter values are used for the final model training. The amount of training time depends on the number of examples in the dataset, and the computer used. SVM is one of the most efficient [78] supervised classifiers used in recommender systems. This is certainly true in our case. The SVM library used in the case study presented in Chapter 7 generated an accurate model in about a minute.

The accuracy of the model is determined using 10 fold cross-validation. The accuracy of the prediction model is provided to the product manager as an indication of the reliability of the recommendation.

The generated SVM model generated could be interpreted as a combination of $M$ classifiers with $M = 4 + R + P$. The four classifiers are for the labels $\Delta y_1, \Delta y_2, \Delta y_3,$ and $\Delta Y$. $R$ is the number of resource types and the number classifiers trained depends on this. Similarly, the number of classifiers trained for stakeholder depends on the number of stakeholders in the release planning project, i.e. $P$. Each classifier $m$ maps the input parameter changes to one of the labels (output parameter changes) listed in Table 4-4. This allows us to provide both high level as well as fine-grained recommendations to the product manager. For instance, consider a future scenario where the product manager is tinkering with the release plan project’s settings. The effort estimates of certain features are changed, and/or resource capacity values are altered. These changes will affect various aspects of the baseline release plan. The recommendation system will use the SVM classifiers to instantly predict the following:
1. By how much is the baseline release plan structure expected to change? The higher the value the more the current release plan structure is expected to change.

2. Predicted change in the stakeholder feature points. An increase indicates improved plan quality, and is considered a good change.

3. The expected impact on the number of postponed features. If fewer features are getting postponed than before, then this is considered as an improvement to the baseline plan.

4. The predicted change on the resource consumption of each resource type. If the consumption rate is going to increase, then more resources are being utilized and this a good change as far as resource utilization is concerned.

5. The expected ‘excitement’ of each stakeholders of the project. An increase in this metric indicates a greater satisfaction of a certain stakeholder by changes just made to the baseline plan.

6. The predicted overall impact on the output parameters. This metric is a weighted combination of the above predictions, the calculations of which were described in section 3.8.1.

The above predictions are provided to the product manager proactively as recommendations in the context of the release planning tool. Even if the product manager is not actively looking for the impact of the changes he/she has just made, the proactive recommendations could bring to light any unwanted impact on output parameters. This could save time for the product manager as unforeseen consequences are avoided.
4.5.4 Model Update

The SVM prediction model is updated if a new baseline release plan is adopted, since the change computations and hence the learned model is relative to a certain baseline plan. The extent of the update process depends on the extent of change in the release planning project (i.e. altering the parameters of the EVOLVE II SRP model).

For this, we first consider the type of changes described below. The types are ordered in increasing degree of change:

- **Level 1 Change**: Altering the numerical value of feature effort estimates, resource capacities, release weight, stakeholder weights, or criteria weights.

- **Level 2 Change**: Adding a new or deleting an existing feature, resource type, constraint, or stakeholder.

- **Level 3 Change**: Change of settings that are usually determined once at the beginning of the project, such as: number of releases, number of criteria, and stakeholder voting.

If there is a level 1 change and a new release plan is generated by the product manager, the simulation process (Chapter 3) will not have to be repeated which the most time is consuming. We only have to re-calculate the sensitivity indices, and perform the SVM learning again with the newly generated plan as the new plan with which to compare the other already existing learning examples. This can usually be done on the fly in a matter of seconds. A level 1 change actually improves the prediction models’ accuracy since there is a new scenario (learning example) that can be used to update the model.
If a new release plan is generated and adopted after level 2 or 3 changes, then the sensitivity indices and the prediction model become invalid. The simulation and analysis needs to be repeated, i.e. performing the RPSim II methods first and then performing the three steps of learning discussed above. This is because such changes are major and completely alter the nature of the release planning project and the baseline plan. But such changes do not happen too often, as level 3 settings are usually done only once during the initial setup of the project.

4.6 Summary

In this chapter we provided the design and application of a recommendation method for predicting the impact of release plan changes. The learning technique is support vector machine with a non-linear (radial basis function) kernel. The data comes from the simulation and sensitivity analysis method explained in Chapter 3. The aim is to assist the product manager in staying informed of the effect of input changes to avoid any unforeseen consequences such as stakeholder disappointment or delay of an important feature.

The method is fully automatic and consists of three steps. The data is pre-processed; to prepare it in the correct format, and discretize the label values. In the second step, grid search and cross-validation is used to find a good pair of values for the C and gamma parameters of SVM, which is then used in the last step to train a set for SVM classifiers, one for each label in the data. The label refers to the metrics devised for the measurement of the characteristics of a release plan.
The next chapter provides another type of recommendation for the product manager to assist during release planning – recommending actions to take for achieving certain release targets.
5.1 Motivation

The goal of software engineering decision support systems (DSS) is to facilitate better project management in the form of enhanced communication between involved parties, increased productivity, time savings, improved customer satisfaction, and better decision quality [86]. Decision making can be done reactively or proactively. In the reactive type, actions are triggered as problems appear. In contrast, being proactive implies planning ahead and anticipating problems that may arise, thus eliminating issues before they appear. Recommendation techniques in the context of DSS can help to achieve this.

In the previous chapters, we provided three recommendation methods that address the information needs of a product manager; informing the product manager with the most important input parameters and helping in determining trends in release plan metrics due to input variations (Chapter 3), and providing a prediction model that goes beyond the traditional what-if analysis and can estimate the impact of input parameter changes (Chapter 4). These methods help with the proactivity by providing project awareness, which could be used to improve the current release plan through discovery, for instance tinkering with stakeholder weights to see if it improves a stakeholder's satisfaction.

We could see the previous task from another perspective. We want to improve a stakeholder's satisfaction, but need to know how to achieve this target. In such cases, the target is known a priori and a recommendation method is needed that guides the manager in achieving the target. In this chapter, we provide such a recommendation method that
assists a manager beyond just *awareness* and by answering a question of *insight:* What are the next actions that could help me achieve release target(s)?

Formally, for a given release project with a baseline $S_0 \langle \text{inputs}(x_1...x_m), \text{output}<y_1...y_j> \rangle$, a function $M$ is needed that recommends a set of changes to the baseline scenario that could help achieve the given release targets $Y$.

### 5.2 Recommendation Method

A utility-based recommendation technique matches a user’s needs with a set of items to be recommended [11]. This involves calculating the utility of each item for the user, ranking the items based on their calculated utility, and recommending the top-N items to the user. The user’s needs or preferences could be inferred implicitly or by asking the user to express them explicitly. The item utility calculation is based on the item attributes as well as domain knowledge.

The recommendation method described in this chapter to assist in achieving certain release plan targets is a utility-based technique. The user is the product manager who can express some targets from various perspectives, such as increased satisfaction of a certain stakeholder. The set of items to be recommended are the release planning scenarios generated from the sensitivity and simulation activity (Chapter 3). The utility of each scenario depends on how much it meets the required target(s) and the degree of changes that needs to be made to the input parameters in order to achieve the target(s). The domain knowledge of strategic release planning is employed to determine what constitutes ‘achieving a target’. The ability to factor non-item attributes and domain
knowledge into the recommendation generation is a major benefit of utility-based recommendation techniques [11].

Next, we detail the recommendation method steps.

5.3 Procedure

The recommendation method involves three steps. We first capture the product manager’s targets, based on which the release planning scenarios are ranked, and finally the top-N scenarios are suggested to the user.

5.3.1 Step 1: Capturing Release Plan Targets

In this step, the product manager explicitly expresses one or more desired targets with respect to the current baseline release scenario: $S_0<inputs(x_1...x_n), output<y_1...y_n>)$. The targets are expressed from one or more perspectives that directly map to the metrics defined in section 3.7.2 for changes in output parameters of a release scenario. The targets are described in Table 5-1.

