University College London

PhD Thesis

Modelling and Analysing Software Requirements and Architecture Decisions under Uncertainty

Author:
Saheed Abiola Busari

First Supervisor:
Dr. Emmanuel Letier

Second Supervisor:
Dr. Earl Barr

A dissertation submitted in fulfilment of the degree of Doctor of Philosophy

in the

Software Systems Engineering
Department of Computer Science

February 2019

Declaration of Authorship

I, Saheed Abiola Busari, declare that this thesis titled, "Modelling and Analysing Software Requirements and Architecture Decisions under Uncertainty" and the works presented in it are my own. I confirm that:

- This work was done mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: ______________________________

Date: ______________________________

Abstract

Early requirements engineering and software architectural decisions are critical to the success of software development projects. However, such decisions are confronted with complexities resulting from uncertainty about the possible impacts of decision choices on objectives; conflicting stakeholder objectives; and a huge space of alternative designs.

Quantitative decision modelling is a promising approach to tackling the increasing complexity of requirements and architectural decisions. It allows one to use quantitative techniques, such as stochastic simulation and multi-objective optimisation, to model and analyse the impact of alternative decisions on stakeholders' objectives. Existing requirements and architecture methods that use quantitative decision models are limited by the difficulty of elaborating quantitative decision models and/or lack of integrated tool support for automated decision analysis under uncertainty.

This thesis addresses these problems by presenting a novel modelling language and automated decision analysis technique, implemented in a tool called RADAR, intended to facilitate requirements and architecture decisions under uncertainty. RADAR's modelling language has relations to quantitative AND/OR goal models used in requirements engineering and feature models used in software product lines. The language enables modelling requirements and architectural decision problems characterised by (i) single option selection similar to mutually exclusive option selection (XOR-nodes) of feature diagrams; (ii) multiple options selection similar to non-mutually exclusive options selections (OR-nodes) of feature diagrams; and (iii) constraints dependency relationships, e.g., excludes, requires and coupling, between options of decisions. RADAR's analysis technique uses multi-objective simulation optimisation technique in evaluating and shortlisting alternatives that produces the best trade-off between stakeholders' objectives. Additionally, the analysis technique employs information value analysis to estimate the financial value of reducing uncertainty before making a decision.

We evaluate RADAR's applicability, usefulness and scalability on a set of real-world systems from different application domains and characterised by design space size between 6 and 2^{50}. Our evaluation results show that RADAR's modelling language and analysis technique is applicable on a range of real-world requirements and architecture decision problems, and that in few seconds, RADAR can analyse decision problems characterised by large design space using highly performant optimisation method through the use of evolutionary search-based optimisation algorithms.

Impact Statement

Designing software systems involves deciding what software functions should be provided, what levels of quality requirements (e.g., performance, security and availability) should be met, what software architecture to use in satisfying these functional and quality requirements, what components and connectors to use and what deployment strategies to use. Making the right decisions is critical to satisfy stakeholders' goals and ensuring successful development and evolution of software systems. Such decisions, however, are complicated by conflicting stakeholders' objectives and significant uncertainty.

This thesis presents the Requirements and Architecture Decision AnalyseR (RADAR) –a lightweight yet expressive modelling language and automated decision analysis open source tool. RADAR is intended to assist requirements engineers and software architects in making informed and confident software requirements and architecture decisions under uncertainty. Using RADAR also benefits the end-users (customers) of the software system as the system-to-be will be performant, reliable, secure, delivered on time and ultimately satisfy their needs.

RADAR can be used as a stand-alone tool by requirements engineers and software architects. The tool can be deployed as an add-on to existing goal-oriented modelling frameworks, such as KAOS/Objectiver. RADAR can also be used as a plug-in to existing software release planning tools, such as EVOLVE, to help elaborate domain-specific release planning decision models and analyse uncertainty in the decision models. Finally, RADAR can be used to improve other software engineering decisions under uncertainty; this includes decisions about test case selection and prioritisation under uncertainty and product (set of software features) configuration decisions in Software Product Line Engineering in the presence of conflicting stakeholder goals (e.g. maximising business values and minimising costs) and uncertainty.

Preliminary work have been done to propagate the benefit of using RADAR to support requirements and architecture decisions under uncertainty. In this regard, parts of RADAR's implementation and its evaluation on five real-world systems (e.g., fraud detection system, security policy system, emergency response system) have been presented in two prestigious software engineering venues (ICSE'17 and ASE'17). In the nearest future, we intend to apply RADAR on a real-world case study and evaluate the benefit and effort required to use the RADAR decision analysis tool in an organisation. We also intend to investigate and implement techniques for validating and calibrating RADAR models against run time data of a system. To achieve these future goals, we may have to collaborate with other researchers and industry professionals in the field of software engineering, and in particular requirements engineering and software architecture.

Acknowledgements

First, I give thanks to Almighty God for making this PhD a success.

Second, I give special thanks to my supervisor, Emmanuel Letier, for his patience, support and continuous feedback. This research would not have been possible without his remarkable experience and tutelage. I thank Earl Barr and David Clark, my examiners in my first and second year vivas. Your experience, knowledge and critical evaluations have been valuable to this work. Many thanks to Yijun Yu and Nelly Bencomo, my examiners in my final viva, for meticulously reading my thesis and providing useful feedback.

My sincere appreciation goes to my sponsor: The Petroleum Technology Development Fund (PTDF) in Nigeria. In addition, I thank the Department of Computer Science for the travel grants awarded to enable me present my research at the 39th International Conference on Software Engineering (ICSE) in Buenos Aires, Argentina and the 32nd International Conference on Automated Software Engineering (ASE) in Illinois, USA.

I would also like to recognise some of the world class researchers at the Software Systems Engineering (SSE) Group and Centre for Research in Evolution, Search and Testing (CREST), University College London. Their comments, feedback and numerous discussions, especially during the reading group, were extremely valuable.

Finally, I thank my wife, Olaide Ruth Awoyale, for enduring and supporting me throughout my studies, and my lovely children, Aishat Akorede Michelle Busari and Ahmad Nathan Busari, for their cooperation as well. Special thanks to my parents for their love and prayers, especially my Dad, Mr Raufu Abiola Busari, for painstakingly proof reading my thesis. To my siblings, Sofiat Busari, Idris Busari and Toyibat Busari, my parent in-laws, brother-in-laws, sister-in-laws, my cousin (Bamidele Busari), nephews, friends and fellow PRESSID scholars. I thank you all for your support and prayers.

Publications

The work presented in this thesis is the original work undertaken between September 2014 and September 2017 at the Department of Computer Science, University College London, United Kingdom. Part of the work presented in this thesis has previously been published in the following venues:

- **Saheed A. Busari**, Emmanuel Letier. RADAR: A Lightweight Tool for Requirements and Architecture Decisions. *In Proceedings of 39th International Conference on Software Engineering,* ICSE'2017, pp. 552-562. IEEE Press, 2017.
- **Saheed A. Busari**. Towards Search-Based Modelling and Analysis of Requirements and Architecture Decisions. *In 32nd IEEE/ACM International Conference on Automated Software Engineering,* ASE'2017, Urbana-Champaign, IL, USA.

The above two papers describe an initial version of the RADAR modelling language and analysis tool. The thesis presents an extended version of RADAR that includes a richer modelling language supporting constraints and decisions with multiple option selections, and a more performant optimisation method relying on evolutionary search-based algorithms instead of exhaustive search.

While conducting this PhD, I also collaborated with other researchers to publish the following work. However, this work does not feature in this thesis.

- Ragkhitwetsagul, Chaiyong, Matheus Paixao, Manal Adham, **Saheed A. Busari**, Jens Krinke, and John H. Drake. "Searching for Configurations in Clone Evaluation: A Replication Study." *In International Symposium on Search Based Software Engineering,* pp. 250-256. Springer International Publishing, 2016.

Other publications include:

- **Saheed A. Busari**, Emmanuel Letier. RADAR: Scalability Analysis of the RADAR Decision Support Tool. arXiv preprint arXiv:1702.02977. 2017 Feb 9. Online: https://ucl-badass.github.io/radar/.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Requirements Engineering and Software Architecture are key software development activities that play significant roles in building and evolving software intensive systems. While requirements engineering aims at identifying and documenting the problems to be solved, software architecture defines the solution(s) to these problems. Making the right requirements and architecture decisions is critical to the success of any software project [264], but such decisions are confronted with significant uncertainty. The thesis presents a novel modelling language and analysis tool to support such decisions under uncertainty.

1.1 Requirements Engineering and Software Architecture

1.1.1 Requirements Engineering

Requirements engineering is a fundamental phase in any software development project. It addresses a wide range of problems from project initiation, design and evolution of software systems [264]. *Requirements* are statements that describe what a proposed system must do, and conditions in which the system must operate. In general, requirements are representations of the decisions concerning the behaviour of a software system [28], and they can be classified as either functional or non-functional [180]. Functional requirements describe the intended functions the system is to perform i.e. how the system responds to certain inputs. Non-functional requirements (NFR) [104], also known as soft goals [180, 242] or quality requirements [140], are constraints on the functions provided by the system. Such constraints could be on the system's performance, availability, security, scalability and reliability etc.

Requirements engineering involves identifying and extracting the purpose of a proposed system; system stakeholders and their needs; conditions in which the system must operate; the interaction between the system and its environment; and documenting these information in a specification that can be used for future analysis, modelling and design [189].

Requirements engineering involves a series of activities that clarify the software system goals, deal with conflicts, uncertainty and risks in requirements [264]. These activities form the basis and prerequisite for a successful delivery of a software project [264], and are mostly iterative and performed in parallel [129]. These activities generally include: requirements elicitation [12], requirements modelling and analysis [11], requirements specification [260], validation & verification and requirements management.

1.1.2 Software Architecture

Software architecture has gained recognition as a key aspect of software engineering. This is due to the increasing complexity and the need for reliable software systems [29]. The architecture of a software intensive system represents the first design artefact, and it is mainly developed at the early stages of a software development process. Devoting substantial effort in designing and documenting the architecture of software systems not only helps to ease the attainment of certain functional requirements, quality requirements (e.g. security, availability, performance etc.) and organisation‘s business goals [86], but also provides a software developer with control of how to evolve and maintain the system [146].

The software architecture of a system is the set of significant decisions that describe the structure(s) of a system in terms of the software elements (e.g. modules, object-oriented classes or packages, database stored procedures, services, data files); the externally visible properties of those elements in terms of quality attributes; the relationships among the elements; and the architectural styles that guide the system structure(s), deployment and evolution [214]. In this definition, the system's structure can be viewed as static and dynamic: the static structure defines the elements and their arrangement whereas the dynamic structure shows how the system responds to external or internal events at run time.

In general, software systems possess an architecture, whether implicitly or explicitly designed [86]. A good software architecture is one that reduces risk and is flexible to handle changes in requirements, hardware and software technology. Deciding an optimal software architecture is an iterative process and involves some activities that guide the derivation of a software architecture from the functional and non-functional requirements. These activities can be categorised into three phases [86]: the first phase is domain understanding of the problem a proposed system intends to solve, and the stakeholder objectives in order to elicit the requirements that are architecturally significant. The second phase involves evaluating alternative software architectures to determine the extent at which it solves the stakeholders' goals. The last phase involves deciding the optimal architectural solution(s) that fulfils the stakeholders' goals. These activities end when decision-makers are confident that the selected candidate solution can solve the problem to an acceptable level, otherwise, the requirements are re-analysed and other alternative solutions are considered.

1.2 Requirements Engineering and Software Architecture Decisions

Requirements engineering decisions involve deciding what software functions should be provided and what levels of quality requirements the system should deliver at run time [42]. In goal-oriented requirements engineering, requirements decisions involves deciding among alternative system designs; making decisions about alternative ways to resolve conflicts and decisions about assigning responsibilities to humans, software or hardware systems [264]. Software architecture decisions involves deciding the structure(s) of a software system; what software and hardware components and connectors to use; and the deployment strategies [214].

Requirements and architecture decisions often have critical impacts on the software development costs, duration, and the system's ability to satisfy stakeholders' goals and deliver business value [214, 264]. Such decisions are complicated by large number of possible alternatives; multiple conflicting stakeholders' objectives; and high levels of uncertainty which may include: uncertainty about the impact of the dynamic nature of the organisation or system run time environment; uncertainty about the accurate project estimations; uncertainty about the availability of human and technical resources; uncertainty about the presence of obstacles that may hinder the achievement of goals; and uncertainty about the possible impacts of decision choices on the stakeholders' objectives [166].

In engineering, uncertainty is commonly defined as the lack of complete knowledge about some entity, such as an unknown quantity, a future event, or even a past event about which we are unsure [87]. There is more than one possible value for the quantity or more than one possible outcome for the event, and the "true" value or outcome is uncertain.

Uncertainty can be measured by assigning probabilities to the different possible values and outcomes.

In this thesis, we will be concerned with analysing the software architects' uncertainty about the impacts of design decisions on stakeholders' objectives. We will not address other design-time uncertainties such as the software architects' uncertainty about having identified all relevant stakeholders' objectives and having identified a suitable set of design alternatives.

We will also not address run-time uncertainties dealing with uncertain information that software systems have about their environment, design goals, and assumptions [231]. Research on adaptive systems aims to manage such run-time uncertainty and have proposed various classification of run-time uncertainty and their sources [172, 200, 207]. In contrast, in this thesis, we are dealing with design-time uncertainty due to incomplete knowledge about the impacts of design decisions on goals. In the final chapter, we will revisit how the design-time techniques presented in the thesis might be adapted to support run-time decisions in self-adaptive systems.

Motivating example. Suppose an architect wants to improve the efficiency of a bank's plastic card financial fraud detection system (shown in Fig. 1.1) so that the system minimises financial loss due to undetected frauds; minimises the cost of investigating frauds; and minimises cardholders' inconvenience due to false alerts that result in their cards being blocked unnecessarily. In this situation, the architect is faced with design decisions such as [42]:

- the *processing type* that can be continuous or batch; continuous processing analyses transactions individually as they arrive, whereas batch processing performs an overnight analysis of the transactions that occurred during the day.
- the *fraud detection method* which can be a two-class supervised classification method in which a classifier is trained from samples of past fraudulent and non-fraudulent transactions, or a non-statistical rule-based method that flags transactions matching configurations known to be high risk.
- if the classifier fraud detection method is chosen, the *alert threshold* defines some threshold above which the classifier should flag a transaction as suspect. A low alert threshold means more alerts will be generated and thus a higher ratio of false alerts.
- the *blocking policy* that can include blocking an account as soon as the fraud detection method flags a transaction as suspected fraud, or only blocking the account after human investigators (e.g., bank staff) have confirmed the suspected fraud.

These decisions impact the number of fraud alerts generated and the speed at which compromised accounts are blocked, which ultimately affects how much effort the bank needs to devote to investigate alerts manually and how much money it loses to fraud. Therefore, the optimisation of plastic card fraud detection systems typically includes two conflicting objectives [115]: minimising financial loss, and minimising manual investigation costs. Deciding what combination of processing type, fraud detection method and blocking policy to use is not trivial. The decisions are complicated by uncertainty about domain quantities, such as the ratio of compromised accounts, and the uncertainty about the impact of decisions on future financial loss and investigation costs.

Relying on intuitions alone to make such critical and complex decisions is not ideal. Intuitive decisions, even by experts, are subject to many cognitive biases and errors [141], which may lead to sub-optimal solutions. Some of these biases have been shown to happen in software engineering contexts [26].

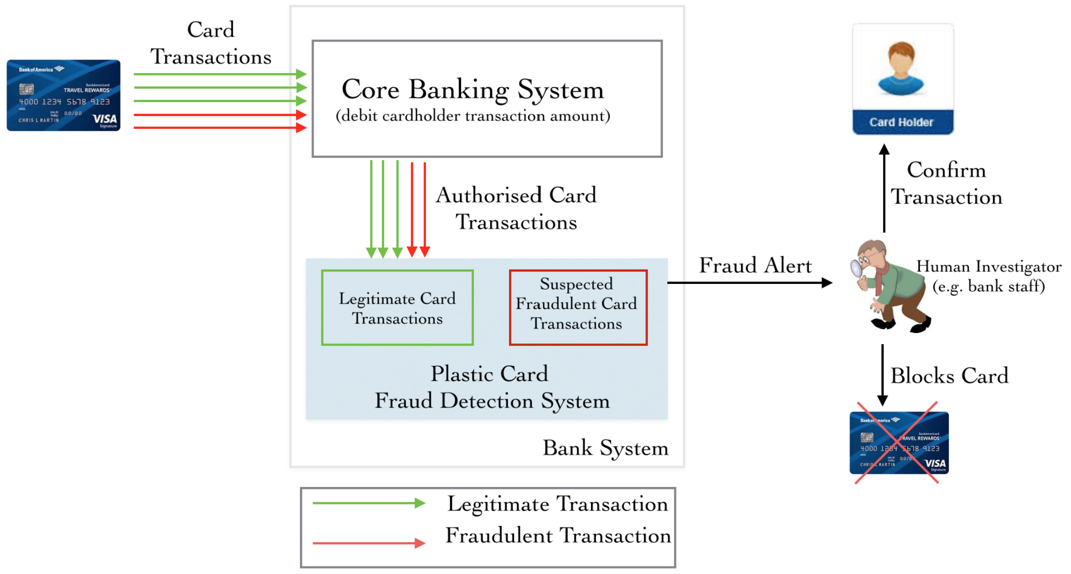

Card Transactions
Core Banking System
(debit cardholder transaction amount)
Authorised Card Transactions
Legitimate Card Transactions
Suspected Fraudulent Card Transactions
Plastic Card Fraud Detection System
Bank System
Fraud Alert
Card Holder
Confirm Transaction
Human Investigator (e.g. bank staff)
Blocks Card
Legitimate Transaction
Fraudulent Transaction

FIGURE 1.1: The plastic card fraud detection system: detects when plastic card accounts may have been compromised by fraudsters who are using the account to steal funds [115]. The system generates fraud alerts when authorised card transactions are suspected to be fraudulent. The bank staff investigates fraud alerts and blocks the card if it is confirmed that such transactions are not performed by the cardholder.