<table>
<thead>
<tr>
<th>Targets: $Y$</th>
<th>Related Metric</th>
<th>Related Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improve stakeholder feature points</td>
<td>$\Delta y_2$</td>
<td>Section 3.7.2.2</td>
</tr>
<tr>
<td>Improve postponed features</td>
<td>$\Delta y_3$</td>
<td>Section 3.7.2.2</td>
</tr>
<tr>
<td>Improve resource consumption of resource $r$, for $r=1$ to $R$ (number of resources types in the project)</td>
<td>$\Delta y_{4r}$</td>
<td>Section 3.7.2.4</td>
</tr>
<tr>
<td>Improve stakeholder excitement of stakeholder $p$, for $p=1$ to $P$ (number of project stakeholders)</td>
<td>$\Delta y_{5p}$</td>
<td>Section 3.7.2.5</td>
</tr>
</tbody>
</table>
The recommendations generated will be based on the selected target perspectives. For instance, if only the first target is selected, then metric $\Delta y_2$ will be used for calculating the utility of the release planning scenarios.

The product manager also provides a weight to each selected target perspective from a scale of 1 (least important) to 5 (most important). A weight of zero is automatically assigned to a perspective not selected by the manager.

5.3.2 Step 2: Selection of Scenarios

In this step, the product manager selects which release planning scenarios are to be ranked. The first option is to select the scenarios generated during the simulation and sensitivity analysis as explained in Chapter 3.

In the second option, the product manager generates new release planning scenarios. The release planning generation follows similar simulation steps as explained in Chapter 3, except for three differences. The first difference is that in order to calculate the output metrics we utilize the SVM classifiers trained (Chapter 4) instead of running ReleasePlanner™. The second difference is that each feature’s effort estimate, and each resource’s capacity is changed only once, whereas in the simulation the product manager could specify the number of times to change each effort estimate and resource capacity. The third difference is that only one-at-a-time type of simulation is considered here.

Using the SVM classifiers for determining the output metrics of the scenarios have the benefit that we can generate hundreds of scenarios in a matter of seconds. Determining these metrics by running the ReleasePlanner™ would take much longer. However the trade-off is that the output metrics are now estimations and not actual
measurements. Therefore, this needs to be factored in when the scenarios are ranked as explained in Step 3.

5.3.3 Step 3: Ranking of Scenarios

First, utility of each scenario \( S_i \in S \), set of scenarios selected in step 2, is calculated in this step according to the following:

Let \( Y \) be the set of the selected targets to be achieved, and a target \( y_j \) can be:

\[
y_j \in \{ \Delta y_2, \Delta y_3, \{ \Delta y_{4r}, r = 1 \text{ to } R \}, \{ \Delta y_{5p}, p = 1 \text{ to } P \} \}
\]

and \( w_j \) is the normalized weight provided by the PM for each target,

\[
w_j = \frac{w_j'}{\sum_{j=1}^{J} w_j'}
\]

and \( z_j \) determines if the corresponding metric for a target \( y_j \), in a scenario \( S_i \) is an improvement over the baseline scenario \( S_0 \),

\[
z_j = \begin{cases} y_j & \text{if } y_j > 0 \\ 0 & \text{if } y_j \leq 0 \end{cases}
\]

and \( a_j \) determines if the metric for target \( y_j \) is an estimation from a SVM classifier or an actual measurement. \( c_j \) is the prediction accuracy of classifier for \( y_j \) (found during a 10-fold cross validation in section 4.5.2 – kernel parameter selection) and has a range of \([0, 1]\),

\[
a_j = \begin{cases} c_j & \text{if } y_j \text{ is based on prediction from classifier } c_j \\ 1 & \text{if } y_j \text{ is from actual measurement} \end{cases}
\]

then the utility of the scenario \( S_i \) is calculated as:

\[
u(S_i) = \frac{1}{\sum_{m=1}^{M} |\Delta x_m|} \cdot \sum_{j=1}^{J} z_j \cdot w_j \cdot a_j
\]
where $\Delta x_m$ denotes the change in the \textsuperscript{m}th input parameter that was made in scenario $S_i$. The input parameter change metrics were described in section 3.7.1. From the above equation, the utility of a scenario $S_i$ is directly proportional to each output metric’s magnitude $z_j$, and weight $w_j$. The utility is indirectly proportional to input parameter metrics $\Delta x_m$. Overall, this gives preference to the scenarios that are close to the desired targets but could be achieved by making the smallest changes to the input parameters.

Next, the scenarios in $S$ are ranked based on their calculated utility value for the product manager.

\textbf{5.3.4 Step 4: Recommendations}

Once all the scenarios are ranked based on their utility (degree of target achievement), the top-N scenarios are recommended to the product manager. In each scenario the following sets of recommendations are provided to the product manager:

1. What are the improvements over the baseline release plan?
2. What are the trade-offs, if any?
3. How to achieve these targets?

For the first two recommendations, let’s consider two sets of metrics;

$$A = \{\Delta y_2, \Delta y_3, \{\Delta y_{4r}: r = 1 \text{ to } R\}, \{\Delta y_{5p}: p = 1 \text{ to } P\}\}$$

$$B = \{\Delta \text{capacity}(r,k): r = 1 \text{ to } R, k = 1 \text{ to } K\}$$

Set $A$ consists of the metrics for output parameter changes for a release planning scenario (section 3.7.2), and set $B$ consists of the metrics for the resource capacity changes for each resource $r$ and release $k$ (section 3.7.1).
Then, an ‘improvement’ represented by a scenario with respect to the baseline release plan could be:

a. An increase in one or more metrics in set A, and/or

b. A decrease in one or more metrics in set B.

Achieving a desired target with respect to the current baseline release plan usually comes at a cost. For instance, the number of postponed features has decreased but at the cost of increasing the capacity of resource type \( r \). Such trade-offs could be:

a. A decrease in one or more metrics in set A, and/or

b. An increase in one or more metrics of set B.

Finally, the targets could be achieved by increasing or decreasing the input parameter values of the baseline release plan project, where the new value of an input parameter \( x_j \) is:

\[
x'_m = (1 + \Delta x_m) \times x_m, \quad for \ \Delta x_m \neq 0
\]

(5-6)

\( \Delta x_m \) is the amount by which the input parameter \( m \)’s value was changed during the simulation for a scenario \( S_i \). In other words, this recommendation is achieved by comparing the input parameter values of the baseline release plan with the scenario \( S_i \).

The product manager could evaluate each top-N scenario recommended, by weighing the targets that could be achieved and the trade-offs. Once a suitable scenario \( S_i \) is selected, that scenario could be chosen to become the new baseline release plan.

5.4 Illustrative Example

In Section 3.10, a simple release planning example with artificial data was used to demonstrate application of RPSim simulation and sensitivity analysis method. We had
also assumed that four release scenarios had been created after the simulation step of RPSim II as shown in Table 3-3. Then, input and output change metrics for each scenario were calculated as shown in Table 3-4. In this section, we continue the illustrative example from Section 3.10 to demonstrate how the method described in Section 5.3 could be applied to the simulation data.

**Step 1:** We assume that a product manager has two release planning targets; a) improve excitement analysis metric of stakeholder Stake1 \( (\Delta y_{5,\text{Stake1}}) \) and the weight for this target is \( w'_{5,\text{Stake1}} = 5 \), and b) improve excitement analysis metric of stakeholder Stake2 \( (\Delta y_{5,\text{Stake2}}) \) with a weight \( w'_{5,\text{Stake2}} \) of 2. In this case, the first target is considered more important than the first. **Step 2:** All the four scenarios are selected for analysis: set \( S = \{S_1, S_2, S_3, S_4\} \). **Step 3:** Utility of each scenario \( S_i \) in \( S \) is calculated, and then the scenarios ranked according to which scenario meets the given targets. Next, we show how utility of scenario \( S_1 \) and \( S_2 \) are calculated in detail.

First, the weights given to each target is normalized according to Eq. (5-1):

\[
w_{5,\text{Stake1}} = \frac{w'_{5,\text{Stake1}}}{w'_{5,\text{Stake1}} + w'_{5,\text{Stake2}}} = \frac{5}{5 + 2} = 0.7143
\]

\[
w_{5,\text{Stake2}} = \frac{2}{5 + 2} = 0.2857
\]

To calculate the utility of \( S_1 \) for this case, while referring to the metrics of \( S_1 \) in Table 3-4, we use Eq. (5-2) to determine that \( z_{5,\text{Stake1}} = 0.3333 \) and \( z_{5,\text{Stake2}} = 0 \) since \( \Delta y_{5,\text{Stake2}} = -0.2 \). Since the scenario’s metrics are from actual measurements, \( a_{5,\text{Stake1}} \) and \( a_{5,\text{Stake2}} \) are both 1 according to Eq. (5-3). Then the utility of scenario \( S_1 \) is calculated using Eq. (5-4):
\[ u(S_1) = \left( z_{5,\text{stake1}} \cdot w_{5,\text{stake1}} \cdot a_{5,\text{stake1}} \right) + \left( z_{5,\text{stake2}} \cdot w_{5,\text{stake2}} \cdot a_{5,\text{stake2}} \right) \]

\[ u(S_1) = \frac{(0.3333 \cdot 0.7143 \cdot 1) + (0 \cdot 0.2857 \cdot 1)}{0.1 + 0} = 2.3808 \]

For scenario \( S_2 \), we determine that \( z_{5,\text{stake1}} = 0.5 \) and \( z_{5,\text{stake2}} = 0.6 \). Again, since the scenario’s metrics are from actual measurements, \( a_{5,\text{stake1}} \) and \( a_{5,\text{stake2}} \) in \( S_2 \) are also 1. Then the utility of scenario \( S_2 \) is calculated using Eq. (5-4):

\[ u(S_2) = \frac{(0.5 \cdot 0.7143 \cdot 1) + (0.6 \cdot 0.2857 \cdot 1)}{0.2 + 0} = 2.6439 \]

In similar manner, the utility of \( S_3 \) and \( S_4 \) are calculated as shown in Table 5-2. \( S_3 \) does not achieve the targets at all; therefore its utility is zero and is not ranked. We can see that scenario \( S_2 \) meets the given targets the most. Therefore as part of Step 4 of procedure from Section 5.3, scenario \( S_2 \) would be on the top of recommended scenarios for the product manager. Then the action recommendation in this case is: The excitement metric of Stake1 and Stake2 can be improved from current level (as in baseline scenario S0), by increasing the capacity of Res1,R1 by 20% - since we can see that in Table 3-4 that the outputs of \( S_2 \) is due to changing capacity(Res1,R1) by 20%. Besides achieving the given targets, other improvements include increased SFP (\( \Delta y_2 \)), reduction of

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Rank</th>
<th>Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>( S_2 )</td>
<td>1</td>
<td>2.6439</td>
</tr>
<tr>
<td>( S_1 )</td>
<td>2</td>
<td>2.3808</td>
</tr>
<tr>
<td>( S_4 )</td>
<td>3</td>
<td>0.4405</td>
</tr>
<tr>
<td>( S_3 )</td>
<td>-</td>
<td>0.0</td>
</tr>
</tbody>
</table>
postponed features ($\Delta y_3$), and increased resource utilization ($\Delta y_{4,Res1}$). The only trade-off is that more of resource Res1 is needed for release R1.

The example given here is simplistic. Real life release planning projects could have tens of input parameters and their simulation could contain hundreds of scenarios. The method described here would help to efficiently, and quickly analyze all the scenarios to find the ones that meet the given targets. Furthermore, the large volume of data involved in such type of recommendation generations emphasizes the need for automated tools. Tool support for the recommendation system, which includes the method described in this chapter, is described next in Chapter 6.

5.5 Related Work

Du defines *backward software release planning* in her thesis [87] as “A type of pro-active software release planning that starts with specifying characteristics of desired release plans in the solution space. Based on this specification, it tries to find release planning scenarios in the problem space which would meet the specified characteristics.” To achieve a release planning goal, input parameters are changed, beginning with the most sensitive parameters, and the output is checked. This step is repeated for all the possible input parameter changes.

The only goal a user can set is the assignment of a feature to a certain release. The method uses trial-and-error, to check if different input parameter changes would achieve the goal. To illustrate this, assume that the method identifies 3 input parameters (a, b, c) as ‘sensitive’. Assuming each input parameter values can be changed by 1 unit, a
total of 7 different parameter changes is possible: \( \{a', b', c'\}, \{a' + b'\}, \{a' + c'\}, \{b' + c'\}, \) and \( \{a' + b' + c'\} \). The number of possible combinations is given by:

\[
N = \sum_{k=1}^{P\times C} \binom{P \times C}{k} = 2^{P\times C} - 1 \tag{5-7}
\]

where \( P \) is the number of sensitive input parameters and \( C \) is the number of changes per input parameter. So in this case, there are \( N = 7 \) possible combinations. New release planning scenarios are generated by changing different combinations of input parameter changes and checked if the goal is achieved.

Scalability is a major issue with this approach to backward release planning. Assume that 4 input parameters are identified as sensitive \((P=4)\), and 4 changes per input parameter are possible \((C=4)\). The number of possible combination of input parameter changes is: \( N = 65,535 \) changes. If it takes 10 seconds to generate one release plan and investigate if the plan helps us to reach the goal, it would take 7 days and 14 hours to investigate all the 65,535 possible scenarios. A goal might never be achieved by these changes in the worst case.

5.6 Summary

In this chapter, design and application of a method was provided which assists a product manager with achieving certain release targets specified by the user such as improving satisfaction of a stakeholder, or decrease the number of postponed features. More than one target can be specified at a time, where each target is given an importance weight. The method then analyzes the release planning scenarios generated during the sensitivity and simulation analysis, and ranks them. The rank or utility score of each scenario
depends on a number of factors; how much it helps in achieving the specified target(s),
the trade-offs of making the changes need to be made to achieve the targets, and
magnitude of these changes. The method can also generate scenarios systematically,
where the output metrics of the scenarios are predicted using the prediction model
presented in Chapter 4.
Chapter Six: **Tool Support**

6.1 **Introduction**

In this chapter, we will describe the design and implementation of the proposed recommendation methods as a single comprehensive tool. The tool integrates the simulation, sensitivity analysis, and the recommendation methods into an automated process. Having tool support allows users to truly evaluate the concepts and effectively apply the methodologies in real life situations. It is even more useful to integrate the recommendation tools and techniques into existing work environments of requirement engineers (e.g. as plug-ins). This would ensure fewer context switches and a smoother and light-weight usage experience [22]. Existing development environments, such as IBM’s Jazz\(^5\) platform and Microsoft’s Team Foundation Server (TFS)\(^6\), have features for gathering and presenting large amount of software project data. But the information and their analysis are targeted towards developers, not product managers. The recommender tool described in this chapter addresses these gaps. The tool is implemented as a plug-in for Microsoft Visual Studio (VS)\(^7\) and interfaces with TFS.

The main features of the tool are as the following.