In this thesis, we aim to support requirements engineers and software architects in making this kind of decisions. The thesis introduces a decision support tool that enables modelling and analysing such requirements and architecture decisions under uncertainty.

1.3 State of the Art

The scientific literature has a large body of research works devoted to assist decision-makers in modelling and analysing requirements engineering and software architecture decision problems. Typical problems studied in the literature include: decisions about which set of software requirements (features) to implement in the next release of a product [30]; selection and evaluation of software architectures that meet certain functional and non-functional requirements; the selection of software and hardware components, their replication, the mapping of software components to available hardware nodes, and the overall system topology.

Many requirements and architectural decision support approaches exist to aid decision-makers in selecting decision choices that produce the best trade-off between conflicting objectives. These approaches can be grouped into three categories [41]:

1) *Qualitative decision models* such as the NFR framework [180] introduced to model, analyse, refine and expose the impact of alternatives on non-functional requirements [180]; the *i** [277] framework which adapts the NFR to model and reason about software systems and the organisational environments where they are used [276]; and the GRL framework that combines *i** and NFR frameworks to model goals, non-functional requirements, alternatives and decision rationale. These approaches are used to model, reason and analyse the impact of decisions on stakeholders' goals. However, they only provide information about how the decisions positively or negatively impact stakeholders' goals. They provide no quantitative information about the impact of alternatives on goals.

2) *Generic quantitative decision models* such as the CBAM (Cost-Benefit Analysis Method) [149] –an economic modelling approach to support software architecture decision making in a multi-stakeholder context; the EVOLVE evolutionary and iterative release planning method to optimally allocate requirements incrementally to software releases subject to resource and budget constraints [216]; and other search based software engineering methods [201]. These approaches employ pre-established model equations (e.g. typically weighted sums) to specify "cost" and "value" scores to alternative solutions. The cost and value scores in these methods usually represent abstract (non-observable) quantities rather than financial metrics expressed in monetary units (e.g. in Dollars or Euros). While the generic decision objectives and models ease the applicability of these methods, they may not express the actual stakeholders' goals and may not correctly predict the impacts of alternatives on these goals.

3) *Problem-specific quantitative decision models* such as the quantitative extensions to NFR/i* [3, 199], KAOS [126, 165] and the Bayesian decision analysis approach in software engineering [166]. These approaches are more accurate in modelling, reasoning and analysing domain-specific decision problems. However, they require higher modelling effort and generally have limited automated tool support for decision analysis. Using these approaches involves manual encoding of decision models in widely used statistical programming languages, such as R and MATLAB. This results in modellers spending time on implementation details rather than concentrating on the conceptual decision problem.

To mitigate the limitations of existing approaches, this thesis aims to introduce a quantitative approach that supports the elaboration and analysis of domain-specific requirements engineering and software architecture decision problems under uncertainty.

1.4 Thesis Objectives and Contributions

The objective of this thesis is to develop a new modelling language and automated decision analysis technique for requirements engineering and software architectural decisions. The result is a tool called RADAR —Requirements and Architecture Decision AnalyseR. RADAR's modelling language is a simplified form of quantitative goal models designed to be similar to simple equations that software architects use for back-of-the-envelope calculations (i.e. rough, imprecise calculations) [214]. RADAR, however, provides sophisticated analysis that cannot be performed through back-of-the-envelope calculations: it allows analysing uncertainty through Monte-Carlo simulations, shortlisting Pareto-optimal solutions through multi-objective optimisation, and computing expected value of information that can be used to decide whether to seek more information or perform a more detailed analysis before making a decision [166].

The proposed modelling language and automated decision analysis tool provide decision-makers with useful information about: which decisions are better than others in the decision model of a particular software intensive system; what objective values can be attained with different designs; what trade-off can be made between shortlisted designs (solutions); what parameter uncertainty may deserve additional data collection and analysis before making their decision; and what parameter uncertainty does not matter to their decision.

In this thesis, we define a novel modelling language and automated decision analysis tool that facilitates the application of the earlier MODA requirements and architecture decision method [166]. The thesis contribution with respect to MODA and other requirements and architecture decision methods are:

1. **A novel modelling language that facilitates the elaboration of domain-specific requirements and architecture decisions under uncertainty**. The modelling language has relation to AND/OR quantitative goal models used in

requirements engineering [126, 165] and similar to feature models used software product lines [236]. The language enables decision-makers model problem-specific decision problems at different level of complexities:

(a) Decisions characterised by single option selection similar to mutually exclusive option selection (XOR-nodes) of feature diagrams used in software product lines.

(b) Decisions with multiple options selection similar to non-mutually exclusive options selections (OR-nodes) of feature diagrams.

(c) Decisions characterised by constraints relationships between options of decisions. The constraint relationship considered are *excludes* and *requires* constraints widely used in feature diagrams and the *coupling* constraints commonly studied in requirements selection and optimisation [216, 285].

2. **An automated decision analysis approach for analysing domain-specific requirements and architecture decision models under uncertainty**.

   The analysis technique generates AND/OR goal graphs and decision graphs from the model's equations; generates design space and infers decision dependencies from model's equations; analyses model uncertainty through Monte-Carlo simulations, shortlists Pareto-optimal solutions through multi-objective optimisation, and employs information value analysis technique to compute expected value of information that can be used to decide whether to seek more information or perform a more detailed analysis before making a decision.

3. **Implementation of the modelling language and automated decision analysis technique in a tool, called RADAR —Requirements and Architecture Decision AnalyseR**. The tool is open source with a GUI and command line capabilities to facilitate wider adoption in practice and can be used as an add-on to existing goal-oriented modelling frameworks, such as KAOS/Objectiver [220].

4. **An evaluation and validation of the applicability, usefulness, scalability and performance of the modelling language and the automated decision analysis technique on a set of real world systems from different application domains**.

   The thesis used the following real-world software systems for evaluation: Plastic Card Fraud Detection System introduced earlier; Emergency Response System, NASA Satellite Processing System, Building Security Policy System, Public Bike Sharing System; the London Ambulance System; Microsoft Word Processor; Commercial Decision Support tool; Berkeley Relational Database Management System; Amazon Web Service Elastic Computing Cloud System; Drupal PHP framework for web content management; E-Commerce Web Portal.

Using these real world systems will help to evaluate the expressiveness and generality of RADAR's modelling language and analysis techniques. We evaluate the RADAR tool by answering the following research questions:

- **RQ1 (Applicability):** Is the RADAR tool applicable to real-world requirements and architectural decision problems (chapter 7.3.1)?
- **RQ2 (Usefulness):** Does RADAR's decision analysis technique provide useful improvements to real-world requirements and architecture decisions (chapter 7.3.2)?
- **RQ3 (Scalability):** What is the scalability of RADAR's exhaustive simulation and optimisation approach (chapter 8)?
- **RQ4 (Performance Analysis):** What is the performance of RADAR's alternative search-based evolutionary algorithms (chapter 8)?

1.5 Scope and Assumptions

The thesis focuses on developing a new modelling language expressive enough to model real world requirements and architecture decisions; implementing the decision analysis technique and evaluating the modelling language and analysis technique on real world examples. However, the thesis did carry out any user study to evaluate the effort required to develop a decision model and the benefit of our approach in industrial case studies.

As in most research works, we have made some assumptions as follows:

1. The thesis assumes modellers have some analytical skills and can identify design decisions and elicit stakeholders' objectives/goals.
2. The thesis assumes that the requirements specification about stakeholders' and system goals are not vague.

3. We also assume modellers can use existing methods for systematic model elaboration [162, 264] when developing decision models.

4. Finally, we assume the modelling language is applicable to a variety of application domains. However, like many other requirements and architecture decision methods, the thesis has not considered techniques for validating decision models against run time data.

1.6 Research Methodology

The research presented in this thesis is both theoretical and empirical. At the early phase, the first task was to identify the research problem and justify its significance to requirements engineering and software architecture research community. I performed a comprehensive literature review summarising the state-of-the-art approaches for modelling and analysing requirements and architecture decision problems, their strength and weaknesses are identified as presented in Chapter 2 of this thesis. The findings in this phase serve as a basis for proposing a new modelling language and automated decision analysis tool.

The next phase involves a number of research activities to design and implement the proposed modelling language and automated decision analysis tool [155]:

- Identify the requirements of the new modelling language and automated decision analysis tool as presented in Section 1.4.
- Identify and document the prototype tool's components and their relationships (see chapters 4, 5 and 6).
- Design the tool's data structure; design the tool's language and automated decision analysis techniques and integrate the different components that make up the tool; implement a GUI for the tool (see chapter 6).

The final phase involves the evaluation of the prototype tool against the formulated research problem. We answer the research questions presented section 1.4 (see chapters 7 and 8).

[0] https://ucl-badass.github.io/radar/

1.7 Thesis Outline

In this chapter, we gave an overview of requirements engineering and software architecture decisions; described state of the art requirements and architecture decision methods and their limitations; described the thesis objectives and contributions; clarified the thesis scope and assumptions. The remaining parts of the thesis are as follows:

Chapter 2 presents a background and the state-of-the-art in requirements engineering and software architecture in detail. Since many requirements and architecture decision problems are multi-objective in nature, we first present an overview of multi-objective optimisation. Next, we summarise and review the state-of-the-art models and approaches employed to tackle requirements and architecture decisions.

Chapter 3 presents a guided tour of the proposed Requirements and Architecture Decision AnalyseR (RADAR) to give a high-level overview of the language capability and to provide background on the decision analysis method used.

Chapter 4 formally describes the modelling language. We describe the language construct for modelling decisions with single option selection; decisions with multiple options selection; and decisions with constraints relationships between options of decisions.

Chapter 5 details the automated decision analysis technique. We describe the technique for analysing model uncertainty using Monte-Carlo Simulation; shortlisting Pareto-optimal solutions through exact and evolutionary algorithms; estimating expected value of information to help in identifying parts of the decision model that need detailed analysis or additional data collection before making a decision.

Chapter 6 describes the tool design and implementation.

Chapter 7 shows the modelling language and analysis technique in action by applying the tool on a set of real world examples. Here, the focus on evaluating the applicability and usefulness of RADAR. For each example real-world requirements and architecture decision problem, we describe the problem statement of the decision problem, present the decision model, presents the results of the model analysis, and where possible compare the analysis results to previous analysis in the literature. We conclude this section with conclusions from the experiment by discussing the applicability, usefulness and threats to validity.

Chapter 8 presents a critical evaluation of RADAR's optimisation analysis technique. The evaluation focuses on the tool's performance in analysing real-world requirements and architecture decision problems. We also discuss the threats to validity.

Chapter 9 presents the conclusion by summarising the thesis contributions and highlights potential future works.

Chapter 2

Background and State of the Art

As mentioned in Chapter 1, this thesis proposes a new modelling language and automated decision analysis techniques for requirements engineering and software architecture decisions under uncertainty. In order to provide a good perspective and foundation for the thesis, this chapter presents a background and state-of-the-art in software requirements and architecture decisions. Requirements and architecture decisions are generally multi-objective optimisation problems. Thus, Section 2.1 provides an overview of multi-objective optimisation problems and the algorithms commonly used to solve these problems. The thesis modelling language is a simplified version of quantitative AND/OR goal models. To this end, we give a brief background on goal-oriented requirements engineering in Section 2.2. Requirements prioritisation and release planning are a vital activities in requirements engineering decisions, therefore we presents its overview in Section 2.3. Software architecture decision methods for evaluating and selecting the optimal architecture of software systems are described in Section 2.4. Finally, Section 2.5 presents some works related to this thesis on product configuration decisions in Software Product Lines.

2.1 Multi-Objective Optimisation

Optimisation is at the heart of software engineering; it entails making decisions aimed at optimising software products and their development processes [28, 107, 205, 226]. Many requirements and architecture decision problems are characterised by multiple and usually conflicting objectives. For example, minimising cost, maximising stakeholders' values and minimising project risks. Additionally, the large number of possible alternative decision choices makes it difficult to explore the design space manually. Decision problems of this nature are known as multi-objective optimisation problems.

Multi-objective optimisation problems have been well studied in a field known as Search-Based Software Engineering (SBSE) [121] — a term, coined by Harman and Jones [119], that seeks to transform software engineering problem to a search based optimisation problems, and then search through a search space of possible solutions guided by objective function to obtain optimal or near optimal solutions.

In the rest of this section, we present a formal definition of multi-objective optimisation problems and Pareto optimality. Then, provide a brief description of the algorithms used to solve multi-objective optimisation problems.

Definition 2.1. A multi-objective optimisation problem (MOOP) [59] is a set of objective functions $\{O_1, ..., O_n\}$, to be either maximized or minimized, where each objective function $O_i : S \rightarrow \mathbb{R}$ assigns a real value to each solution s in the problem solution space S. The solution s is also known as *decision vector* —a vector of decision variables $\{s_1, s_2, \ldots, s_n\}$, where $s_1, s_2, \ldots, s_n$ could be Boolean variables commonly used in encoding requirements selection problems in which a value of 1 (or 0) for s_1 means a requirements r_1 is selected (or not selected).

Definition 2.2. A solution s *dominates* another solution s' if (i) it is better than s' in at least one objective and (ii) not worse than s' in all other objectives. Assuming all objective functions have to be maximised, these conditions are formalised as (i) there is an objective function O_i such that $O_i(s) > O_i(s')$, and (ii) for all other objective functions O_j with $j \neq i$, $O_j(s) \geq O_j(s')$.

Definition 2.3. A solution $s \in S$ is *Pareto-optimal* if it is not dominated by any other solution when all objectives are considered. The set of all Pareto-optimal solutions is called the *Pareto set*. The *Pareto front* is the set of objective values allowed by the Pareto set.

MOOPs can be solved through exact algorithms [72, 112, 139] that evaluate possible solutions in the solution space and find exact optimal solutions. MOOP can also be solved through approximate algorithms or meta-heuristics techniques to search a problem solution space guided by objective (fitness) function to obtain optimal or near optimal solutions [119].

Exact algorithms guarantee finding optimal solutions, if they exist [112], but spend longer time for large-scale problems. Exact algorithms, such as dynamic programming [33, 53], branch and bound [154, 160, 183], explore the search space in a non-exhaustive way, i.e. a huge size of the sub-space are dropped on the basis that the sub-space cannot contain the optimal solutions [187]. Enumerative methods, such as exhaustive search [187, 252] and Guided Improvement Algorithm (GIA) [139], however, scan all possible alternative solutions in the search space. In general, when the size of the problem is very large, exact algorithms scale poorly [79, 112, 139, 193]. Parallel exact approaches have been developed to improve the scalability of exact algorithms. Examples of parallel exact algorithms includes: Two Parallel Partitioning Method [161] and K-Parallel Partition Method (K-PPM) [72] used for the job scheduling problem, Parallel Guided Improvement Algorithm (ParGIA) [112] used for the feature selection in software product lines. While these parallel algorithms have been attempted on relatively large problems, they still do not scale for many real world problems with extremely large solution space [112]. There has been considerable amount of efforts devoted to developing approximate algorithms that scale well in tackling real world problems with extremely large solution space.