- **F1.** Remotely connect to ReleasePlanner\(^{TM}\), and retrieve a list of a user's projects.
- **F2.** Create a planning project by importing a ReleasePlanner\(^{TM}\) project data.
- **F3.** Provide an interface between the tool and Visual Studio and Team Found Server in the form of a plug-in.

---

\(^5\) http://jazz.net/
\(^7\) http://www.microsoft.com/visualstudio/en-us
F4. Create a planning project using a TFS work items and user inputs.

F5. Save a project for permanent storage.

F6. Open an already existing project.

F7. Display a project's details for the product manager.

F8. Create a new baseline release solution set.

F9. Create, save, and open a simulation.

F10. Calculate the sensitivity indices of the input parameters, and visually display it to the user.

F11. Visually display the impact of changes of a certain input parameter on one or more output metrics.

F12. Analyze the simulation data to train a SVM classifier for each output metric.

F13. Use a trained SVM classifier to predict impact of input parameter changes.

F14. Rank the release planning scenarios based on how much they achieves the given product manager's target(s).

F15. Display details of a ranked scenario to product manager detailing what changes would achieve which targets, by how much, and what are the trade-offs.

Next, the overall architecture of the tool is provided and the platform on which the tool is developed is explained.
6.2 Development Platform

A plug-in is a software program that integrates with another (larger) software application with the goal of adding new functionality to it. IDEs such as Eclipse\(^8\), NetBeans\(^9\), and Microsoft Visual Studio have plug-in architectures that allow third-party tools to be added to them, as desired by the user. Plug-in based software architectures have several benefits, such as light-weight core application that is easy to manage and maintain, avoiding feature over-crowding, and most importantly allowing developers to add new features to an application in the future [88], [89]. The strategic release planning recommendation system proposed in this work is implemented as a plug-in for Microsoft Visual Studio 2010.

Visual Studio 2010 is an application lifecycle management (ALM) tool from Microsoft, which among other components includes an integrated development environment, Visual Studio Team Explorer (VSTE), and Team Foundation Server. The popular Visual Studio IDE is used by developers during coding, compiling and testing software projects. TFS is the centerpiece of Visual Studio that functions as an information hub and facilitates communication between the components. Using an MS SQL database server, TFS stores the entire team project’s information such as requirements, source code version control details, bug tracking, test case and build management information, and other project properties. Third party applications can

\(^8\) http://www.eclipse.org
\(^9\) http://www.netbeans.org
utilize services provided by TFS using the provided application programming interface\textsuperscript{10} (API). The functionality of Visual Studio can be extended by developers through plug-ins. Plug-ins can add new or customized functionality by exploiting the extensibility features of VS components. Numerous extensions for Visual Studio can be found in the Visual Studio Gallery\textsuperscript{11}.

A "MSF process template" in VSTE defines types of work items, rules, policies, security groups, and queries used in team development. Process templates are provided that are suitable for CMMI, Agile, and Scrum development models. Even if one of these templates is used, the user still has to specify manually which feature or requirement will be implemented in which release. There is a need for Visual Studio plug-ins that assist with formal strategic release planning.

Previously, we had described the integration of ReleasePlanner\textsuperscript{TM} with Visual

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{07026-Figure6-1.png}
\caption{SRP-Plugin architecture in context of Visual Studio 2010 and ReleasePlanner\textsuperscript{TM}.}
\end{figure}

\textsuperscript{10} \url{http://msdn.microsoft.com/en-us/library/bb130146}
\textsuperscript{11} \url{http://visualstudiogallery.msdn.microsoft.com}
Studio in the form of a Visual Studio plug-in (called SRP-Plugin) and a customized process template [90]. This integration brought together the rich application lifecycle management already provided in Visual Studio with the powerful release planning capability of ReleasePlanner™. SRP-Plugin demonstrates that tools implemented as plug-ins for widely used development platforms (such as Visual Studio) help to increase efficiency of the development process. The plug-in augments the rich Visual Studio environment with advanced release planning capabilities which result in better release planning quality, increased productivity and enhanced communication among project stakeholders.

The plug-in, developed in C#, connects the Visual Studio platform and ReleasePlanner™ and hence has two types of functions: integrating with the Visual Studio's IDE and Team Foundation Server, and providing an interface to use the services provided by ReleasePlanner™. The overall architecture of the SRP-Plugin and its central role is shown in Figure 6-1.

The recommender tool presented in this chapter extends SRP-Plugin to not only provide release planning capability but also proactive recommendation features for a product manager. In the next sections, the design of the tool is described.

6.3 System Architecture of SRP-Plugin 2.0
SRP-Plugin 2.0 is designed in a modular manner, consisting of 6 modules. Each module performs certain functions independently, while the other functionalities are achieved through interactions and coordination with one or more other modules. The modules and
their relationships are demonstrated in Figure 6-2. The role of each module is explained in the following subsections.

6.3.1 ReleasePlanner\textsuperscript{TM} Interface

ReleasePlanner\textsuperscript{TM} is a web based application that allows users to perform strategic release planning. When developed initially, it was mainly intended to run independently and not within another application environment. The ReleasePlanner\textsuperscript{TM} Interface module of SRP-Plugin 2.0 provides a link to ReleasePlanner\textsuperscript{TM}, and gives us programmatic access to ReleasePlanner\textsuperscript{TM}'s services i.e. project creation, release plan generation, and retrieval of its analysis such as stakeholder excitement analysis. When this interface is invoked in the code, the following sequence of events occurs:

1. Setup a secure remote connection to ReleasePlanner\textsuperscript{TM} using the product manager's login credentials.
2. Call the appropriate functions of ReleasePlanner\textsuperscript{TM} with the inputs.
3. Retrieve the output and parse it in a suitable format for the tool.

6.3.2 Visual Studio Plugin Interface

This interface integrates the tool with Visual Studio's IDE and TFS. It allows the tool to run from within the context of Visual Studio IDE. It also handles the data flow with the TFS. For instance, this interface can retrieve the team project's work items and update it if necessary. Simulation data is also maintained by this interface, which is stored in XML format.
6.3.3 *RPSim II Simulation Module*

The simulation module implements the simulation methodology detailed in Chapter 3. It takes a product manager's inputs for a simulation preparation, and then systematically performs the input parameter changes to generate new release planning scenarios. In each scenario one or more values of the baseline plan input parameters are changed. The
module systematically makes these changes to the baseline plan through the ReleasePlanner™ interface. The output of ReleasePlanner™ is parsed and the output metrics are calculated.

This module also saves the simulation data and meta-data in CSV and XML formats for future use. The module also handles retrieval and display of previous simulations of a project in an appropriate format for the product manager. The use case of this module is further explained in Section 7.3.

6.3.4 Sensitivity Recommender Module

This module implements the sensitivity computations explained in section 3.8, to find the sensitivity index of each input parameter that had been selected during the simulation. A product manager selects the output perspectives based on which the sensitivity is to be calculated and the module analyzes the release plan scenarios generated during the simulation to calculate the sensitivity indices. This recommendation is provided both in textual and visual format.

The module also visually provides the impact of a series of input parameter changes with respect to one or more output metrics. This allows the product manager to observe how some release plan metrics are trending. The applicability of the recommendations from this module is highlighted in Sections 7.4.1 and 7.4.2.
6.3.5 Change Impact Recommender Module

For a given simulation, this module performs a one-time operation of training a set of SVM classifiers. This achieved using the open source LIBSVM library\textsuperscript{12}. The simulation data is pre-processed, the output metric values discretized, and a SVM classifier is trained for each output metric. The library automatically performs a grid search to find an optimal pair of C and gamma values for the SVM classifier, and it also performs a 10-fold cross validation to find the average accuracy of the each classifier. This process was explained in detail in Chapter 4.

Once the classifiers are trained, this module can predict the estimated impact of any future changes in the baseline release plan with respect to one or more output metric. The information is provided instantly to the product manager. Using the classifier, the product manager could generate a series of scenarios or changes to the baseline release plan. The output metrics are then predicted using the respective classifiers, and we do not have to resort to expensive ReleasePlanner\textsuperscript{TM} runs. The applicability of this module is illustrated in Section 7.4.3.

6.3.6 Target Achievement Recommender Module

This module is the implementation of recommendation method, described in Chapter 5, for assisting a product manager in achieving certain release targets, and its applicability is discussed further in Section 7.4.4. The release plan scenarios generated during the simulation are ranked based on how much each fulfils the given targets of the product manager. The module calculates the utility of each scenario, and ranks them (sort in

\textsuperscript{12} http://www.csie.ntu.edu.tw/~cjlin/libsvm/
descending order based on calculated utility). Once a product manager selects a scenario, it visually provides him/her with the information of which targets are achieved, by how much, by which changes to baseline plan, and finally what are the trade-offs associated with the suggested changes.

6.3.7 Recommendation Engine
The recommendation engine utilizes one or more of the three previous recommender modules as seen appropriate for a given context. It gathers the user's query and inputs, calls the necessary recommendation generation module(s), and provides the combined results to the user in either textual or visual format. Hence, this module acts as a coordinator role, and abstracts internal complexities of the recommendation generations from the user. The module has been designed keeping future extensions in mind, i.e. it could utilize any other recommender module added in the future to assist the product managers in some other manner.

6.4 Summary
The tool is developed using C# programming language. It runs within the Visual Studio IDE as a plug-in. The plug-in integrates ReleasePlanner™ with Visual Studio, and provides recommendation functionalities to the product manager within the context of certain project. In this chapter, the overall system architecture and its modular design was presented. Each modules functionality and interaction with other modules was explained.

The tool is meant to help product managers with automated, fast, and concise release recommendations, and in the next chapter we present how the tool can be used
through a case study. The tool’s various GUI screens are also explained in the next chapter.
Chapter Seven: Case Study

Case study is an empirical method for investigating a contemporary phenomenon within their real life context, and is particularly useful method to answer 'How?' and 'Why?' research questions in an exploratory, descriptive, or explanatory manner [91]. In this chapter, we illustrate how the recommendation system presented in this thesis can be applied to assist a product manager during strategic release planning. The case study is setup following the guidelines for conducting case studies in software engineering presented in [92].

The case study design presented in the next section describes the case study type, objective, and research questions. A description of the selected case is provided next, followed by the case study results and analysis. Finally, the threats to validity are discussed.

7.1 Case Study Design

The case study presented is a descriptive case study which involves illustrating a situation or phenomena with in-depth real life examples. We study one project as a whole, therefore this is a single-case holistic type of case study [93]. For the case selection, we needed a release planning project from industry with a large number of planning objects and several stakeholders. The objective is to describe how the recommendation methods and its implementation (SRP-Plugin 2.0) described in thesis can assist a product manager with better release decision making in a real-life release planning context. For this, the following recommendation questions are formulated.

Rec 1. How do the input parameters rank in terms of their sensitivity?
Rec 2. Are there any overall trends of input parameter changes with respect to the output metrics?

Rec 3. What is the predicted impact of some release plan changes? And how accurate is the prediction model?

Rec 4. How do the release planning scenarios rank with respect to achieving certain release planning targets of the product manager? And what are the trade-offs?

The aim of the above case study questions is to provide various types of recommendations for a product manager with respect to an on-going release planning project, i.e. the preliminary data such as planning objectives, effort estimates, resource capacities, and stakeholder details are available along with a baseline release plan. The case study results will describe the applicability of the SRP-Plugin 2.0 tool and its usefulness in this situation.

7.2 Case Description

The case comes from a real life strategic release planning project from the health insurance software application domain. The firm uses the baseline release plan already generated using the EVOLVE-II SRP model implemented in ReleasePlanner™. The recommendation concepts introduced in this thesis are applied at this stage of the project to assist the product manager to better understand the situation, and to make better decisions with respect to release planning.
To maintain confidentiality, the release plan project data has been made anonymous by changing the original project name, requirement details, and stakeholder details.

Details of the release planning project used in this case study are provided below:

- 2 Releases to be planned \{R1, R2\}
- 3 Planning criteria \{Urgency, Value\}
- 3 Resource types \{Res1, Res2, and Res3\}
- 58 Requirements to be prioritized \{REQ-1, ..., REQ-58\}
- 19 Stakeholders \{Stake.1, ..., Stake.19\}
- There are no requirements dependencies
REQ-29 and REQ-35 are pre-assigned to Release 1 (R1)

The stakeholder vote coverage is 88.57% at this stage since some stakeholders have not yet voted on some of the requirements. Regardless of this, a baseline release plan has been generated. For future references, this is called Scenario-0. The screen in Figure 7-1 shows how the SRP-Plugin 2.0 presents the project details and baseline release plan for the product manager.

7.3 Data Generation and Results

The data based on which the recommendation questions are to be investigated is generated using the simulation and sensitivity analysis method RPSim II as presented in Chapter 3.

All input parameters of the project are selected for simulation, and their respective sampling ranges are provided in Table 7-1. In real life use of the tool, a product manager selects the input parameters and determines the sampling ranges as necessary.

<table>
<thead>
<tr>
<th>Table 7-1. Selected input parameters and their sampling ranges</th>
</tr>
</thead>
<tbody>
<tr>
<td>Groups</td>
</tr>
<tr>
<td>Effort (E)</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Resource Capacities</td>
</tr>
<tr>
<td>Stakeholder Weights</td>
</tr>
<tr>
<td>Release Weights</td>
</tr>
<tr>
<td>Criteria Weights</td>
</tr>
</tbody>
</table>
We have selected both the simulation types. For the all-together approach, each change level is considered 30 times. For the one-at-a-time approach, each requirement effort estimate is changed 15 times. Similarly, the resource capacities input parameters are varied 15 times. The simulation settings screen is shown in Figure 7-2.

The tool starts the automatic data generation and analysis process. This is a time consuming process and the duration depends mainly on the size of the planning project. The project used in this case study is considered to be large (58 requirements and 19 stakeholders). However, the whole simulation process is automatic and the product manager does not need to interact once the initial selections are made. As a result, the

Figure 7-2: Screen of the simulation tab of the tool
tool could be kept running during off-peak hours (e.g. overnight), and the results would be ready automatically.

Based on the simulation settings, a total of 1287 scenarios are generated. At this stage, the data has also been analyzed, i.e. the data has been pre-processed and the SVM classifiers have been trained. The other data analysis is performed on the fly such as calculating the sensitivity index of the input parameters. Figure 7-2 also partially shows the raw data of simulation. The product manager has the option to export the data in CSV format and analyze it in other tools such as Microsoft Excel.

7.4 Analysis Results

In the following sections, the recommendations provided to the product manager are discussed with respect to the case study questions.

7.4.1 Rec. 1: Sensitivity of the Input parameters

Data generated from OAT sensitivity analysis is used to rank the input parameters based on their sensitivity score. The calculation steps were discussed in section 3.8. The product manager has the option to see how the inputs’ sensitivity compares from various combinations of output metric perspectives, which are:

- Excitement Analysis of Stakeholders (EA)
- Resource Consumptions (RC)
- Plan Structure (PS)
- Postponed Features (PF)
- Stakeholder Feature Points (SFP)
A screenshot where these recommendations are provided is shown in Figure 7-3. Interpretation of the ranking depends on the selected output metric perspectives. For example, if the product manager is interested in seeing which parameters affect the most the stakeholder excitement, then only the EA would be selected. But if the structure of the release plans and number of postponed features are considered to be relevant for the product manager’s situation, then only the PS and PF metrics would be selected. For cases when we are interested to know the effect of input parameter changes on all aspects of release solution set, then all the output metrics can be selected to perform a comprehensive ranking.
Figure 7-4: Plot of the 10 most and least sensitive input parameters
All the 68 input parameters of the project were selected for simulation and analysis in this case study, and the tool provides recommendations related to all these input parameters. For simplicity, we first consider 20 of the input parameters whose sensitivity indices have been calculated based on a) combined perspective of all output metrics (OVERALL), b) plan structure (PS), and c) postponed features (PF).