Approximate algorithms address the scalability limitations of exact algorithms, but they do not guarantee finding exact optimal solutions and are sensitive to algorithmic parameter settings, which determine the accuracy of their search process. Approximate algorithms can be classified into *single point local-search methods* and *population based methods* [122]. In the case of the former, a single candidate solution is considered at a time and improved by making alterations incrementally until an optimal solution is found. The population based methods evolve and improve a set of solutions simultaneously by iteratively making changes to highly fit solutions in the population until a pre-defined terminating criterion (e.g. number of iterations or fixed run-time) is attained. Hill climbing [58, 240], simulated annealing [58, 78], tabu search [105, 106] are single point local search, while evolutionary algorithms [287], such as NSGAII [69], SPEAII [289] and IBEA [288], and evolutionary strategies [93, 245] are population based methods. Among these approximate techniques, evolutionary algorithms are the most widely used to solve multi-objective optimisation problems. They use the principles of natural selection and evolution to evolve a set of encoded solutions, called a population, towards Pareto-optimality [287].

In summary, requirements engineering and software architecture decisions are generally multi-objective optimisation problems. In this section, we have provided an overview of multi-objective optimisation problems and the algorithms commonly used to solve these problems.

2.2 Goal-Oriented Requirement Engineering

In this thesis, the proposed RADAR modelling language has relations to AND/OR quantitative goal models commonly used in Goal-oriented requirement engineering (GORE). Therefore, this section provides a background on GORE; the common GORE frameworks/approaches (e.g., NFR [180], i* [277], KAOS [264]); quantitative extensions to these approaches and their limitations with respect to making requirements and architecture decisions.

Goal-oriented requirement engineering (GORE) research emerged to address some of the limitations of requirements engineering approaches [212, 219], which failed to capture a proposed system's motive and reason about the system and its operating environments [159]. GORE have gained popularity over the past years as they have been widely used to provide systematic support for eliciting, modelling and reasoning about the proposed software system and its environment. Some research areas where GORE has been successfully applied include: goal elicitation [20, 73, 74, 258, 263], goal refinement and analysis [66, 66, 75, 103, 137, 163, 165, 180, 237, 248], obstacle analysis [19, 256, 266], agent-oriented requirements engineering, handling conflicts [36, 132, 210, 256, 257] and variability [137, 165, 211] in GORE, from requirements to architecture design [55, 108, 125, 261, 279] and analysing security requirements [24, 170, 262, 277, 280].

2.2.1 Goals

Goals are *prescriptive* statements of the objectives a proposed system should achieve through the cooperation of agents that form the system and the environment [125, 260]. Goals can also be defined as statements of high-level objectives of the system, which provide information about the purpose of the system and the guidance on both strategic and operational decisions during the different activities of the system development process [22].

Goals have been classified based on different taxonomies in the literature [56, 259]. The first classification is functional and non-functional goals [259]. Functional goals, also known as satisfaction or behavioural goals are declarative statements about the services the proposed system is expected to achieve, while non-functional goals refer to the constraints on the functions provided by the system. Such constraints could be on the performance, availability, security, scalability and reliability etc. For example, in the design of the plastic card fraud detection system introduced in Chapter 1.2, a functional goal could be: "*the system should send a fraud alert once a fraud is suspected on a customer's card*" and a non-functional goal for this system might be that "*the system should generate such alerts within 8 hours*". Goals can also be classified as either hard goals [65] or soft goals [180]. Hard goals can be satisfied in an unambiguous manner, while soft goals are goals whose satisfaction cannot be established in a clear-cut manner. Soft goals are related to the concept of goal satisficing [159]: they may never be satisfied, which implies that a good enough solution is only needed to satisfice them to an acceptable level. Soft goals are particularly used when comparing alternative goal refinements to determine the one that produces the best contribution towards the overall goal [259].

Goals have also been classified according to the temporal behaviour prescribed by a goal [65, 259]. For example, an *Achieve* (or *cease*) goal can be used to describe some target future state required by the system to satisfy (or deny). A *maintain* (or *avoid*) goal can be used to restrict the behaviour of the system to constantly satisfy (or deny) all future states of the system. *Optimise* goal can be used to compare alternative system designs, and favours the alternatives that best meet some soft target property.

Another classification of goals proposed by Sutcliffe et al. [246] is based on the desired states of the system (e.g. positive, negative, alternative, feedback, or exception-repair) and desired goal level (i.e., policy level, functional level and domain level). These taxonomy of goal types have been used in formulating goals [21, 180, 211], defining heuristics for acquiring goals, refining goals, deriving goals, and for checking semi-formal consistency and completeness [21, 23, 56, 65, 246].

In GORE, the system of interest and the environment where the system operates are considered as a series of *agents* [159, 259]. An agent in this context refers to an active component, such as human, hardware or software component that has some responsibility towards satisfying a goal. A goal whose responsibility is assigned to a human agent is regarded as a *domain assumption*, and when a responsibility of satisfying a goal is assigned to a single software agent, it is termed a *requirement*. While requirements can be enforced by the proposed system, domain assumptions rely on an organisation's policies and norms to be enforced.

2.2.2 Goal Refinement

Goal refinement (or decomposition) is another vital aspect of GORE. Goals can be refined (or decomposed) and related into sub-goals using AND/OR refinement graphs [259]. The AND/OR graphs capture alternative goal refinements [65, 182, 211], possible goal conflicts [40, 65], and how the lower level goals contribute partially, positively or negatively to higher level goals through the use of AND/OR contribution links [56, 180, 182]. The AND–refinement link implies the satisfaction of sub-goals ensures the parent goal is satisfied. An OR–refinement link relates the parent goal to a set of alternative AND–refinements, such that any one of the AND–refinements is sufficient to satisfy the parent goal. The OR–refinements are commonly used to model alternative design choices for satisfying a parent goal.

Goals are continuously refined into sub-goals until their sub-goals are assigned to system agent(s). At this point, only the agent can restrict the goal's behaviour to ensure their satisfaction. OR–responsibility assignments are used to model alternative assignment of goals to agents [126]. This corresponds to alternative design choices called options at decision points, where a decision point could be OR refinement or an OR responsibility assignment in the goal model.

To illustrate the concept of goal refinement, we have used Fig. 2.1, an example goal model of the plastic card fraud detection system introduced in Section 1.2. The parallelograms represent goals. Agents in this figure are the human, software and hardware devices attached to the goals. In Fig. 2.1, the goal **ACHIEVE**[Card Fraud Detected and Resolved] is AND-refined into the goals **ACHIEVE**[Card Transactions Processed] and **ACHIEVE**[Compromised Card Resolved]. The goal **MAINTAIN**[Accurate Fraud

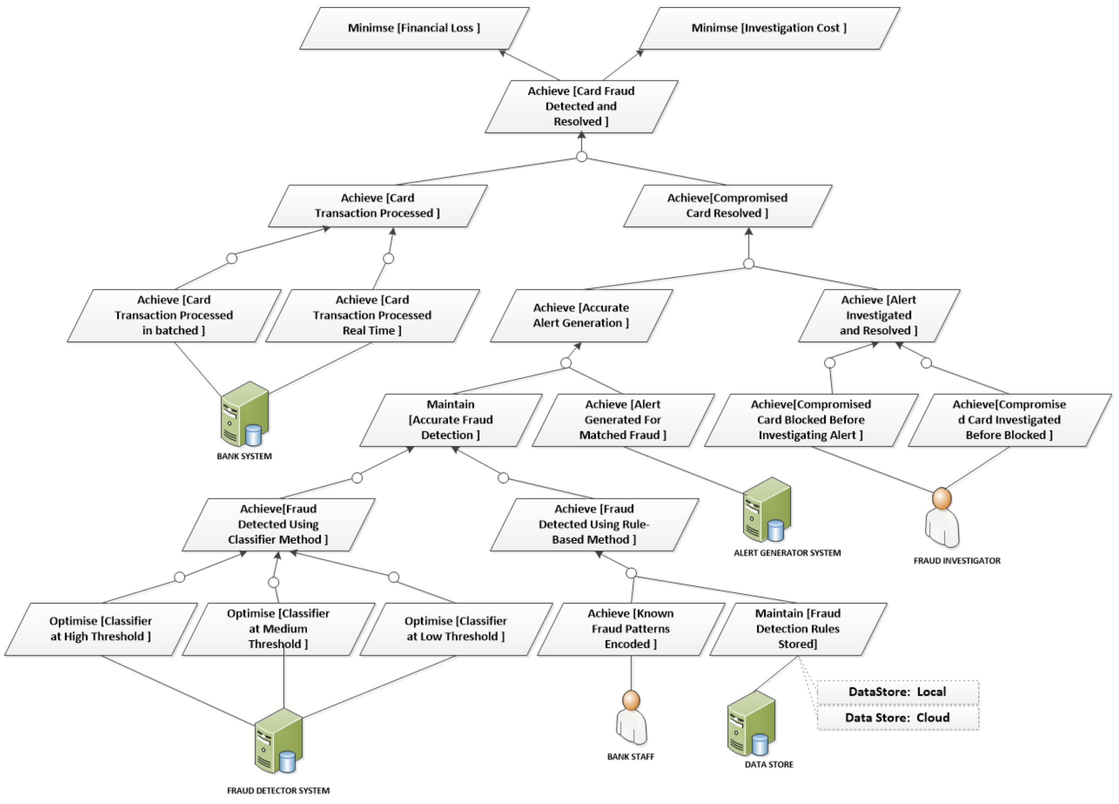

Minimse [Financial Loss]
Minimse [Investigation Cost]
Achieve [Card Fraud Detected and Resolved]
Achieve [Card Transaction Processed]
Achieve[Compromised Card Resolved]
Achieve [Card Transaction Processed in batched]
Achieve [Card Transaction Processed Real Time]
Achieve [Accurate Alert Generation]
Achieve [Alert Investigated and Resolved]
BANK SYSTEM
Maintain [Accurate Fraud Detection]
Achieve [Alert Generated For Matched Fraud]
Achieve[Compromised Card Blocked Before Investigating Alert]
Achieve[Compromised Card Investigated Before Blocked]
Achieve[Fraud Detected Using Classifier Method]
Achieve [Fraud Detected Using Rule-Based Method]
ALERT GENERATOR SYSTEM
FRAUD INVESTIGATOR
Optimise [Classifier at High Threshold]
Optimise [Classifier at Medium Threshold]
Optimise [Classifier at Low Threshold]
Achieve [Known Fraud Patterns Encoded]
Maintain [Fraud Detection Rules Stored]
DataStore: Local
Data Store: Cloud
BANK STAFF
DATA STORE
FRAUD DETECTOR SYSTEM

FIGURE 2.1: AND/OR goal model of the plastic card fraud detection system.

Detection] is OR-refined into the goals **ACHIEVE**[Fraud Detected Using Classifier Method] and **ACHIEVE**[Fraud Detected Using Rule-Based Method]. The agent *fraud investigator* attached to the goal **ACHIEVE**[Compromised Card Investigated Before Blocked] is expected to manually carry out the investigation on a suspected compromised account (this is an example of a domain assumption), whereas the agent *alert generator* attached to the goal **ACHIEVE**[Alert Generated For Matched Fraud] is required to generate alert when a card transaction is suspected to be fraudulent (an example of a requirement). The leafgoal **MAINTAIN**[Fraud Detection Rules Stored] could be assigned to a system agent *Data Store*, with option choices of either using a local database or cloud-based database system (this is an example of OR responsibility assignment).

2.2.3 Goal-Oriented Requirement Engineering Approaches

Many goal modelling approaches exist, but the main ones include: NFR [180], i^*/TROPOS [47, 276], URN/GRL [14] and KAOS [264]. These approaches rely on having sufficient conceptual meta-model to systematically capture a complete and consistent functional and non-functional requirements of a proposed system [65]. These goal modelling approaches provide diverse notations, refinement patterns and support for analysing goals using informal, semi-formal and formal methods [20, 103, 164, 165, 180, 237, 256]. In the rest of this section, we present a brief description of the core ideas of the main goal-oriented approaches.

2.2.3.1 NFR Framework

The NFR framework [56, 180] was proposed to reason, model, refine and analyse non-functional requirements. The NFR framework is a process oriented approach that rationalises the development process with respect to non-functional requirements [180]. Its main activities deal with the vagueness, priorities, dependencies and trade-off among non-functional requirements; also, it deals with capturing and exposing the positive and negative contributions of design alternatives on non-functional requirements. The NFR framework's main modelling tool is the Softgoal Inter-dependency Graph (SIG), which is a graph that represents non-functional requirements as softgoals and their AND/OR–refinements, positive/negative contributions, operationalisation and claims. The softgoal operationalisation models low-level technical details of satisficing non-functional requirements softgoals, and the softgoal claims enable architects to document design rationale for the refinements of softgoals [56].

The NFR framework uses a label propagation algorithm to determine which design decisions best satisfice the high-level non functional requirements[180]. The algorithm starts from the leaf-level softgoals, which encode the design decisions of the architect, and traverses the graph upward towards the top-level softgoals, while considering the labels on the softgoal refinement links. This type of alternative operationalisation guides an architect in selecting the design that gives the best contributions towards the high-level non-functional requirements.

The NFR framework provides an architect with three main types of directory or catalogue for storing a list of design knowledge [56]. First, it provides directory for storing the ideas about the different types of non-functional requirements, such as performance, availability, security, scalability and reliability etc. Second, it provides a directory for storing methods used in guiding softgoal refinements and operationalisation. For example, this catalogue could have a method which states that when a softgoal is applied to a data item with different sub-components, then the softgoal can be refined into other subgoals for each of the components in the item. Third, the correlation directory encodes knowledge to assist in identifying softgoal dependencies.

2.2.3.2 i^* Framework

The i^* framework [276, 278] is an agent-oriented modelling framework to support both early and late requirements engineering phases. At the early phase of requirements engineering, i^* framework facilitates domain analysis by enabling visualisation of the current organisational/business processes, the stakeholders relationships and objectives. The models developed in this phase provide rationale for the proposed software and organisational processes. In the late requirements engineering phase, emphasis is on how to use the i^* models to present a new system design and organisational processes, and an evaluation of how the processes and design impact the stakeholders' functional and non-functional requirements.

The i^* modelling approach is based on the idea of intentional actor and intentional dependency [170]. Actors in this context can be agents of the proposed system, a role and a position. An agent could be a software, hardware or human designated with certain functionalities; a role is an "abstract actor embodying expectations and responsibilities" [170], e.g. a *Fraud Investigator* in the Plastic Card Fraud Detection System; and a position is a list of roles carried out by a single agent. In general, actors have certain attributes, notably goals, capability and responsibility. They depend on other actors to achieve their goals and task executions efficiently. However, an actor is at risk when other actor(s) it relies on fail to deliver. Thus, it becomes imperative for actors to strike an equilibrium between the opportunities they are presented with and the possible risks in order to achieve their objectives. Intentional dependency captures the relationship between actors. For example, an actor may depend on another actor(s) to achieve a goal. Dependency relationships can be classified according to the goal, softgoal, task and resource [170].

The i^* framework consists of two main components, namely the Strategic Dependency (SD) model and the Strategic Rationale (SR) model. An SD model is "a network of intentional dependencies"[170]. An SD model is used to analyse: actor dependencies, organisational changes due to the proposed new system and the possible opportunities and risks an actor could face. An SR model enables the exploration and description of the rationale for the system and organisational processes in terms of the process elements (e.g. goals, softgoals, tasks and resources) and their relationships. Unlike SD model, which focus on the external relationships among actors, SR model focus on the internal processes of actors, thus providing fine-grained abstraction for representing intentional features of the organisations and systems.

In the i^* model, AND/OR refinements of process elements, such as goals, softgoals, tasks and resources, can be achieved using the *decomposition*, *means-ends* and *softgoal contribution links*. The decomposition link attaches a goal/task with a subtask and soft goals. The means-ends links are mainly used for goals and the alternative means of achieving them. The softgoal contribution links can be used to relate a softgoal/goal/task to other softgoals. The softgoal contribution link, which are similar to the contribution links in NFR framework, uses two levels of positive ("+" and "++") and negative ("–" and "– –") contributions links. The i^* meta-framework, which describes the semantics and constraints of the i^* are elaborated in a tool called Telos [181] –a tool that enables diverse analysis of the i^* models, such as consistency checking and scalability management.

2.2.3.3 TROPOS Framework

The TROPOS framework [47] is a requirement-driven agent oriented development approach that guides the analysis, design and implementation of agent-based systems. This framework models, reasons and analyses system requirements and design choices using the i^* framework. However, it extends i^* by providing textual symbols for i^* models and methodology for describing dynamic constraints in first order temporal logic [47].

2.2.3.4 URN/GRL Framework

The GRL framework [14] is one of the sub-languages of a semi-formal, goal and scenario based modelling language known as User Requirements Notation (URN) [1]. GRL means Goal-oriented Requirement Language. GRL combines the i^* and NFR frameworks to model *goals* (represented by rounded rectangle); *softgoals* (represented using cloud shape); *alternatives* and the impact of decisions on high-level softgoals; and *rationales*, which are declared as beliefs and represented using eclipse shape. GRL is implemented in the jUCMNav tool [2], which is currently integrated in the Eclipse development environment.