Figure 7-4 shows the plot of the sensitivity ranks of these inputs based on the three perspectives. The top portion of the plot shows the 10 most sensitive parameters based on OVERALL perspective, and the bottom portion shows the parameters the 10 least sensitive parameters based on the same perspective. The following observations can be made.

1. Ranking of the input parameters depend on the selected output metrics. For example, the weight of release 1, RW1, is ranked as the third most sensitive from the combined metrics perspective, but ranks very low from the PF metric's perspective. As mentioned before, using the combined metrics perspective gives us a good indication of the overall sensitivity of the input parameters, but this is not suitable for all situations. Other combinations of metrics might be more suitable for the product manager's specific situation. This is something that the manager would decide based on personal judgement.

2. Resource capacities allocated for each release sometimes play an important role on the release plans, and this is true for this project since all the six resource capacity input parameters are among the top 10 most sensitive. A product manager would find it beneficial to know how these parameters compare to each other. We can see that
overall Res1 and Res3 for either R1 or R2 are more sensitive than Res2 from the OVERALL perspective. We can also observe that from the PF perspective, resource capacities for release 2 rank higher than release 1 capacities. This tells us that if the product manager would like to make some changes with regards to the number of features postponed, then the capacities allocated for release 2 should be reconsidered.

3. This project has 19 stakeholders, and usually the weights assigned to each stakeholder by the product manager effects the release plans since the features favored (through voting) by a stakeholder with a higher weight should be assigned to earlier releases if possible. Hence, it is expected that the weights assigned to stakeholders would impact the release plans. However, the sensitivity indices results show that none of the stakeholders' weight has any impact on the overall output, and hence their sensitivity indexes are zero making them the least sensitive input parameters. In the graph, all the 19 stakeholder weight parameters are represented by the one parameter 'Any SW' since they all have zero values. This information would be very practical for the product manager since she would have to spend time trying to vary the weights of different stakeholders in cases when some stakeholder's satisfaction level is to be improved.

4. The weight assigned to release 1 ‘RW1’ is more sensitive to changes than release 2's weight 'RW2' from the overall and plan structure perspective. But RW2 is ranks higher than RW1 from postponed features perspective, implying that the weight
assigned to release 2 should be re-considered if the product manager would like to alter the number of features postponed.

5. Similar to the previous observation, we usually expect that the weights assigned to the criteria parameters would be have higher effect on the output. But in this project, these parameters 'CW1' and 'CW2' are among least sensitive parameters.

6. The effort estimates parameters of the requirements labelled as 'REQ-X' are less sensitive compared to the other input parameters. Only two of them are among the 10 most sensitive. It would be useful to compare the sensitivity of the 'REQ-X' parameter types separately.

The radar plot in Figure 7-5 demonstrates the ranking of the requirement effort estimate parameters from the overall perspective and the five individual output metric perspective. The stakeholder excitement plot has been scaled down by factor of 4 (divided by 4) for better visibility. Once again, we see the requirements are ranked differently by different perspectives and the one to use depends on the product manager's decision making context. However, we can see that from the ranking from EA perspective closely follows the OVERALL perspective, and the PS perspective follows the same trend but to a lesser degree.

Effort estimates of the requirements is a major source of uncertainty and risk in release planning. Over or under-estimation of the efforts could lead to unforeseen consequences and a plan that does not reflect the realities. Sensitivity ranking recommendations provided by the tool helps the product manager to take proactive steps towards mitigating the risks involved due to often inaccurate effort estimates.
Anticipating and devising strategies to counter the uncertainties would help the release plan to stay on track. For example, in this case we can see from the OVERALL, EA, and PS output metric perspectives, that the most sensitive requirements are REQ-12, 15, 18, 23, 30, 49, 50, etc. A product manager can re-consider the effort estimates of as much as of the 'sensitive' requirements as possible; to make sure they are accurate and close to reality. Better effort estimates would improve the quality of the plan and stakeholder satisfaction. Also the plans generated by a SRP model, such as EVOLVE II, are as good as the estimated values provided to it in the first place.

### 7.4.2 Rec. 2: Trends in Output Metrics Due to Input Parameter Changes

Although useful, the sensitivity ranking is a concise and condense type of information. The next question would be; what makes the parameters more or less sensitive? In other words, how each output metric gets affected due to input variations? The PM can use the sensitivity recommender module of the tool to drill down into the sensitivity data and pay closer look into this aspect. Specifically, the impact of a certain series of input changes on one or more output metrics is visually presented in the form of scatter plots. Figure 7-6 shows the screen of this module for this case study.

As an example, let us consider the parameter 'Res1, R2', capacity of resource-1 allocated for release 2. This parameter is among the 10 most sensitive parameters of the case study project. Since this parameter affects the release plans to a larger degree, it
Figure 7-5: Sensitivity of the effort estimate parameters from different output perspectives
would be useful to investigate what happens to the satisfaction level of the stakeholders if Res1, R2 values change. In a typical use case, the PM goes to the 'Sensitivity Recommender' tab of the tool, selects 'Res1, R2' as the input perspective and one or more EA metrics in the output perspective corresponding to each stakeholder. For this example and for the sake of simplicity, the EA of stakeholder #6, 9, 11, and 14 is selected as shown in Figure 7-6.

We can see that changing the capacity of Res1, R2 have varying impact on the excitement of each stakeholder. The EA of stakeholder #11 and #14 increases as the capacity of Res1, R2 is increased. It is usually expected that adding more resources would help the satisfaction of the stakeholders since more of their favored features gets allocated to a release rather than postponed, and this seems to be the case for our stakeholder #11 and 14. However, this is not always true since we can observe that increasing Res1, R2 capacity in fact decreases the EA of stakeholder #6, and have no effect on the EA of stakeholder #9. Having this information, the PM would now be in a better position in case this resource's capacity is to be changed, and the ultimate decision depends on the goal of the PM. For example, if stakeholder #6 is deemed more important than the other three, then the capacity should not be changed. If the goal was to increase EA of stakeholder #9, then changing the capacity of Res1, R2 have no effect and something else should be done.
Similar to the above example, the tool allows the PM to investigate the effect of various input changes on one or more output metrics depending on the goal and decision making context. The PM can also export this sensitivity data as CSV that could be analyzed using third party tools such as Excel.

As an example, let’s assume the PM intends to decrease the number of features being postponed. From the tool, the PM would select the 'PF' output metric and similar to the above example, the PM can set the input perspective to different input parameters to see if changing any of parameter values would help. Of-course it is possible that any parameter changes would affect positively the PF. One of the obvious ways would be to investigate the effect of increasing the resource capacities. But then the question would be which resource capacity should be increased and for which release.
Using the sensitivity data from the tool, Figure 7-7 shows the scatter plot of resource capacity changes versus the impact on the PF metric. The trend in the graph shows that increasing the Res1, R1 capacity and Res1, R2 capacity would increase the PF metric, i.e. decrease the number of features getting postponed. But increasing the capacity of other resource parameters actually slightly decreases the PF metric, at a constant level. Therefore in this case, the PM could either increase the Res1, R1 or Res1, R2 capacity.

7.4.3 Rec. 3: Predicted Impact of Release Project Changes

Here the use case of the 'Change Impact Recommender' module of the tool is presented, which is the implementation of the prediction method described in Chapter 4. As a one-
time process, the tool first discretizes the output metric values of the simulation data, and uses the transformed data to train a SVM classifier for each output metric. For example, the classifier for PF metric would allow estimating the impact of input changes on the number of features getting postponed.

In this case study, two class widths were investigated during discretization; 0.05 and 0.1 and two sets of classifiers trained for each output metric. Ten-fold cross-validation was used to determine the average accuracy of each classifier for each discretization class width. The accuracy of the SVM classifiers is shown in Figure 7-8. We can observe see that the average accuracy for either class width is about 73%. This means that the trained SVM classifiers correctly predict the impact level of input changes 73 percent of the time. Since a class width of 0.05 is more finer-grained and there is no loss in accuracy, we will use the set of the SVM classifiers trained on this data.

The graph in Figure 7-8 also shows that two classifiers; EA of stake. 17 and EA of stake.9 are almost 100% accurate. A closer look at the data shows that the output metric values for these are all zero which is the reason behind such a high accuracy. Ignoring these two classifiers, the most accurate classifier is the one for 'PS' metric with 92.7% accuracy. The worst case is the classifier that predicts EA of stakeholder number, which has an accuracy of only 45.4%.

Next the use case scenarios of this recommendation module of the tool are demonstrated through examples.
Figure 7-8: Prediction accuracy of SVM classifiers
Figure 7-9 shows the 'Systematic Changes' option of the 'Change Impact Recommender' module, where a PM can select the input parameter to be changed and the range within which the values to be changed to. After this selection, the 'Ok' button is clicked and the tool automatically generates a series of release scenarios corresponding to each change. The output metrics are estimated using the SVM classifiers trained instead of running the ReleasePlanner™. As a result this process is extremely fast (takes less than a second). The PM can then go through the scenarios, and upon selecting one of the scenarios the impact of the changes with respect to the baseline project is shown visually to the PM based on the selected output perspectives. The PM can select different output
perspectives as desired and the impact information is shown instantly.

This feature of the tool is useful for PM in situations when the interest is to investigate what happens to the output metrics when one change is made to an input parameter (similar to OAT sensitivity analysis). As an example, we can see in Figure 7-9 that the PM has selected seven of the most sensitive requirement parameters namely REQ-12, 15, 18, 23, 30, 49, and 50. This information was retrieved from 'Sensitivity

Figure 7-10: Screen of the change impact recommender tab - manual
Recommender' module and explained earlier. The PM is interested to investigate the impact on output metrics if effort estimates of these requirement parameters increases from 5% to 40% with 5% increments. Upon clicking 'Ok' button, the tool generates 55 scenarios and their output metrics estimated using the trained SVM classifiers. The scenarios have varying level of impact for each output metric Figure 7-9 shows that scenario #39 is selected where the effort estimate of REQ-30 is increased by 40%. The plot below in the screen displays the impact of this change on PF, SFP, and EA metrics of some of the stakeholders. We can see the overall this change has a negative effect on all these metrics. The number of features postponed is predicted to increase, and the stakeholders are estimated to be unhappy due to this. In this manner, the PM can browse other scenarios and observe the impact of the changes. Such information would help the PM to better anticipate the consequences of the uncertainties involved with effort estimates.

The 'Manual Changes' option of the 'Change Impact Recommender' module is shown in Figure 7-10. This feature assists the PM when the goal is to find what happens to the output metrics when more than one input parameters' values are changed. Upon changing one or more inputs, the impact of this change(s) is predicted using the trained SVM classifiers and are displayed to the PM visually based on the selected output perspectives. For example in Figure 7-10 we see that the PM has increased the capacity of "Res1,R1" by 10%, and has decreased the capacity of 'Res3-R1' and 'Res3-R2' by 15%. The impact information shows that this change is positive and improves the current baseline plan. The number of features postponed is expected to decrease by 70%, and the
stakeholder satisfaction is either neutral (no change) or positive to various degrees. This information helps the PM to re-allocate the resources to different releases if possible.

In this manner, the PM can change various input parameter values, and their estimated impact on the output metrics is instantly provided.

**7.4.4 Rec. 4: Achieving Release Plan Targets**

This question of the case study is investigated using the 'Target Achievement Recommender' of the tool which is the implementation of the recommendation method described in Chapter 5 to assist in achieving one or more release targets. For example, an increase in a PF metric is considered an improvement with respect to the baseline plan since the number of postponed features is decreased. Figure 7-11 shows the screen of this module. In typical usage of this module, PM follows the following steps.

1. PM selects the desired release targets, and assigns a weight on the scale of 0 to 5 to each target (top left part of the screen in Figure 7-11). Zero implies that it should be ignored, and the higher the weight the more it should be favored compared to other targets.

2. PM selects which scenarios to analyze; the scenarios generated during the simulation and/or new scenarios to be generated systematically by changing the input parameters (considers all of them) based on the specified ranges. The output metrics of these newly generated scenarios are estimated from the trained SVM classifiers.

3. Upon clicking the 'Apply' button, all the scenarios are ranked based on how much each scenario meets the PM targets. The formula used was explained in section 5.3.3.
The ranked scenarios are listed for the PM as shown in the bottom left part of the screen.

4. When the PM selects a scenario from the list, the tool provides three types of information in the bottom right part of the screen.
   a. Bar plot of targets that get achieved. This includes the targets that were desired by the PM or any other targets that gets improved.
   b. Bar plot of the trade-offs, and their magnitude.
   c. A list of parameter changes that achieve the targets.

As an example, we can see in Figure 7-11 that the PM would like to improve the all stakeholder's excitement score and each stakeholder is considered equally important since the weights for the targets are set to 5. The PM has also opted to rank both the simulation scenarios as well as generate new scenarios systematically. The highest ranked scenario (Id. 1091) is selected and its corresponding recommendations are displayed. The recommendation is that if the capacity of Res1-R1 is increased by about 8 units then the desired targets will be achieved and in addition the PF and SFP metric gets improved as well. Each stakeholder’s EA gets improved to varying degrees as shown in the Figure 7-11. However, it is usually very difficult to make all stakeholder's happy at the same time. We can see as part of the trade-off, two of the stakeholder's EA (#10 and #15) deteriorate.
The PM can browse through other scenarios and various different recommendations are provided. For example, the scenario with Id. 1090 in fact improves all the stakeholders’ excitement and meets all the targets. But it's ranked lower since the trade-off is high, i.e. increase the capacity of Res1-R1 by 20%. However, the ultimate decision is left for the PM. It is possible that PM is willing to accept such high increase in the capacity in return to increase all the stakeholders’ excitement score.
7.5 Threats to Validity

Since we demonstrated the applicability of the recommendation system for a single release planning project in a single company, the external validity cannot be ensured. The generalizability of the recommendations provided is a concern, and more case studies of projects in different companies are needed. However, the recommendations provided by the tool are project specific and subjective. It is project specific since recommendations are extracted from data generated by creating various release scenarios of a project during simulation. And the recommendations are subjective since a certain advise might be considered useful by one product manager, but not another.

An industry evaluation would also help, in the form of survey of product managers to investigate if the recommendations are accepted by the managers, and to which extent.