2.2.3.5 GBRAM Framework

GBRAM [19, 20] is a Goal Based Requirements Analysis Method that emphasises early elicitation and abstraction of goals from diverse sources. The framework supports goal analysis and refinement process. Goal analysis entails three main activities, namely the exploration, identification and organisation activities, in which the different sources of information are *explored* in order to *identify* goals and agents responsible for those goals, and then *organise* the goals using dependency relations (similar to inter-actor dependency in i^* models) that declare which goals must be satisfied before other goals. Similarly, goal refinement involves three activities, such as refinement, elaboration and operationalisation activities, i.e. the identified goals are refined so that redundancy among goals are discarded; goals are elaborated to identify hidden goals and obstacles to the goals; and operationalisation of the goals to identify agents (e.g. entities or processes) that ensure the goals. GBRAM does not support graphical syntax, rather it uses goal schemas to specify agents, goals and stakeholders in textual form.

2.2.3.6 KAOS Framework

KAOS means Keep All Objects Satisfied [266] or Knowledge Acquisition in autOmated Specification [65]. The KAOS framework is a multi-paradigm language characterised by a two-layer structure, namely: (i) the outer conceptual modelling layer for specifying concepts, their attributes and relationships with other concepts; (ii) an inner formal layer for accurate reasoning about the concepts [162]. A KAOS specification consists of core models, which can be related to one another by inter-model consistency rules. The core models are enumerated below [162]:

1. A *goal model* made up of a two-layer structure. The first layer is an outer semantic net [38] used to declare goals and their links. The second layer is a textual component used to define goals in natural language, or formally in real time temporal logic.

2. An *object model* which represents objects of interest, their attributes and association links to other objects. Objects are classified as entities, relationships, agents or events.

3. An *agent responsibility model* which represents assignment of goals to agents responsible for ensuring the satisfaction of the goals. Goal responsibility assignment provides the basis for halting a goal refinement process. Thus, when a goal is assigned to a single agent, such goal cannot be refined further. Alternative goal responsibility assignment is possible using the OR–refinement link.

4. An *operation model* that represents transitions between states within an application domain. Operations are defined by domain pre, post and trigger conditions and the operational model uses these conditions to specify operational requirements, which are related to goals by operationalisation links and ensure the achievements of goals assigned to agents.

2.2.4 Quantitative Goal Oriented Requirements Engineering

The GORE approaches described in previous section, such as the NFR and KAOS frameworks, have quantitative extensions [3, 4, 44, 57, 64, 81, 126, 165, 199, 222, 223, 265, 270] that provide more precision for decision support than qualitative approaches [103, 165]. In addition, quantitative GORE approaches help to clarify decision models and evaluate the impact of decision choices on the stakeholders' objectives.

The NFR framework and its variants (e.g i^* [276, 278], TROPOS [47] and GRL [14]) have been quantitatively extended by replacing the "++/+" and "– –/–" symbols on the goal refinement contribution links with numbers (usually between -1 and 1), and attaching weight values (between 0 and 1) to the leaf goal/softgoal to denote the extent of satisfaction of the goal/softgoals [3, 4, 81, 199, 270]. The degree of satisfaction of a goal/softgoal is computed using the weighted average of the degree of satisfaction of the sub-goals.

Giorgini et al. [103] developed a formal framework that uses NFR AND/OR refinements to qualitatively and quantitatively reason about the goal model of a US car manufacturing company. The authors also presented a qualitative and quantitative axiomatisation for goal model primitives, and label propagation algorithms for each axiomatisation.

This approach does not capture uncertainty in decision models explicitly and lacks tool support for decision analysis.

Affleck et al. [3, 4] extended the NFR framework by proposing an optimisation model that aims to maximise the attainment score (i.e. a score representing the degree of satisfaction of the softgoals); minimise denial of softgoals; and avoid superfluous implementation through the acceptance and rejection of leaf softgoals according to their scores and decomposition. Developing the optimisation model involves converting the NFR softgoal inter-dependency graph (SIG) to a directed graph with associated constants and variables, and the objective function is constructed from the graph constants and variables. The authors evaluated the optimisation model through simulation-based analysis of synthetic test data. While this approach aims at tackling the scalability issues in SIG, the optimisation model does not support elaboration of domain-specific decision models, and does not take into consideration the fact that real world problems have multiple objectives, for example, the model did not consider the cost of alternative operationalisations in determining the optimum solution; it analysed uncertainty in the graph variables through sensitivity analysis, which does not consider the probability of change in decisions that optimise stakeholder goals and probability of changes in model parameters' values [166]; and finally, it lacks tool support for automated analysis.

Pasquale et al. [199] introduced an approach to quantitatively analyse security goals and their trade-offs between other goals, such as availability and cost, for a service-based email system. Their approach consists of three phases, namely: modelling, formalisation and analysis phases. The modelling phase involves developing a threat model, goal model and asset model of the security concerns and goals of the system. The formalisation phase combines the models using mathematical representations of metric functions that quantify the satisfaction of the security goals and concerns. Finally, the analysis phase involves encoding the formalised model into a Boolean satisfiability problem, by adding a set of constraints to the model and then solving the problem using SMT Solver (Z3). This approach is semi-automated as the analysis phase requires manual encoding of the satisafiability problem in the solver, thereby requiring that decision makers have an understanding of how the solver operates. In addition, the authors compute goal satisfaction using point based estimate using range normalisation i.e. normalise the final output between -1 and 1.

Wei et al. [270] developed an approach, driven by quality requirements, for making implementation decisions according to the quality goals of a system, in the context of internet aware. The approach consist of four steps, namely: the model generation; satisficing status assignment; model reasoning and decision making. The model generation step involves transforming the goal models to a tree model and then to a symbolic formular, according to the syntax of the proposed symbolic modelling language $\sum$. The satisficing status assignment assigns quantitative (e.g. numbers in the range of 0 and 1) and qualitative (e.g. "++" and "– –") values to the generated symbolic formular. The next step is to reason about the model in order to determine all candidate implementation decisions. Model reasoning can be quantitative, qualitative or hybrid of both; it is quantitative when the model's contributions and satisficing statuses are fully quantified; it is qualitative when they are not quantified and hybrid when partially quantified. Finally, the implementation decision is made by incorporating users' preferences and priority on goals. This approach, however, does not handle uncertainty in model parameter, and while it has tool support, there is still the issue of the scalability and complexity of goal models. The effect of this is that symbolic formula generated in step one becomes complex to read when dealing with large goal models.

Qin et al.[171] proposed a goal-based decision making framework that clearly demonstrates the link between goal-oriented requirements engineering and multi-criteria decision making technique. Using a case study that involves selecting an IT solution for registering customer profiles, in the insurance domain; they evaluated their approach by importing quantitative goal models developed in the jUCMNav tool [213] into Excel, where strategies for alternative selection and/or elimination is implemented. The strategies used are disjunctive rules and Rank Order Centroid (ROC) formular. Disjuctive rule simply states that an alternative that exceeds certain threshold are jettisoned, whereas the ROC formular is used to convert the importance ranking assigned to different alternative's attributes (e.g., cost, scalability and interoperability) to quantitative weights. While this approach tackles decision making in a multi-objective optimisation context, it lacks an integrated tool support for analysing decision models and suffers from the scalability and readability problems in cases where the decision models are large.

While the techniques described above enable the elaboration of domain-specific decision models in requirements and architecture decisions, the numbers (-1 or 1) assigned to the contribution links in the goal models have no physical interpretations in the application domains [126]. In contrast, Letier et al. [165] proposed a quantitative extension of the KAOS framework [65, 266] where levels of goals satisfaction are defined using domain-specific metrics with physical interpretation. They developed a technique that specifies partial degrees of goal satisfaction, and quantifies the impact of alternatives on the extent of goal satisfaction in terms of refinement equations, which are defined over random variables involved in the system‘s functional goals. They computed their objective functions for higher-level goals using estimated probability distribution functions from the leaf or low level quality variables. This approach does not cater for automated analysis of goal models, as the analysis is done through an ad-hoc process that involves the use of analytical and numerical methods.

Heaven et al. [126] leverages the quantitative goal refinement model based on the KAOS framework presented in [165] to simulate and optimise the impact of alternative system designs on high level goals using multi-objective genetic algorithm (NSGA2). They found the Pareto optimal design options among the alternatives options that optimises the achievement of 8 and 14 minutes response time of the London Ambulance System, at a low cost. This approach, however, lacks tool support and requires manual encoding of the simulation models in a general programming language, such as R and MATLAB.

Sabetzadeh et al. [222] proposed a goal based approach, using the KAOS framework, for assessing the satisfaction of new technology goals, such as safety and reliability goals. The approach uses sensitivity analysis to figure out system components that needs further improvements in satisfying the system goals. This approach involves the following steps: (1) developing a goal model of the system; (2) plan and collate evidences in order to quantify the probabilities that low level goals are satisfied, probabilities of obstacle blocking low level goals and probabilities of risk occurring due to incomplete decomposition of goals; (3) Elicit the probabilities from experts according to the evidences presented; (4) propagate the elicited probabilities from the leaf goals to compute the probability distributions for the satisfaction of the high level goals; (5) given that uncertainty about the satisfaction of the overall goal exist, perform sensitivity analysis to determine which model parameter inputs lead to great variations in the model output parameters. Although this approach has tool support that is presented in [223], it is only targeted at safety and reliability technological quantification and the complexity of goal models limits the scalability of this approach.

In summary, Section 2.2 describes existing qualitative and quantitative Goal-oriented requirement engineering (GORE) approaches used in making design decisions in Requirements engineering. Qualitative goal oriented approaches such as the NFR framework [180], *i**/TROPOS [47, 276], URN/GRL [14] and KAOS [264] allow one to model, reason and analyse the impact of decisions on stakeholders' goals. However, they only provide information about how the decisions positively or negatively impact stakeholders' goals. They provide no quantitative information about the impact of alternatives on goals. Quantitative goal oriented approaches, however, are more accurate in modelling, reasoning and analysing domain-specific decision problems [3, 199] [126, 165]. But they require higher modelling effort and generally have limited automated tool support for decision analysis.

2.3 Software Release Planning Decisions

Requirements modelling and analysis are critical activities carried out during requirements engineering of a proposed software system [264]. A prominent research area that tackles requirements decisions during modelling and analysis of a proposed system is Software Release Planning [208]. Since this thesis proposed to develop a novel modelling language and automated decision analysis technique for requirements engineering decisions, this section reviews the state-of-the-art software release planning decisions approaches.

Software release planning addresses decisions related to the selection and prioritisation of requirements when developing a sequence of consecutive product releases that satisfy important and often conflicting constraints, such as benefits to business and stakeholders, cost of implementation, development effort and risk, delivery time and dependencies among requirements [208].

Software release planning decisions can be viewed from two perspectives [7]: first, as strategic release planning [215], where decision-makers make relevant decisions about the prioritisation and assignment of requirements (features) to different releases in a way that both resource and technical constraints are met and at the same time satisfying the stakeholders' goals. Second, as operational release planning [5, 25], in which resources, such as developers and testers, are allocated to the tasks required to implement the set of requirements agreed for a particular release.

Release planning decisions are generally complicated by the uncertainty about the impact of decision choices on the business and technical constraints. Poor decisions, such as abrupt exclusion and inclusion of requirements, may lead to stakeholders' dissatisfaction; a software release plan that does not meet the technical, resource, budget and risk constraints [218].

The literature has several decision support methods for modelling and analysing release planning decision problems. Examples of these methods are cost-value approach [143], next release problem [30], EVOLVE release planning method [216], incremental funding method. Many of these methods have been applied with techniques from search based software engineering, which offer adaptive automated and semi-automated solutions. The major limitation of these methods is that they rely on pre-defined generic equations (generally weighted sums) to assign scores to the objective metric(s), e.g. 'cost', 'revenue' and 'value', for each alternative design. The metric scores in these methods usually represent abstract (non verifiable) quantities rather than obtaining metric scores from domain specific measurable values. For example, using financial metric that is expressed in monetary units (e.g. in Dollars or Euros). Also, the generic equations do not adequately define the stakeholders' goals and correctly predict the impacts of decisions on these goals. The remaining part of this section describes some of the quantitative requirements decision support methods applied to software release planning.

2.3.1 Cost-Value Approach

Traditional cost-value approach [143] uses the relative costs and value assigned to each requirement to compute a ranked set of requirements, and a cost–value plot of the requirements is used by stakeholders as a basis of decision-making. Examples include; AHP process [221] , Karl Wiegers Requirements Prioritisation Model [273], Rational Focal Point and Volere requirements prioritisation technique [209].

Karlsson et al. [144] proposed an efficient cost-value approach, based on AHP [221], for managing and quantifying the differences in requirements importance. Using pre-defined criteria, such as requirement importance and cost, the author demonstrated how their approach could be used by decision-makers to decide on what set of requirements to be selected according to individual requirement contributions.

2.3.2 Next Release Problem Model

A common strategy to software requirements selection and prioritisation, which are vital activities of release planning decisions, is the Next Release Problem (NRP). The NRP tackles the selection of a deliverable set of requirements within a company's budget and at the same time satisfying the stakeholders. This sections reviews the NRP optimisation approaches to release planning decisions. We categorise these approaches based on the novelties introduced to the NRP model, notably: single-objective optimisation approach, multi-objective optimisation approach and uncertainty handling in the NRP model.

2.3.2.1 Single Objective Optimisation Approach to NRP

Bagnal et al. [30] was first to formulate the NRP as a single objective optimisation problem. The author aimed to maximise the cumulative measure of stakeholders' importance given the cost constraints, and found the subset of customers whose requirements are to be satisfied in the next release of an existing software product.

Ruhe et al. [217] extended the NRP model by proposing an evolutionary and iterative approach, called EVOLVE, which is aimed at maximising the advantages of optimally allocating requirements incrementally to software releases, subject to resource and budget constraints. The EVOLVE approach evaluates and optimises the degree to which requirements ordering conflict with stakeholder priorities and balances the required and available resources.

Albourae et al. [9] proposed a release re-planning process model. Their approach uses Analytical Hierarchy Process (AHP) to compare old requirements to the newly added ones, and then applied a greedy algorithm to obtain the most promising requirements that accommodate the varying demands driven by the market. Baker et al. [31] also formulated the ranking and selection of candidate software components as a series of feature subset selection problem. Their approach was evaluated using large scale data sets from a telecommunication company. They employed the greedy and simulated annealing algorithms to the formulated problem and compared the optimum solution obtained with human experts' judgement. Their empirical results show that both algorithms convincingly outperformed the human experts.

The models and problems described above do not handle dependencies between requirements. However, in reality dependencies, such as precedence, and , or, cost and value based, exist among requirements and they tend to complicate the requirements selection and prioritisation decision making process [282, 285].

Many research studies have been conducted to handle requirements inter-dependencies. Carlshamre et al. [46] conducted a survey studying requirement dependency, such as and, requires, temporal (precedence) or, cost and value dependencies. In this study, there were 5 distinct sets of requirements, each including 20 high-priority requirements of 5 distinct products from 5 different companies. Three out of the five products have been implemented and their improvements are market-driven. The remaining two products were bespoke software. The author found that bespoke software often have functionality based dependencies (e.g, and/requires) while market-driven software products tend to have value based dependencies.

Van den Akker [255] studied a variation of the NRP model through the combination of requirement selection and scheduling. This work applied a mathematical formalism and optimisation technique based on Integer Linear Programming (ILP) to find exact optimal solutions to the maximisation of projected revenue in the presence of budgetary constraints within a given time period. The author, in addition, modelled team transfers, extensions in release deadline and resources, and considered five different types of requirements dependencies, such as implication, combination, exclusion, revenue–based and cost–based.

2.3.2.2 Multi Objective Optimisation Approach to NRP

The release planning models and approaches discussed thus far are single objective optimisation problems. However, most real world requirements decision problems have multiple objectives. The challenge with single objective requirements decision problems is that the optimisation of one objective may be at the expense of another, thereby leading to a bias in the search process towards one of the objectives.

Recently, many requirements selection and prioritisation approaches have been viewed as a multi-objective decision problems. Zhang et al. [283] was the first to extend the Next Release Problem by generalising it to a Multi-Objective Next Release Problem (MONRP). Unlike the formulation by Bagnal et al. [30], Zhang treated the cost constraints as an objective. The author used search based techniques, such as GA and NSGAII, to find the subset of customers' requirements to be fulfilled that will give maximum value to the stakeholders at a minimum cost.

Finkelstein et al. [91, 92] introduced the concept of fairness in requirement analysis and optimisation in order to balance the satisfaction among stakeholders with different preferences. They introduced three fairness models, which are fairness on the absolute number of fulfilled requirements; fairness on the absolute value of fulfilled requirements and fairness on the percentage of value and cost of fulfilled requirement.

Zhang et al. [284] extended their previous work on the MONRP [283] to find an optimal set of customers' requirements that balances the initial set of requirements to be selected, i.e. "needs of today", against the requirements to be selected later i.e. "needs of the future". They formulated three objectives which are: minimise the cost of requirement implementation and maximise stakeholders' *value* for today and *value* for the future. Zhang et al. [282, 285] further extended their previous work on the MONRP[283] by conducting an empirical study to tackle requirements interactions and dependencies, such as and, or, precedence, cost– and value–based constraints.

Ruhe et al. extended the EVOLVE [217] method to propose EVOLVE*, a hybrid method that combines experts' judgement with mathematical models and evolutionary computation, to find the optimal solutions that maximises time, benefit and quality while planning two releases ahead. They also considered requirement changes and two requirements interaction relationship, such as coupling and precedence. Saliu et al. [226] extended the EVOLVE* approach to include proactive analysis of risk involved when combining existing system requirements to new ones and the benefit of estimating the effort required to add the requirements.

Saliu et al. [225] presented a decision support approach that formulates the release planning problem as a bi-objective optimization problem. In their work, they proposed a tool called Bi-Objective Release Planning for Evolving Systems (BORPES), which is aimed at optimizing the overall value of release plans from both the business perspectives and the implementation perspectives.

Durillo et el. [76] conducted an empirical study to check the performance of three state of the art multi-objective evolutionary algorithms (e.g. NSGAII, MoCell and PAES) on the MONRP [283], using a benchmark that consisted of six academic problems and a real world data set from Motorola. Srinivas et al. [243] also introduced a Quantum-inspired elitist multi-objective algorithm (QEMEA) to solve the MONRP, using six test problems from the literature. The authors use QEMEA to improve and balance the exploration and exploitation of solutions within the search space.

Veerappa et al. [267] studied the cost-value requirements selection problem in the context of MONRP and proposed a hierarchical clustering technique to identify a group of strongly related Pareto optimal solutions for requirements selection problems. The author used a series of visualization tools to support decision makers in (i) understanding how group of solutions are spread on Pareto front (ii) identifying areas where strongly divergent solutions achieve similar objectives. iii) make incremental decisions by first selecting among group of solutions before selecting one of the variants within the chosen group.

2.3.2.3 Uncertainty Handling in NRP

The release planning models and methods described so far do not handle uncertainty explicitly in the decision model. They used point-based estimation of model parameter inputs rather than using range of model parameter values. The point estimates fail to take into consideration the inherent uncertainty in model parameter values, and such underestimation of uncertainty may result in project failure and bring risks into the project [166].

Uncertainty can be analysed either as a post-optimisation process or during the optimisation process. The post-optimisation process includes uncertainty analysis and sensitivity analysis. Uncertainty analysis measures the total uncertainty about conclusions of the model and how the uncertainty propagates, i.e., it provides quantitative information about the probable occurrence of different outcomes [241]. Sensitivity analysis, however, concerns the study of the effect of uncertainty about model inputs on the cumulative uncertainty of model outputs [227, 241]. That is, sensitivity analysis provides useful information about the input model parameters that causes the highest variations in the output parameters.

With respect to handling uncertainty as a post analysis, Harman et al. [123] introduced an exact requirements sensitivity analysis approach to help users explore sensitivity of requirements attributes, such as cost estimates, for the NRP using both synthetic and real-world data. The optimisation algorithm used an exact algorithm instead of approximate algorithms in order to be sure that the variations in results obtained from the original and perturbed NRP problems are actually due to inherent sensitivity and not due to the stochastic nature of the approximate algorithms. The exact algorithm used is a variant of dynamic programming, called Nemhauser-Ullmann algorithm [186].

Al-Emran et al. [6] applied a post analysis integrated method that combines Monte Carlo Simulation and process simulation, in order to study the impact of uncertainty in release planning. The focus was on operational and product release planning.

In dealing with uncertainty as part of the optimisation process, Li et al. [169] studied the trade-off between the robustness and performance of optimal solutions. They optimized the fitness value (measures design solution performance) and a robustness index (quantitative measures for solutions that are sensitive to parameter variations), and presented a metric for uncertainty to guide multi-objective optimization problem. In their work, the uncertainty of a parameter's actual value and solutions were represented as interval and tolerance region (uncertainty size) respectively.

Paixao et al. [196] introduced a scenario based robust formulation of the NRP to obtain robust optimal solutions. They maximised the overall requirements importance for all possible scenarios (i.e. values that symbolise the occurrence of certain events in different context) subject to cost constraints. This approach produces a conservative robust solution that does not cater for the impact of uncertainty in the worst case scenario.

Li et al. [167] extended the MONRP formulation by integrating Monte Carlo Simulation in evaluating alternative designs and then finding robust optimal solutions that maximise the expected revenue, expected cost and minimise the size of uncertainty. The author introduced two notions of uncertainty measurement, which are the probability that actual cost exceeds a set value (failure possibility) and the size of uncertainty region. In a later work, Li et al. [168] proposed an approach to handle algorithmic uncertainty resulting from the use of stochastic optimisation algorithms. They achieved this by augmenting a variant of dynamic programming (Nemhauser-Ullmann algorithm) with Conflict Graph that transforms the NRP model to a search tree whose leaves are independent sub-problems.

2.3.3 Incremental Funding Method

The Incremental Funding Method (IFM) [70, 71] was developed to address the inherent risk in implementing a monolithic software system in a single development period. The approach advocates incremental delivery of software features in bits in order to deliver early business values to the stakeholders. The advantage IFM has over the other release planning methods, such as the Next Release Problem methods, is that it supports financially informed decision making on projects that are yet to be funded and the flexibility of the approach to dynamic project environments.

According to Denne et al., the IFM is “a financially informed approach to software development, designed to maximize returns through delivering functionality in chunks of customer valued features, carefully sequenced so as to optimize Net Present Value (NPV)” [71]. The IFM typically breaks down a monolithic software system into smaller components, known as Minimum Marketable Features (MMF) that can provide market value to customers. The MMFs are assumed to be implementable over a specific period and their development and delivery are sequenced in order to minimise the investment cost to the business, maximise the generation of revenue quickly and hopefully move the projects to a self-funding status.

Figure 2.2 illustrates a typical pattern that a successful project that benefited from the application of the IFM should follow. In the IFM parlance, a *self funding status* describes the situation where an initially funded project starts to generate revenue, which can then be used to develop additional MMFs. Consequently, as more MMFs are developed and rolled out to the market, stakeholders gradually recuperate the initial investment costs. This is called the *repayment period.* A *break even point* is reached when the Net Present Value is zero. Beyond this point, stakeholders accrue profit from the software project.

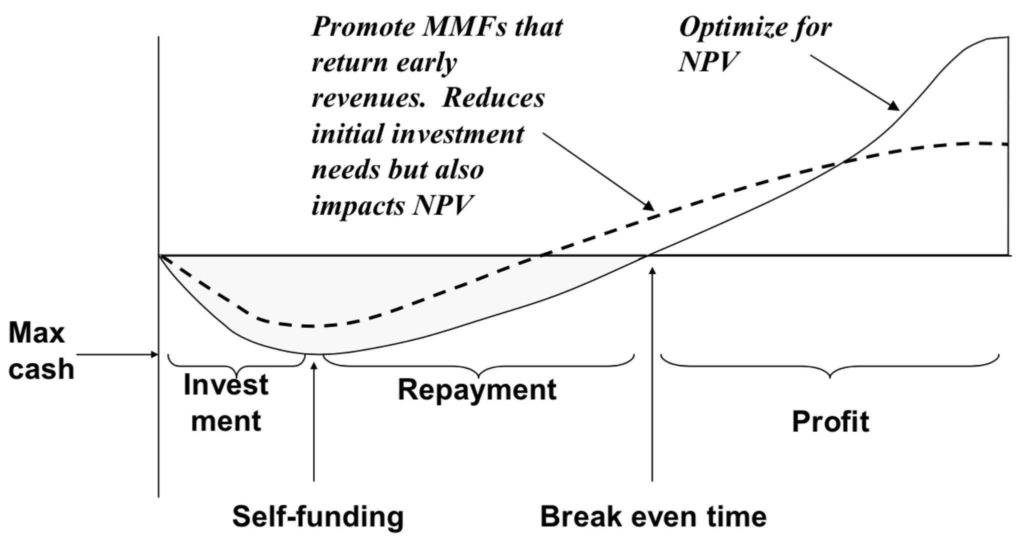

Promote MMFs that return early revenues. Reduces initial investment needs but also impacts NPV
Optimize for NPV
Max cash
Invest ment
Repayment
Profit
Self-funding
Break even time

FIGURE 2.2: Ideal financial pattern of a project that uses the IFM. [71].

The IFM has been tackled traditionally using the IFM heuristic [71] and greedy method [10]. The IFM heuristic selects the most promising MMF by analysing the whole MMFs according to the generated revenue in the current and future periods, while the greedy approach to IFM optimises the NPV by selecting the next MMF according to the MMF with the highest NPV in the current development period.

The original IFM model has a few limitations: (i) the IFM optimisation was not solved using any of the state of the art multi-objective optimisation algorithms, such as NS-GAII [69], SPEAII [289] and IBEA [288]; (ii) it does not handle uncertainty in the estimation of future cash values in cost and revenue; (iii) the model ignores the fact that competition exist in business environments; (iv) the authors treated the IFM as a single objective optimisation problem, thereby ignoring the fact that real world software engineering problems tend to have multiple and conflicting objectives; (v) the decision model equations use weighted sums of cash flows to compute the NPV of alternative delivery sequences; (vi) there is lack of tool support for automated decision analysis. Modellers have to encode the decision models in general programming language.

Although there have been some extensions to the original IFM, these extensions only addressed the first four limitations. Alencar et al. [10] extended the IFM with a statistical approximation approach to obtain approximate delivery sequences that are close to the exact delivery sequence with some level of confidence. In particular, their approach was applied on a large set of interconnected MMFs and architectural elements. Murray et al. [45] extended the IFM to incorporate uncertainty in the cash flow, and to determine adaptable investment policies, depicted using decision trees. They modelled uncertainty using triangular probability distribution and Monte Carlo Simulation. Eduardo et al. [63] also extended the IFM to introduce the application of Game Theory in maximising the financial returns of a software projects in the context of a duopolistic market. Their main contribution was to model the IFM as a strategic game to represent the competition that exist in real world business settings. Oni et al. [194, 195] extended the IFM from a single objective optimisation problem to a multi-objective optimisation problem. They considered three objectives, namely: NPV, investment cost and the investment risk, and also represented uncertainty in model parameters using the triangular distribution and Monte Carlo simulation.

So far, there is no improvement of the IFM model with respect to the last two limitations highlighted. However, the modelling language and automated decision analysis tool proposed in this thesis addresses these concerns.

In summary, Section 2.3 described some related work to this thesis on Requirements prioritisation and release planning which are vital activities in requirements engineering decisions. We reviewed existing quantitative software release planning approaches such as the Cost-Value approach, Next Release Problem model (single objective optimisation approach, multiple objective optimisation approach and uncertainty handling in NRP) and Incremental Funding Method (IFM). These approaches ease their applicability by relying on pre-defined generic equations (generally weighted sums) to assign scores to the objective metric(s), e.g. 'cost', 'revenue' and 'value', for each alternative design. However, such generic equations may not express the actual stakeholders' goals and may not correctly predict the impacts of alternatives on these goals. In addition, existing software release planning approaches generally lack automated technique for analysing uncertainty and informing decision makers about the financial value of reducing uncertainty in a decision model.

2.4 Software Architecture Decisions

Early decisions about the architecture of a software system have significant impact on the business and quality goals of a system. Deciding which architecture among alternative software architectures would satisfy these goals is not trivial. This is due to the multiple stakeholders and their conflicting goals; the presence of many alternative architecture solutions; and uncertainty about the impact of an architectural solution on the stakeholders' goals. Thus, there is a need to support decision-makers in evaluating the impact of alternative architectural solutions on stakeholders' conflicting goals and then selecting the one that gives the best trade-off between these goals.

Many approaches exist to help software architects model and analyse software architecture decision problems, such as selecting and evaluating software architectures that meet certain functional and non-functional requirements; the selection of software and hardware components, their replication, the mapping of software components to available hardware nodes, and the overall system topology. Many of these methods use generic decision models and objectives, and assess alternative architectural solutions quantitatively and choose the ones that satisfy the desired stakeholders' goals. Examples of these methods include: the Architecture Tradeoff Analysis Method (ATAM) [147], the Cost Benefit Analysis Method (CBAM) [150], GuideArch [84] and Multi-objective Decision Analyser (MODA) [166].

2.4.1 Architecture Trade-off Analysis Method— ATAM

The Architecture Trade-off Analysis Method (ATAM) [147, 148] evaluates the effects of architecture decisions on the quality requirements of a proposed software system. It provides insights into the relationships that exist among the quality requirements. ATAM focuses on the use of scenarios to identify critical stakeholders' goals; alternative architectural styles and approaches needed to satisfy the goals; potential risks and possible means of mitigating them; and sensitivity points within the architecture and trade-off points. The ATAM steps are as follows [147]:

1. Elicit scenarios from representative stakeholders. Scenarios can be a description of: (1) the system usage; (2) how the system will accommodate increased load;

(3) extreme changes in system functionalities and running platforms/environments [148]. Scenario elicitation helps in assigning responsibilities of achieving functional and non-functional requirements to agents; and to facilitate communication and common understanding of the system goals among stakeholders.

2. Elicit requirements/constraints/environment. This step also entails identifying and characterising attribute-based requirements.

3. Using architecture views (e.g. process view, module view, class view or data flow view [32]), describe and model the multiple and competing candidate system architectures derived from analysing the requirements and scenarios. The architecture models (qualitative or quantitative) help to reason about the system architectures, which are described in terms of the architectural components and properties that determine important quality requirements.

4. Analysis of the individual quality attributes with respect to the identified candidate architectures identified in step 3. Such analysis is important as it enable separation of concerns by allowing stakeholders with expertise on each system's quality attributes to perform independent quality attribute analysis. The result of this analysis is a statement about the system behaviour with respect to specific quality attributes, for example, "the average response time of the system is 30ms".

5. Identify Sensitivity points. These are points that significantly bring about variations in one or more quality attributes in the architecture. When such variations occur, the architecture model is updated to cater for the changes in design, and the updated model is evaluated.

6. Identify Trade-offs points. Trade-off points refer to architectural elements that are sensitive to or dependent on multiple quality attributes.

While the ATAM method evaluates the impact of alternative architecture on multiple and conflicting quality goals, it has some limitations: it does not handle uncertainty in model parameters; it does not emphasise detailed analysis of a system's measurable quality attributes, such as latency and response time, to be successful [148]; and the analysis technique of ATAM lacks tool support.

2.4.2 Cost Benefit Analysis Method— CBAM

The Cost Benefit Analysis Method (CBAM) [149, 179] was proposed in the early 2000s, at the Carnegie Mellon Software Engineering Institute. It is an economic modelling approach that extends ATAM to aid software architecture decision making in a multi-stakeholder context. CBAM models the cost and benefits of using alternative architectural strategies in achieving the system's quality goals (e.g. performance, availability and security) identified by the stakeholders during the ATAM process. Similar to ATAM, the CBAM makes use of different kinds of scenarios (e.g. usage scenarios, growth scenarios and exploratory scenarios [148]), and each scenario is associated with a quality attribute response goals. The response goals depict various ways by which an architecture responds to quality attribute stimuli. Response goals can be classified as *current*, *desired*, *best-case* or the *worst-case*. The level of utility (i.e. a measure of the benefit of a response for each scenario) associated to each response goal is described using utilty-response curve. Below is a summary of the CBAM steps:

1. Representative stakeholders collate the newly elicited scenarios and the ones obtained during the ATAM process .

2. Representative stakeholders define the quality attributes associated with each scenario.

3. The collated scenarios and quality attributes are prioritised using a voting system, where individual stakeholders distribute an allocated 100 points among the scenarios based on the desired response goal of each scenario. Typically, a weight of 1.0 is assigned to the scenario with the highest priority and other scenarios assume weights relative to the highest one. At the end of this step, some scenarios are shortlisted based on their weights.

4. For each shortlisted scenario, stakeholders assign utility score to each quality attribute response goals (e.g. current, desired, best-case and worst-case).

5. Next, stakeholders define a set of architectural strategies and their corresponding expected quality attribute response levels for each scenario. Each strategy has an associated cost.

6. For each scenario in each strategy, estimate the expected utility of the quality attributes response levels using interpolation.

7. For each strategy, evaluate the total benefit as the weighted sum of the difference between the expected and current utility score, for all scenarios. The weight values come from step 3.

8. For each strategy, compute the Return on Investment (ROI) as the ratio of the benefit to the cost and then rank the strategies according to their ROI.

9. Select a new architectural strategy with the best ROI and satisfies costs and schedule constraints.

10. Confirm the results with expert stakeholders and when necessary, repeat any of the steps that needs refinement and analysis.