7.6 Summary

Applicability of the recommendation system described in this thesis was evaluated in this chapter through a case study of a real life release planning project. The user interface of the tool developed for the recommendation system was also demonstrated in this chapter. Through this case study, we demonstrated that the tool is capable to perform simulation and sensitivity analysis for a large project. From the data generated, four different types of recommendations were provided to assist the PM with better release decision making. The PM was provided with most and least sensitive input parameters. An example of how to use the tool to identify trends in release plan metrics was also provided in this chapter. The case study showed that the prediction classifiers trained are reasonably accurate and
we demonstrated its useful application for predicting the estimated impact of one or more input changes. The last type of recommendation was to assist the PM with achieving certain desired planning targets.
Chapter Eight: **Summary and Future Research**

A summary of this thesis is provided in this chapter by describing the contributions made towards achieving the research objectives of the thesis. The limitations and applicability are discussed and the chapter is concluded with a discussion of possible future research.

**8.1 Contributions and Summary**

**8.1.1 C1: Comprehensive Systematic Review and Mapping of RSREs**

As part of the literature review, we applied systematic mapping to provide a comprehensive overview of recommendation systems for requirements engineering process, their characteristics, and state of validation. The review methodology is based on the software engineering systematic mapping guidelines by Petersen et al [25]. This is the first systematic review of literature that considers RSREs. Since strategic release planning is one of the first phases of requirements engineering, we also investigated the recommendation methods or tools for release planning as well.

The mapping of the existing literature determines the coverage of the relatively new research field of RSREs, and identifies the research gaps. Our results are also helpful for requirement engineers and product managers who need to know how recommendation systems can assist them in their tasks. The results of the mapping also reaffirm the need for a recommendation system such as the one described in this thesis that helps product managers during the release planning activity.

**8.1.2 C2: Simulation and Sensitivity Analysis Method for SRP**

We designed a sensitivity analysis method that analyzes the influence of input changes on output of a SRP model by systematically changing input parameters and investigating the
consequences of the changes on the output. The systematic changes are performed using a simulator, that interfaces to a implementation of a SRP model. The concepts developed are generic and could apply to any SRP model. The method is only dependent on definition of a set of metrics for measurement of input changes and output changes. For demonstration purposes, we demonstrated how this method applies to a SRP model called EVOLVE II [3] which is implemented in a tool called ReleasePlanner™. A set of metrics for input changes and output changes was also developed.

The method ranks the input parameters of the SRP model based on their sensitivity, and assist the PM in identifying any trends in the outputs due to the input changes. The results of this method also have an important role towards achieving our main goal of developing the SRP recommender tool. Data generated during the simulation is used by the recommendation methods, explained later, for further analysis. The method is not only a completely automatic process, but is also highly customizable which allows us to integrate an expert's judgement into the method. During the simulation, a user can select any number of input parameters of the model, specify by how much to change each parameter, and in which manner (one at a time or all together). During the analysis, once again the user has the option to calculate the sensitivity of the input parameters from various perspectives such as stakeholder satisfaction. In the same manner, the user can view and compare the trends from various output perspectives.

8.1.3 C3: Recommendation Method for Predicting Impact of Changes on the Release Plans

A recommendation method was designed that recommends the impact of future input changes on the release plans. This is achieved by utilizing data generated during the
simulation and sensitivity analysis, to dynamically train a set of SVM classifiers, which are then used to recommend the effect of input changes. The aim is to assist product managers stay informed of the effect of input changes to avoid any unforeseen consequences such as stakeholder disappointment or delay of an important feature. The processes involved in this method are fully automated; data from simulation and sensitivity analysis is pre-processed to prepare it in the correct for the SVM trainers, the labels are also discretized for better prediction accuracy of the classifiers, the method also finds a good pair of values for the C and gamma parameters of SVM, and those are used to train the SVM classifiers.

8.1.4 C4: Recommendation Method for Supporting Release Plan Target Achievement

A utility-based recommendation method was designed that assists in achieving certain release targets specified by a product manager. Examples of targets are improving satisfaction of a stakeholder, and reducing the number of features getting postponed. More than target can be specified at a time, where each target is given an importance weight. The method then analyzes the release planning scenarios generated during the sensitivity and simulation analysis. The scenarios are ranked based on how much it helps in achieving the specified targets or in other words how much utility it has for the user. The utility score also depends on the trade-offs and magnitude of changes that needs to be made in order to achieve the target. The trade-offs could include adding more resource capacity or dissatisfaction of some stakeholders, or postponed for some features.

If desired by the product manager, the method can generate scenarios systematically using the simulation methods devised earlier. The output metrics of the
scenarios are predicted using the prediction model devised in C3. The prediction model's accuracy is factored in determine the utility of scenarios generated in this manner.

8.1.5 C5: Implementation and Evaluation of Recommendation Tool for Product Managers

A comprehensive recommendation tool aimed for product manager was designed and implemented. The tool integrates the simulation, sensitivity analysis, and recommendation methods described in this thesis, into one automated process. The tool is implemented as a plug-in for Visual Studio IDE and also interfaces with Microsoft's Team Foundation Server. Through an interface, the tool can also programmatically connect to the web-based ReleasePlanner™, an implementation of the EVOLVE II SRP model. This allows us to automatically and systematically generate release plans and retrieve its outputs.

The recommendation tool was evaluated through a case study of a real life strategic release planning project. Through the case study we demonstrated the applicability of the methods described in this thesis to assist the product manager with four different types of recommendations. The tool is intended as a means for product managers to make data centric release decision making, in addition to applying their expert judgment. Intuition-based decision making alone does not always work and might even have negative consequences [94].

8.2 Limitations and Applicability

The recommendation methods and the tool presented in this thesis helps the product manager with better release decision making, but only in a few scenarios. The recommender was not intended to solve every problem of the product manager in release
planning, and this is not claimed here. Recommendation systems in release planning field is considered very useful [22] and yet not researched or applied in industry extensively as our literature survey in Chapter 2 indicates. The recommender for SRP presented here is meant to be a good starting point in this direction.

The recommender methods described here are more applicable for organizations which are at the Capability Maturity Model Integration (CMMI) [95] level 4 (processes are measured and controlled) and level 5 (process improvement is the focus). Such organizations have an established planning process where it is understood and accepted that earlier planning is necessary for success of a product. The planning process is overseen by at least one product manager in the firm, who also works closely with the stakeholder to consider their opinions during the release planning activity. Quality of the release plans and consequently the applicability of the recommendations generated from the methods described here depend on the project data used in the first place. Therefore, it is important that relevant data keeping measures are in place, such as effort estimates. And finally, this recommender system and the underlying SRP model are more suitable for larger projects (20+ features).

8.3 Future Research

The recommendation methods described can be further improved and the modular design of the implemented tool gives us the option to extend it in the future. Some of the proposed future works are:

1. Evaluation of the tool in an industrial setting would give us a better understanding of the performance of the tool towards assisting the product managers.
2. Design of metrics and a recommendation method for analyzing the impact of adding new features to a project, and the impact of delaying certain features.

3. Considering quality of a product being planned for during the sensitivity analysis and target achievement recommendations. New metrics needs to be developed to quantify the 'quality' of the product.

4. Design of a method that performs data mining and analysis across several release planning projects. The methods described here focuses on one project. Additional insight and recommendations can be derived by considering a series of release planning projects.

5. The tool implemented can be further improved in several ways. The code responsible for the simulation and analysis can be made more efficient, and more light-weight. Although multi-threading is currently used for the main methods of the tool, more methods could be changed to work in parallel. And finally, the graphical user interface can be made more user-friendly and intuitive.

6. The tool is currently implemented as a plug-in for Visual Studio IDE. It can be integrated into other platforms as well such as IBM’s Jazz platform and IDEs such as Eclipse.

7. Besides strategic release planning, operational planning is also frequently done in industry. Simulation, sensitivity analysis, and recommendation concepts could be applied in the context of operational release planning as well.
References