The CBAM steps, described above, provide a transparent and structured software architecture decision-making process. Thus, the final architecture would most likely be agreed by the stakeholders and any resulting disagreement would be handled quickly. However, the CBAM decision objectives focus on the cost–benefit trade-offs of changing an architecture and may not be adequate to model stakeholders' objectives defined in terms of software quality attributes, such as response time and latency. In addition, the CBAM approach elicits uncertainty from the variations in the single point estimates supplied by the stakeholders. Such approach mixes up the stakeholders' disagreement about a parameter's probable value with uncertainty about a parameter's range of values. Finally, CBAM uses predefined equations and weighted sums of utility scores to compute the overall benefit of an architecture for each scenario. This ignores the possibility of having relationships among scenarios. More so, the assigned "scores" usually represent abstract (non verifiable) quantities that cannot be easily distinguished to mean either "importance" or "likelihood".

2.4.3 GuideArch

The GuideArch approach is a fuzzy logic-based framework that was proposed by Esfahani et al.[84] to explore architectural solution space to aid informed decisions in the presence of uncertainty. Using this approach, the authors simultaneously optimised the design of a Situation Awareness System (described in section 7.2.2) by considering seven system properties, such as development time, battery usage, response time, cost, ramp uptime and reliability. The GuideArch approach compares candidate architectures under uncertainty through the following four steps:

1. Identify a subset of candidate architectures that are valid according to some critical constraints.
2. Formulating the problem as a linear programming problem subject to architectural constraints, such as dependency constraint between architectures, conflict constraint between architectures and property constraints, such as the cost of an architecture cannot exceed a certain threshold. Then, find the optimal architecture by using fuzzy operators in comparing two architectures.
3. Rank the optimal architecture from the best to worst. This help domain experts incorporate domain knowledge, which may not be possible to model in the tool to select the final architecture for implementation.
4. Identify critical decisions that have big impact on the properties of ranked architectures and high level of uncertainty on the ranked architectures. Such decisions are given extra attention during the next iteration of decision making.

The drawbacks of the GuideArch approach are: the use of pre-defined fixed equations in assigning scores to candidate architecture; the use of fuzzy logic values that are not falsifiable or cannot be validated empirically, and the lack of techniques to analyse model uncertainties.

2.4.4 PRISM/Evochecker

PRISM [131, 156], a tool for modelling and analysing the dynamic behaviour of systems in diverse application domains, such as communication, energy and financial systems. PRISM also performs model checking of probabilistic models and enables the quantitative extensions of such models using the cost and rewards functions. The probabilistic models that PRISM supports are discrete-time Markov chains (DTMCs), continuous-time Markov chains (CTMCs), probabilistic automata (PAs), probabilistic timed automata (PTAs) and Markov decision processes (MDPs).

The PRISM models are expressed in a PRISM language, which is a state-based modelling language and based on the Reactive Module Formalism [13]. In the PRISM language, probabilistic models are represented as a set of modules, with each module state defined by some local variables that have a finite range of values, and the dynamic behaviour of a module can be described using probabilistic guarded commands as shown below:

$$[action]guard- > \sum_{i}^{n} expr_i : update_i \tag{2.1}$$

where $expr_i(1, 2, \ldots, n)$ is an arithmetic expression defined over all model variables, *guard* is a Boolean expression, which when evaluates to *true*, gives the probability (or rate) at which an $update_i$ change happens on a model variable, for a discrete-time model (or a continuous time model).

An extension of the PRISM modelling language is the Evochecker [101], which is a search based synthesiser of probabilistic models that satisfy design time quality of service requirements of software systems. Evochecker language extends the PRISM language with three constructs as follows:

1. *evolvable parameters* used to declare range of values for model parameter of type "int" and "double" in any command field except the *action* command. For example, the equation **const int** $x = 10$ is written as **evolve const int** $x = [9 \ldots 20]$.
2. *evolvable probability distributions* used to declare more than one element discrete probability distribution and the different probability ranges of the elements. For example, the equations **const double** x1 = 0.3; **const double** x2 = 0.7; **const double** y1 = 0.45; **const double** y2 = 0.55; will be written as **evolve distribution**$x[0.2 \ldots 0.4][0.8 \ldots 1.0]$; **evolve distribution** $y[0.4 \ldots 0.7][0.5 \ldots 0.8]$;
3. *evolvable modules* used to define at least two alternative modules. For example, **module** FraudDetector will be written as **evolve module** FraudDetector.

Evochecker [101] takes as input: (1) a probabilistic model template, which encodes alternative system architectural designs and their parameter ranges and (2) Quality of service requirements that specify both the optimisation objectives and constraints. For example, a constraint statement for the plastic card fraud detection system could be that " the investigation cost of all daily alerts must not exceed £5,000". The quality of service requirements are modelled using the *reward* probabilistic temporal logic formular [16, 61]. It then automatically analyses the quantitative probabilistic model using multi-objective genetics algorithms (such as NSGAII) to find the set of Pareto optimal probabilistic models that satisfy the quality of service requirements.

The PRISM and Evochecker models are best suited for modelling system's dynamic behaviours. They are not adequate for modelling stakeholders objectives and/or-refinement structure that the thesis proposed modelling language is well suited for. In addition, PRISM and Evochecker do not analyse model uncertainty.

2.4.5 SysML

SysML, which stands for Systems Modelling Language [96, 247] is an all-purpose systems engineering modelling language for analysing, specifying, designing and verifying a variety of complex engineering systems, namely: software, hardware, embedded, control and electro-mechanical systems. It also models other system entities, such as agents, procedures, data and facilities. SysML reuses existing Unified Modelling Language (UML) [254] diagrams, such as the behavioural diagrams (e.g. use cases, activity and sequence diagrams) and package diagrams. However, it provides additional modelling ability through the introduction of requirements diagrams— to model system requirements and the traceability to the design, and parametric diagrams— to model and analyse constraints on system properties, such as performance and availability, using constraint blocks.

In a SysML parametric model, a *constraint* is synonymous to a mathematical equation, e.g. the Newtonian equation $F = m * a$; a *constraint block* defines such equations in a manner that they are amenable to different modelling and quantitative analysis. A *constraint property* is a specific usage type of a constraint block in analysing a particular design. In a constraint, a *parameter* represents an equation variable, such as F, m and a in the Newtonian equation. A *value property* is a measurable quantity of a system or its components to be used for analysis, for example, the "mass" of an accelerating car or the "response time" of a London Ambulance System. Value properties of a constraint block are bound to value properties of other constraint blocks or other constraint parameters through *connectors*, represented by solid lines.

Quantitative modelling with parametric diagrams is based on the concepts of blocks and ports, which are used to model conceptual entities of systems, system hardware and software components, information or data that flow from one component to another. A *SysML block* is an extension of the UML structured class that represents a system at different hierarchy, from the top-level system to subsystems and physical or logical components or system environment. A *SysML port* allows access to the inner structure of a block. They can be categorised as standard and flow ports. Standard ports specify client-server communications using the *required* and *provided* interfaces for describing the required services by a block and the set of services a block provides, respectively. Flow ports are entry or exit points of a block. They describe the data or information allowed between a block and its immediate environments. Figure 2.3, extracted from [96], illustrates the concept of block and ports in details. The rounded rectangle represents constraint blocks and the small tiny boxes depict the ports and the solid line that links the blocks are the connectors.

SysML parametric models are executed using external engineering tools (e.g. mathematica, Abaqus and Ansys) to perform diverse model analysis, such as sensitivity analysis, trade-off analysis and design optimisation [96]. Sensitivity analysis in this context involves varying the *value properties* of the model to determine the ones that impact certain requirements greatly. Trade-off analysis involves the comparison of alternative designs using some set of metrics or system properties known as Measure of Effectiveness (MOE) [96] as shown in figure 2.3. In a trade-off analysis, the objective functions are described using SysML *constraint block* and their *parameters* are associated to the MOEs of interest with the aid of a parametric diagram . The analysis solutions are portrayed as specialised blocks, which are defined in a general block and have different instance values of the MOEs. Using the parametric diagram, the design optimisation defines an opitmisation function that models the net operational effectiveness of a system in terms of different MOEs. Such model visualises how the leaf-level parameter diagrams contribute to the MOEs at the top-level parameter diagrams as shown in figure 2.3.

SysML parametric modelling language and the modelling language proposed in the thesis are similar in many ways: They both use declarative equations for modelling system

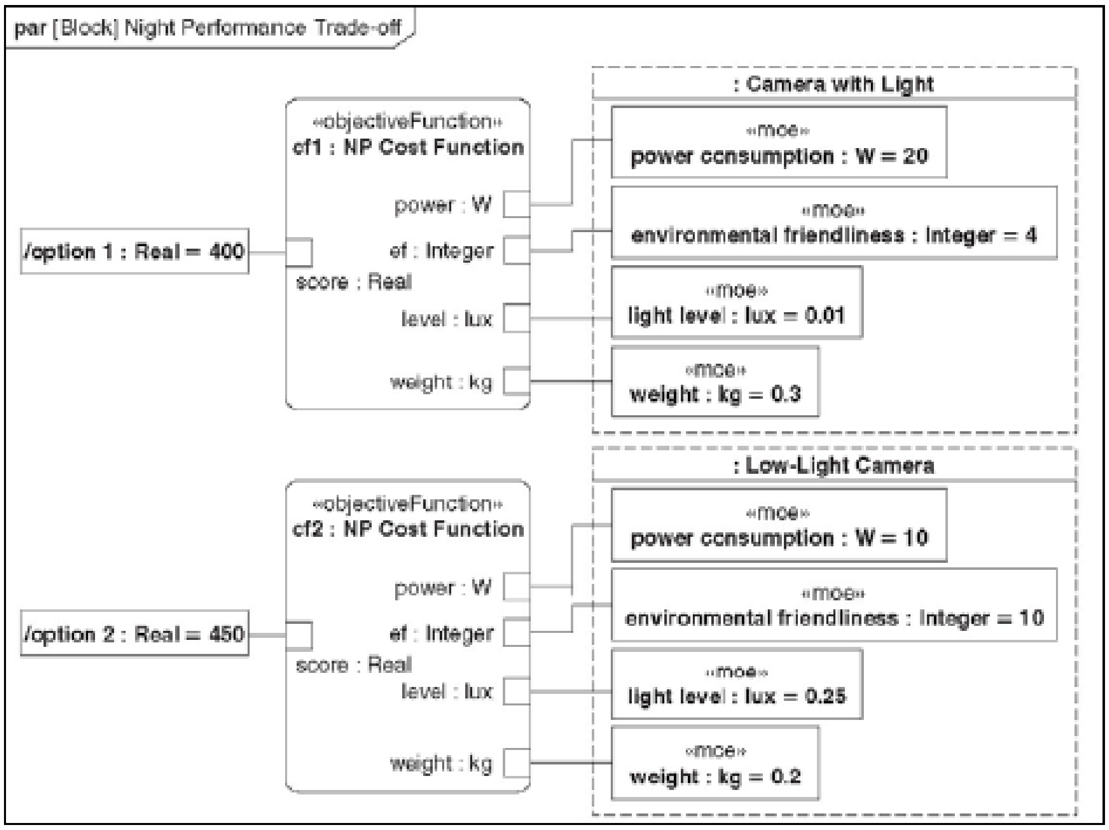
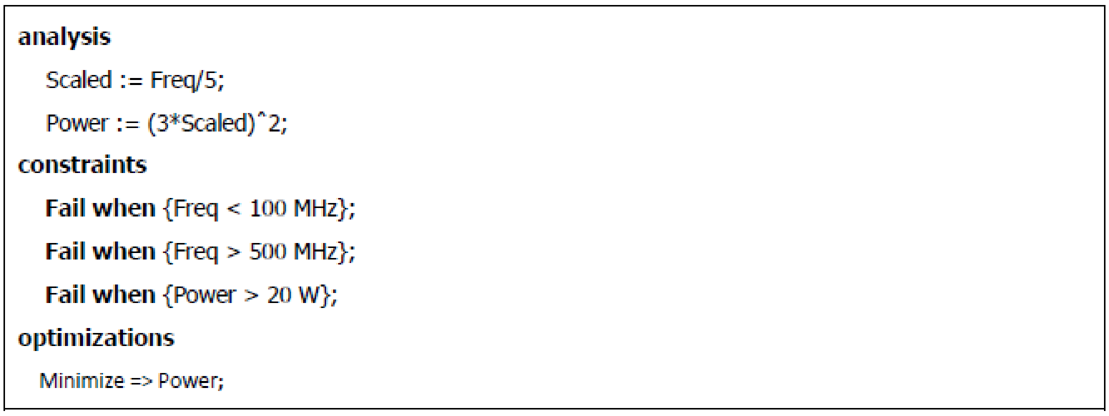


FIGURE 2.3: A parametric model for two design alternatives with the MOEs related to the parameters of an objective function [96].

quality attributes (e.g. performance and availability); allow trade-off analysis and design optimisation. However, since SysML parametric models are based on block diagrams, it suffers similar scalability and complexity issues like the quantitative goal modelling approaches described in section 2.2.3. SysML parametric modelling language would not be adequate to model the objectives and/or-refinement structure that the proposed thesis language is well suited for. Unlike the proposed modelling language which has an integrated tool support for performing decision analysis, SysML parametric modelling language requires external engineering analysis tool. Using such tools, however, involves transforming SysML parametric models to other models that are executable by the external tools. Such model transformation could lead to model synchronisation complexity and a risk of model inconsistencies. Finally, our modelling language and automated analysis tool analyses uncertainty using the concept of expected value of information, unlike SysML parametric models where uncertainty is not explicitly represented but can be modelled using constraint blocks and the engineering analysis tools (e.g. mathematica, Abaqus and Ansys) used by SysML parametric modelling language do not perform information value analysis.



```
analysis
   Scaled := Freq/5;
   Power := (3*Scaled)^2;
constraints
   Fail when {Freq < 100 MHz};
   Fail when {Freq > 500 MHz};
   Fail when {Power > 20 W};
optimizations
  Minimize => Power;
```

FIGURE 2.4: An example model in ACOL language [158].

2.4.6 ACOL

ACOL [158] is an annotation based modelling language for specifying architecture models using three types of expressions, namely: analysis, constraints and optimisation expressions. As shown in figure 2.4, the *analysis expression* allows a modeller to analyse custom and derived non-functional properties of interest. The *constraint expression*, which follows the analysis expression allows the specification of bound(s) for the non-functional properties' values. The optimisation expression maximises or minimises one or more non-functional properties.

The drawbacks of ACOL is that the tool requires external analysis tools to perform its optimisation analysis; it does not analyse the uncertainty in model parameters, and is not adequate for modelling the stakeholders' objective and/or-refinement structure that the proposed thesis modelling language is well suited for.

2.4.7 POISED

POISED, POssIbilistic SElfaDaptation, is an approach introduced by Esfahani et al. [83] to deal with uncertainty in architecture decisions at run-time in self-adaptive systems (SAS) –systems that adapt to changes to requirements or environement conditions in order to meet certain goals. POISED supports SAS in making optimal adaptation decisions when there is uncertainty in the impact of such decisions on a system non-functional goals, such as response time, power consumption etc.

Esfahani et al. [83] formally defined Self-Adaption decision problem as:

1. A system consists of a set of components with different configuration decisions that have alternative options that are mutually exclusive.
2. A set of constraints that define a valid architecture configuration.
3. A set of quantifiable system quality attributes of interests. The quality attributes are observed through different configuration combinations during system execution.
4. A set of available resources accessible to the system during execution.
5. A set of user preferences to capture stakeholders' satisfaction about the changes in the quality attributes.

The optimisation problem involves maximising the weighted sum of the overall stakeholder satisfaction of the quality attributes subject to configuration and resource constraints. POISED approached this problem using Linear Programming optimisation techniques augmented with fuzzy set theory to represent and estimate the positive and negative consequences of uncertainty on alternative adaptation decision choices of a system.

The thesis proposed approach is similar to POISED in some ways: both approaches emphasise and capture uncertainty explicitly and they capture constraints between options of decisions. However, both approaches differ in uncertainty representation: POISED uses Possibility theory based on fuzzy set defined over fuzzy variables, this thesis uses Bayesian probability defined over random variables and further analyses uncertainty through information value analysis. Also, the optimisation model developed in POISED uses weighted sum of utility functions while our approach supports elaboration of domain specific decision models. Finally, unlike our approach, POISED does not have an integrated tool support for automated decision analysis as it uses an online NEOS sever for solving numerical optimisation problems [62].

2.4.8 Multi-Objective Decision Analyser— MODA

The Multi-Objective Decision Analyser (MODA) [166] is Bayesian decision analysis approach in software engineering, which was proposed to aid software architects in dealing with uncertainty about the impact of alternatives on stakeholders' goals. MODA employs statistical decision analysis technique and Pareto-based multi-objective optimisation to early requirements and architecture design decisions. The MODA approach formalises a requirements and architecture decision problem in terms of domain-specific measurable goals; elicits and presents uncertainty as probability distributions; uses Monte-Carlo (MC) simulations to simulate the impact of alternatives on goals; and shortlists the Pareto set. In quantifying and reducing uncertainty, MODA uses a powerful decision analysis concept known as expected value of information, which is the expected benefit accrued due to having additional information about a model parameter.

With respect to previous requirements and software architecture decision methods such as NRP, EVOLVE, IFM, ATAM, CBAM, POISED and GuideArch, the MODA method addresses some of their limitations which include: the use of unprincipled methods to elicit uncertainty; the evaluation of alternatives using criteria that are not falsifiable, or cannot be validated empirically; the lack of provision for information about the risks that could result from uncertainty; the lack of support for determining the degree to which the risks can be reduced by obtaining additional information about model parameters.

The authors of MODA noted the complexity of developing and validating sound requirements and software architecture decision models that capture the real stakeholders' goals. The complexity of developing such models comes from the fact that real world models typically have: (i) large number of model parameters and are composed of submodels of the software systems, which measure the impact of alternative design decisions on software quality attributes, such as security and availability; (ii) sub-models of the application domain, which measure the impact of the software and alternative design decisions on the stakeholder goals, such as reducing financial loss due to fraud in a fraud detection system. The MODA approach deals with the complexity described through the following steps [166]:

1. Software architect define the architecture decision model.
2. Software architect define the cost-benefit decision model.
3. Software architect define the decision risks.
4. Software architect elicit domain and system parameters' values from stakeholders
.

5. Next is the shortlist of candidate architectures among alternative architectures.

6. Software architect identify closed and open design decisions. Closed decisions are said to be made when all the shortlisted architectural solutions agree on the options to be selected for those decisions; otherwise, we say the decision is open.

7. Next is the computation of the expected value of having additional information of a model parameter.

This thesis builds on the statistical decision analysis technique used in MODA. However, the thesis approach extends MODA by presenting a modelling language that helps software architects write requirements and architecture decision models that are sound and falsifiable. The automated decision analysis technique presented in this thesis hides manual implementation details about evaluating the impact of alternatives on stakeholder goals and estimating the financial value of reducing uncertainty in model parameters. This enables software architects focus on the conceptual decision modelling of the problem. In addition, unlike MODA, the thesis approach (i) generates AND/OR refinement graphs and decision graphs from the model's equations (see chapter 4.2 and 4.4). Such graphs can enhance model communication with stakeholders [264]; (ii) minimises the optimisation problem's search space by inferring the set of minimal and complete solutions associated to a decision model (see chapter 4.3).

2.4.9 Other Search-Based Architecture Decision Methods

The literature is abound with many search based approaches to model and analyse software architecture decisions from the perspective of run time and design time of a system. A comprehensive survey on software architecture decisions exist in [117, 121, 122, 232]. But this section focuses only on the approaches applied in the context of multi-objective decision problems in the early phase of architecture design of software systems. These approaches, however, either lack tool support for automated analysis of decision models, or those that provide tool support do not support elaboration of domain specific decision models, but rather rely on pre-defined models and objective equations, which often do not capture the actual stakeholders objectives and are not valid models of the impacts of decisions on objectives.

In the area of Quality of Service, Web services and component selection and allocation; Fredriksson et al. [95] presented an approach for optimally allocating components to real time tasks while considering trade-offs like CPU-overhead and resource usage. In their work, they developed a model for tasks and components allocations, deriving memory consumption and CPU overhead for task deployment. Their objective was to minimise resource usage (e.g CPU time and memory) using scheduling and optimisation techniques and incorporate real time analysis to achieve feasible allocation.

Liu Zhuang et al. [286] proposed a Muti-Criteria Decision Making (MCDM) approach to enable users (or experts) to search Pareto Optimal Design Alternatives (PODA). Amongst the PODA, they found the best one by incorporating corresponding weighted fuzzy preferences in the global plan of web services selection based on QoS criteria such as price, response time, availability, reliability and reputation.

Wang et al. [268] proposed multi-objective genetic algorithm (MOGA) to find a set of Pareto optimal solutions that optimise three objectives such as minimise cost, maximise reliability and minimise reponse time, while still satisfying the functional requirements.

With respect to research done in the area of software quality; Khoshgoftaar et al. [152] proposed a Module Order Model using Genetic programming to predict the relative quality of each software module, particularly the most faulty ones. They simultaneously optimise four performance objectives and evaluated their approach using two real-world software systems.

Grunske et al. [109] applied an evolutionary approach in architecture refactorings, to select good design alternatives within reasonable time for a satellite control system that is based on Bi-spectral InfraRed Detector [39]. They optimised the system architecture such that the reliability of the satellite is maximised and cost is minimised, subject to a particular weight constraint to limit the number of redundant components.

Harman et al. [120] extended and improved previous work on search based re-factoring approaches that produce single sequence of re-factorings through combined complex metrics; their approach, however, finds multiple Pareto optimal sequence of refactorings that takes into account the availability of resources and giving users the opportunity to specify the level at which to re-factor.

Simons et al. [239] introduced an interactive search based approach to support a human designer at the start and during conceptual design of a software system before the actual implementation of the design is transformed to source code. They optimised the design of a Cinema booking system by maximising cohesion and minimising coupling between class objects using UML class diagram as representations.

Raiha et al. [206] proposed a multi-objective genetic algorithm to find the optimal trade-off solutions to the design of an electronic home system (that controls devices and provides interfaces to enable users manage their home) given two quality attributes such as modifiability and efficiency.

Al-Naeem et al. [8] presented a quality driven approach called ArchDesigner that obtains the optimal trade-off between conflicting stakeholders‘ goals, project constraints (cost and time) and competing architectural concerns. In their work, they maximised the accumulative value score subject to constraints that specified cost and time are not exceeded and only one alternative is selected.

Meedeniya et al. [174] presented a simulation-based method that handles parameter range estimates using probability distributions. The authors used Monte Carlo simulation in the estimation of a software architecture reliability through the combination of the reliability of its component elements.

Noppen et al. [188] presented a design tree approach that scans a design space for design decisions, their sequences and all the alternatives considered in the presence of imperfect information about estimates and requirements.

Cortellessa et al. [60] proposed an optimisation framework called CODER for a component based selection and optimisation procedure based on cost minimization of the proposed system while ensuring a certain level of satisfaction of the system reliability and delivery time.

To summarise, Section 2.4 reviewed some related work to this thesis on software architecture decisions which impact stakeholders’ business and non-functional (quality) goals of a system. We described the state-of-the art software architecture decision methods such as Architecture Trade-off Analysis Method (ATAM) [148], Cost Benefit Analysis Method (CBAM) [149, 179] , GuideArch [84], PRISM [131, 156], Evochecker [101], Systems Modelling Language (SysML) [96, 247], POISED [83], MODA [166] and other search-based software architecture decision methods [109, 117, 121, 122, 152, 174, 232]. These approaches require higher modelling effort and generally have limited automated tool support for decision analysis. Using these approaches involves manual encoding of decision models in widely used statistical programming languages, such as R and MATLAB. This results in modellers spending time on implementation details rather than concentrating on the conceptual decision problem.

2.5 Product Configuration Decisions in Software Product Line Engineering

The proposed thesis modelling language is aimed at enabling software architects in modelling requirements and architectural decision problems characterised by (i) single option selection similar to mutually exclusive option selection (XOR-nodes) of feature diagrams used in Software Product Line Engineering (SPLE); (ii) multiple options selection similar to non-mutually exclusive options selections (OR-nodes) of feature diagrams; and (iii) constraints dependency relationships, e.g., excludes and requires constraints used in SPLE. Since RADAR's modelling language extends SPLE feature models, this section reviews some of the related work to this thesis on product configuration decisions in SPLE.

Software Product Line Engineering (SPLE) is the engineering of a portfolio of similar products with variations in some features and functions [203]. In SPLE, products can be software, or a system that have software running within it, and can be similar with respect to requirements, design specification, test cases, project schedules and project budget. Products are described by *features*. The similarities and variations in products are depicted using *feature models*.

A feature model defines constraints between features and specifies which combination of features defines valid products. For example, Fig. 2.5 is an example of an attributed feature model of GPS software. The nodes in the model represent features and arrows depict relationships between features. These relationships can be a (i) *mandatory parent child relationship*, in which a child must be included in all the products where the parent feature appears, e.g., all products configuration possible in the GPS example must have Routing feature; (ii) *optional parent child relationship* where a child may be included in

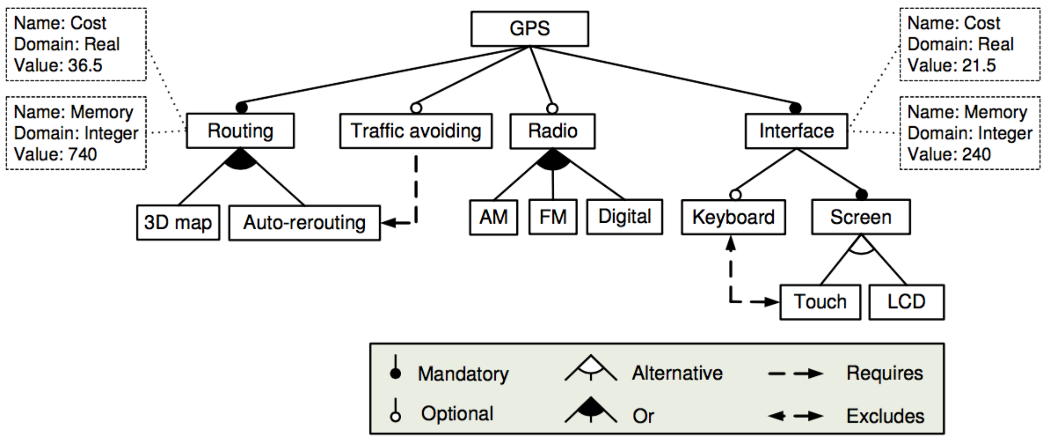


FIGURE 2.5: An example attributed feature model for GPS software [130].

products that have its parent feature. The Keyboard feature in the GPS feature model example is an optional feature of the user Interface of GPS products; (iii) *alternative relationship* where only one feature among a list of features should be selected when its parent feature is selected in a product. In Fig. 2.5, a GPS product can have either a Touch screen feature or an LCD screen feature but not both; *or-relationship* states that at least one child feature can be selected in the products where the parent feature appears. e.g., a GPS product can have the 3D map viewing feature, Auto-rerouting feature, or both of them.

There are also constraint relationships between features: (i) a feature f_1 *requires* a feature f_2, if when f_1 is in a product, then f_2 must be included in this product; (ii) a feature f_1 *excludes* a feature f_2, means that features f_1 and f_2 cannot be in the same product.

Many tools have been proposed for reasoning [153, 175, 253] and configuring [17, 52, 145, 190] systems based on feature models. These tools allow modelling different relationships (XOR and OR) and constraints (exclude and requires) between features. However, they have no means to explicitly capture and analyse model uncertainty.

Another line of research in Software Product Lines is optimal product selection based on feature models and stakeholders' preferences such as minimising cost, minimising feature defect counts. Initial approaches to tackling optimal product selection used search-based single-objective optimisation [110, 271, 272]. But these techniques could lead to a bias in the search process, as the optimisation of one objective may be at the expense of the other. To address this problem, several state of the art Evolutionary Multi Objective Algorithms (EMOAs) have been explored [235], with further enhancements to increase the number of valid solutions obtained from the EMOAs. These include seeding the search with valid products [233], augmenting the search with SAT solvers and smart mutation and crossover implementation [128], and introducing a novel solution encoding which removes features that appear in all valid products and does not add features whose children are not included in a product [130].

The main limitation with existing SPLE approaches for product configuration decisions is that they generally use weighted sums of feature attribute (cost, defect count, frequency of use, peformance) values. These approaches do not handle Boolean expressions and do not allow the elaboration of domain specific decision models that RADAR supports through AND/OR refinements. In addition, existing approaches generally lack automated technique for analysing uncertainty and informing decision makers about the financial value of reducing uncertainty in a decision model.

2.6 Summary

This chapter presented a background and state of the art requirements and architecture decisions. The chapter first gave a general background on multi-objective optimisation since requirements engineering and software architecture decisions are generally multi-objective optimisation problems. This is followed by a presentation of the state-of-the-art approaches for modelling and analysing requirements engineering and software architecture decisions. Table 2.1 summarises the these approaches. The main drawbacks of these approaches is that they are limited by the difficulty in elaborating problem specific decision models and/or lack integrated tool support for automated decision analysis under uncertainty.

The objective of this thesis is to mitigate the limitations of existing requirements and architecture decision approaches. The thesis achieves this by introducing a new modelling language and automated decision analysis technique implemented in a tool called RADAR —Requirements and Architecture Decision AnalyseR. We evaluate the tool on a number of real-world systems (see Table 7.1) by answering the following research questions:

- **RQ1 (Applicability):** Is the RADAR tool applicable to real-world requirements and architectural decision problems (chapter 7.3.1)?
- **RQ2 (Usefulness):** Does RADAR's decision analysis technique provide useful improvements to real-world requirements and architecture decisions (chapter 7.3.2)?
- **RQ3 (Scalability):** What is the scalability of RADAR's exhaustive simulation and optimisation approach (chapter 8)?
- **RQ4 (Performance Analysis):** What is the performance of RADAR's alternative search-based evolutionary algorithms (chapter 8)?

Existing Approach	Use of Generic Fixed Non-falsifiable Equations	Decision Types (Single, (Multiple Option Selection)	Handle Constraints in Decisions	Forms of Uncertainty	Parameter Estimations	Uncertainty Analysis	Provides Language for model elaboration	Automated Decision Analysis Tool	Optimisation Approach	Design time or runtime analysis
Cost-Value approach, Karlsson et al. [143]	Yes	Multiple option selection	NA	NA	point estimates	NA	No	No	Cost-value plot	design time
Cost-Value approach, Veerappa et al. [267]	Yes	Multiple option selection	NA	NA	point estimates	NA	No	No	Approximate optimisation	design time
NRP, Bagnall et al. [30]	Yes	Multiple option selection	NA	epistemic	point estimates	NA	No	No	Approximate	design time
EVOLVE/EVOLVE* Ruhe et al. [216]	Yes	Multiple option selection	Yes	epistemic	point estimates	NA	No	Yes	Approximate	design time
NRP Albourae et al. [9]	Yes	Multiple option selection	No	NA	point estimates	NA	No	Yes	Greedy	design time
NRP Baker et al. [31]	Yes	Multiple option selection	No	NA	point estimates	NA	No	Yes	Greedy/Simulated Annealing	design time
NRP Van den Akker et al. [255]	Yes	Multiple option selection	Yes	NA	point estimates	NA	No	Yes	Exact	design time
NRP, Zhang et al. [282, 283]	Yes	Multiple option selection	No	NA	point estimates	NA	No	No	Approximate	design time
NRP, Zhang et al. [285]	Yes	Multiple option selection	Yes	NA	point estimates	NA	No	No	Approximate	design time
NRP, Finkelstein et al. [91, 92]	Yes	Multiple option selection	No	NA	point estimates	NA	No	No	Approximate	design time
NRP, Harman et al. [123]	Yes	Multiple option selection	NA	epistemic	point estimate	sensitivity analysis	No	No	Exact	design time
NRP, Paixao et al. [196]	Yes	Multiple option selection	No	epistemic	point estimates	NA	No	No	Approximate optimisation	design time
NRP, Li et al. [167]	Yes	Multiple option selection	No	epistemic	value ranges	NA	No	No	Approximate optimisation	design time
NRP, Li et al. [168]	Yes	Multiple option selection	Yes	epistemic	value ranges	robust optimisation	No	No	Exact	design time
IFM, Denne et al. [70, 71]	Yes	Multiple option selection	Yes	NA	point estimates	NA	No	No	Greedy	design time
IFM, Alencar et al. [10]	Yes	Multiple option selection	Yes	NA	point estimates	NA	No	No	Statistical approximation	design time
IFM, Cantor et al. [45]	Yes	Multiple option selection	Yes	epistemic	value ranges	NA	No	No	approximate optimisation	design time
IFM, Eduardo et al. [63]	Yes	Multiple option selection	Yes	NA	point estimates	NA	No	No	Game theory	design time
IFM, Oni et al. [195]	Yes	Multiple option selection	Yes	epistemic	Bayesian distribution	robust optimisation	No	No	Exact	design time
Quantitative NFR, Affleck et al.[3, 4]	No	Single option selection	NA	epistemic	point estimates	sensitivity analysis	No	No	Exact	design time
Quantitative NFR, Pasquale et al. [199]	No	Single option selection	NA	NA	point estimates	NA	No	No	Exact	design time
Quantitative NFR, Wei et al. [270]	No	Single option selection	NA	epistemic	point estimates	NA	No	No	Exact	design time
Quantitative KAOS, Letier et al. [165]	No	Single option selection	NA	epistemic	probability distribution	NA	NA	No	Analytical methods	design time
Quantitative KAOS, Letier et al. [126]	No	Single option selection	NA	epistemic	probability distribution	NA	NA	No	Approximate optimisation	design time
Quantitative KAOS, Sabetzadeh et al. [222]	No	Single option selection	NA	epistemic	probability distribution	sensitivity analysis	No	No	NA	design time
ATAM, Kazman et al. [147, 148]	Yes	Single option selection	NA	epistemic	point estimates	NA	No	No	NA	design time
CBAM, Kazman et al. [149, 179]	Yes	Single option selection	NA	epistemic	point estimates	NA	No	No	NA	design time
GuideArch, Esfahani et al. [84]	Yes	Single option selection	Yes	epistemic	fuzzy values	NA	No	Yes	Exact	design time
POISED, Esfahani et al. [83]	Yes	Single option selection	Yes	alaetory	fuzzy values	NA	No	No	Exact	design time
PRISM, Hinton et al. [131, 156]	No	Multiple option selection	NA	alaetory	value ranges	NA	Yes	No	Exact	design time
Evochecker, Gerasimou et al. [101]	No	Multiple option selection	Yes	alaetory	value ranges	NA	Yes	Yes	Exact	design time
SysML, Friedenthal et al. [96, 247]	No	Single option selection	No	epistemic	point estimates	sensitivity analysis	Yes	No	NA	design time
ACOL, Langsweirdt et al. [158]	No	Single option selection	Yes	NA	point estimates	NA	Yes	No	Exact	design time
ArchDesigner, Al-Naeem et al. [8]	Yes	Multiple option selection	No	NA	point estimates	NA	No	No	Approximate	design time
MODA, Letier et al. [166]	No	Single option selection	Yes	epistemic	Bayesian probability	expected value of information	No	No	Exact	design time
SPL SPLOT, Mendonca et al. [175]	Yes	Single and multiple option selection	Yes	NA	point estimates	NA	Yes	No	NA	design time
SPL FeatureIDE, Kastner et al. [145]	Yes	Single and multiple option selection	Yes	NA	point estimates	NA	Yes	No	NA	design time
SPL Clafer, Antkiewicz et al. [18]	Yes	Single and multiple option selection	Yes	NA	point estimates	NA	Yes	No	NA	design time
SPL ClaferMOO, Olaechea et al. [192]	Yes	Single and multiple option selection	Yes	NA	point estimates	NA	Yes	No	Exact	design time

TABLE 2.1: Summary of existing requirements and architecture decision analysis approaches. These approaches are compared based on whether they use generic fixed non-falsifiable model equations; the type of decisions , i.e., single and multiple option selection; whether the approach handles any form of constraints between decisions; parameters estimations (point values, value ranges); approaches to handling uncertainty (e.g. sensitivity analysis, robust optimisation, expected value of information); whether an approach provides a modelling language for model elaboration using simple mathematical equations; an approach provides tool support for automated decision analysis; optimisation technique (e.g., exact and approximate optimisation techniques); whether the approach is applicable at design time or runtime of the system.

Chapter 3

RADAR: A Guided Tour

3.1 Introduction

Many software requirements and architecture decisions have to deal with multiple conflicting objectives. For example, deciding which set of requirements to implement in the next release of a product [30] or deciding among alternative system designs in a goal model [126], such that we minimise cost, maximise value and minimise risks. In both of these decisions, decision-makers are often confronted with high levels of uncertainty and a huge space of alternatives to choose from [166].

Making the right requirements and software architectural decisions is critical to the successful delivery of software intensive systems [265]. Poor decisions, however, may lead to delay in project delivery, huge financial loss, and stakeholders' dissatisfaction. This justifies the need for automated techniques that aid decision-makers in evaluating the impact of decision choices on stakeholders' objectives and in selecting the one that produces the best trade-off between their objectives. Such techniques are only possible and valid in the presence of detailed decision models that are amenable to decision analysis and capture both the proposed software system and the application domain of interest.

Quantitative decision models allow requirements engineers and software architects to analyse requirements and architecture decisions using quantitative techniques, such as stochastic simulation and multi-objective optimisation, but the difficulty of elaborating the decision models is an obstacle to the wider adoption of such techniques. Many requirements and architecture decision support methods, such as EVOLVE [217], CBAM [149, 179] , GuideArch [84], avoid this problem by relying on pre-defined model equations, which fail to model the real stakeholders' objectives. Other techniques that use quantitative goal-oriented decision models [3, 4, 103, 126, 165, 222] require decision-makers to manually encode the decision models in general programming language, such as R and MATLAB. This impairs model readability and forces modellers to consider implementation concerns instead of focusing solely on the conceptual decision problem.

To address these limitations, this thesis introduces RADAR, a novel modelling language and analysis tool, intended to facilitate requirements and architecture decision analysis. The language has relations to quantitative AND/OR goal models described in Section 2.2, and to feature models used in software product lines described in Section 2.5. However, RADAR simplifies such models to a minimum set of language constructs essential for decision analysis. This chapter presents a guided tour on how to make requirements and architecture decisions with RADAR. The purposes are to give a high-level overview of the modelling language and to provide background on the decision analysis method used [166]. Formal descriptions of the modelling language and analysis technique will be given in Chapter 4 and 5, respectively.

3.2 Running Example

We illustrate RADAR's modelling language and decision analysis technique using a public bike sharing system used in many urban cities, such as Beijing, London and New York. This example is a case study of an European project (QUANTICOL) used in the design and quantitative analysis of Collective Adaptive Systems (CAS) [177, 250].

The main goal of a bike sharing system is to increase commuters' transit options, minimise energy consumption, enhance the quality of life by reducing air and noise pollution, and lessen traffic congestion within the city. The operation of the bike sharing system at a high-level is thus: a city has bike docking stations usually within walking distances from the train and tram stations. A registered user rents a bike from a docking station and returns the bike, after using it, to any docking station.

The bike sharing system consists of different components that pose high level decision choices. These decisions are described below:

- The *bikes security strategy* that can use an optional localisation feature or an optional anti-theft feature that requires a GPS device.
- If the localisation security strategy is selected, the bikes can be equipped with a *tracking mechanism* that uses a GPS or RFID or both.
- The bikes' *manufacturer brand* that can be any one of "Cortina Cyclese", "Derby Cycle", "Catrike", "Bianchi-Bike" or "A-Bike", with A-Bike not able to support the localisation feature.
- The bikes *docking station* whose capacity is either permanently fixed, temporarily fixed, or flexible.
- The *system registration management* in which users can register through a website, at the dock stations, or in a kiosk located around the dock stations.
- If the kiosk registration option is selected, the *kiosk registration method* can be equipped with optional features, such as a touch screen option to improve user experience, a keycard reader option to check out usage statistics, a credit card option for payment, and a keycard dispenser to disburse cards for short-term passes.
- The *system access management* that can have an option to use a smart card technology, or a smart phone technology, or a keycard technology which requires a key card reader and must be implemented together with the keycard dispenser.
- The *non-mandatory system components* that include a bike maintenance program option for small-scale repairs such as flat tyres; a bike redistribution mechanism option for timely distribution of bikes to dock stations; a real time system status option for providing updates about the availability of bikes.
- If the bike redistribution mechanism is implemented, a *reward program* for users to encourage returning of bikes to inconvenient locations, such as locations with higher or lower elevations.

- If the system is equipped with real time information update, the *system status method* in which real time system update is provided through a website, or a mobile application that uses the mapping and localisation features of smart phones.

Suppose, a decision-maker wants to decide on the best combination of the different component options (features) to deploy to maximise the overall system's benefit at minimal cost. The decision making process is beyond human capabilities: it is compounded by the stakeholders' conflicting concerns; uncertainty about domain quantities (e.g. number of bicycles to deploy); uncertainty about the impact of different decision choices on the stakeholders' concerns; the different complexity of decisions which include: single option selection (e.g. in the bike manufacturer brand and dock station capacity); multiple option selections (e.g. between the bike anti-theft feature and the GPS option of the bike localisation strategy); and constraints relationship between options of decisions (e.g. A-Bike does not support the localisation feature).

3.3 Making Decisions with RADAR

The following are the steps involved when making decision with RADAR:

Step 1. Model the requirements and architecture decision problem. This involves identifying the decisions to be made; defining stakeholders objectives of interest; and modelling the impact of the decisions on the objectives. The result of this step is a decision model.

Step 2. Analyse the decision model. This involves i) analysing model uncertainty and evaluating the impact of decisions on the objectives using Monte-Carlo Simulation; ii) shortlisting the optimal decisions using multi-objective optimisation techniques; iii) evaluating and quantifying the financial value of additional information or model refinement to reduce uncertainty about model parameters before making a decision. Depending on the outcome of the analysis, stakeholders can either make a decision or perform step 3.

Step 3. Get additional information or perform additional analysis through requirements elicitation, prototyping and modelling. Then update the decision model and this will trigger a new modelling and analysis cycle. This cycle stops once the financial value of additional information is not sufficient to justify the effort required.

3.4 Developing Decision Models

Before we formally define RADAR in detail in Chapter 4, we first illustrate the application of RADAR on two examples: first is a simple, complete but illustrative example about refactoring the architecture of a system. The second example is the design of a public bike sharing system previously described in Section 3.2.

3.4.1 Example 1: Software Architecture Refactoring

Imagine having to perform a cost-benefit analysis for deciding whether to re-factor the architecture of an existing application. With the current architecture, the application generates relatively predictable benefits. Refactoring creates the possibility of generating much higher benefits, but the refactoring costs and benefits are highly uncertain.

A RADAR model for this decision problem might look like this:

```
1   Model Refactoring;
2   Objective Max ENB = EV(NB);
3   Objective Min LP = Pr(NB < 0);
4   NB = Benefit − Cost;
5   Cost = decision("Architecture choice"){
6           "As-is" : deterministic(0);
7           "Refactoring" : normalCI(1, 5);
8   }
9   Benefit = decision("Architecture choice"){
10          "As-is" : normalCI(0.9, 1.1);
11          "Refactoring" : normalCI(1, 9);
12   }
```

The language keywords are in bold. The first line declares the name of the model problem. The next two lines define the optimisation objectives: maximising expected net benefit (ENB) and minimising loss probability (LP). The function **EV** denotes the expected value (or mean) of a random variable and **Pr** denotes the probability of a Boolean expression. The model's fourth line then defines net benefit (NB) as the difference between benefit and cost.

The next four lines state that cost depends on the architectural choice. The **decision** keyword expresses alternative options at decision points in which only a single option can be selected. If the choice is to keep the as-is architecture, we assume the cost to be zero. The **deterministic** keyword means we believe this cost to be certain. If the choice is to refactor, we believe the cost has a 90% chance of being between £1m and £5m. The expression **normalCI**(1, 5) means the cost follows a normal distribution whose 90% confidence interval is between 1 and 5. Similarly, the last four lines state that benefit depends on the architecture choice and records our beliefs about the benefit's likelihood for the as-is and refactored architecture.

Probabilities in our approach are Bayesian; probability distributions denote the decision makers' beliefs about the likelihood of uncertain quantities and events. These beliefs can be informed by subjective judgements, objective data, or a combination of both. Bayesian methods typically start with probability distributions informed by subjective judgements alone, then update the distributions (using Bayes rule) as new data and information becomes available [275].

Reliable methods exist for eliciting a person's beliefs about uncertain quantities or events, and model these beliefs as probability distributions [191]. A recommended simple approach consists in eliciting 90% confidence interval as used above [136]. For these elicitation methods to be reliable, people providing estimations have to be 'calibrated' on a set of estimation exercises intended to mitigate their under- or over-confidence biases.

3.4.2 Example 2: Design of a Bike Sharing System

Below is a partial decision model developed for the bike sharing system example using RADAR's modelling language. This model consists of decisions with single and multiple option selections. It also includes constraints relationships between options of decisions. The remaining model is presented in Chapter 7.2.5.

```
1   Model BSS;
2   Objective Max ExpectedNetBenefit = EV(NB);
3   Objective Min LossProbability = Pr(NB < 0);
4   NB = Benefit − Cost;
  // Lines omitted represent equations for system's Benefit
60   Cost = CostOfBikes
              + CostOfSecuringBicycles
              + CostOfDockStations
              + CostOfSystemAccessMgt
              + CostSystemRegistrationMgt
              + CostOfOtherComponents;
66   CostOfBikes = (NbrBikesToDeploy - NbrBikesDeployed) * UnitCost;
67   NbrBicyclesToDeploy = triangular(500, 550, 600);
68   NbrOfBicyclesCurrentlyDeployed = deterministic(500);
69   UnitCost = decision (“Bike Manufacturer Brand”){
70              “A-Bike”: normalCI(80 ,100);
71              “Bianchi” : normalCI(200 ,300);
72              “Cortina Cyclese”: normalCI(100 ,150);
73              “Derby Cycle”: normalCI(140, 200);
74              “Catrike” : normalCI(250 ,350);
  }
75   CostOfSecuringBikes=decision-subset(+)(“Bikes Security”){
76              “Anti-theft feature” : normalCI(1,5);
77              “Localisation feature” : CostOfLocalisation;
78   }
79   CostOfLocalisation=decision-subset(+)(“Tracking Mechanism”){
80              “GPS feature” : normalCI(5,10);
81              “RFID feature” : normalCI(2, 7);
82   }
```

```
CostOfDockStations = decision (“Dock Station”){
        “Permanently Fixed” : triangular(25,30, 35);
        “Temporarily Fixed ” : triangular(30,35,40);
        “Flexible ” : triangular(30,40,50);
}
CostOfSysAccessMgt = decision-subset(+)(”Access Management”){
        “Smart card” : triangular(30,35, 40);
        “Smart Phone” : triangular(25, 30,35);
        “Key Card” : triangular(35, 40,45);
}
CostSysRegMgt = decision-subset(+)(“Registration Management”){
        “Kisok Reg” : CostOfKioskReg;
        “Dock Station Reg” : triangular(28, 30,32);
        “Web Reg” : triangular(30, 40,50);
}
CostOfKioskReg = decision-subset(+)(”Kisok Registration”){
        “Touch Screen” : triangular(10, 15, 20);
         “Key card reader” : triangular(15, 20,25);
         “Credit Card” : triangular(20, 22,25);
         “Card Dispenser” : triangular(20, 25, 30);
 }
 CostOfOtherComponents =
        decision-subset(+)(“Non Mandatory Component”){
                “System Status Info” : CostOfStatusInfo;
                “Bike Maintenance” : triangular(8,10, 12);
                “Bike Redistribution” : CostOfRedistribution;
        }
 CostOfStatusInfo = decision-subset(+)(”System Status”){
         “Real Time Web Info” : triangular(35,40, 55);
         “Real Time Mobile App Info” : triangular(50, 80, 100);
 }
```

```
114  CostOfRedistribution =
115          RedistributionCostWithoutReward + CostForRewardingUsers;
116  RedistributionCostWithoutReward = normalCI(8, 10);
117  CostForRewardingUsers = decision("Reward Users"){
118          "Without reward" : deterministic(0);
119          "With Reward" : deterministic(10);
120  }
121  Constraint  "Bike Manufacturer Brand" : "A-Bike" excludes
122           "Tracking Mechanism ": "GPS feature";
123  Constraint  "Kiosk Registration" : "Card Dispenser" couples
124                "System Access Management": "Key Card";
125  Constraint  "System Access Management" : "Key Card" requires
126                "Kiosk Registration" : "Key card reader";
```

In the bike sharing model described above, lines 2 and 3 declare the model objectives: the first objective is a maximisation of the expected net benefit of the system (ExpectedNetBenefit) and the second objective is a minimisation of the loss probability (LossProbability).

Line 4 states that the net benefit is benefit minus cost. Lines 60 to 65 state that the total cost is the sum of the cost of different system components. Lines 66 states that the cost of bikes is the product of the unit cost of a bike and the difference between the number of bikes to deploy and the number of bikes already deployed.

In line 67, the expression **triangular**(500, 550, 600) means that the decision-maker believes the number of additional bikes to deploy (NbrBikesToDeploy) is between 500 and 600, with a likely value of 550. Line 68 states that the decision-maker knows with certainty that the number of bikes currently deployed (NbrOfBikesCurrentlyDeployed) is 500.

In lines 69 to 74, the keyword **decision** expresses a decision with single option selection. The expression states that the unit cost of a bike (UnitCost) depends on the bike manufacturer brand. If the choice is to purchase the A-Bike brand, the cost is believed to be between £80 and £100, and if the choice is to purchase the Bianchi brand, the cost is believed to be between £200 and £300. Similar interpretations can be made if the option is to purchase Cortina Cyclese brand, Derby Cycle brand, or Catrike brand.

Also, in lines 78 to 82 the keyword **decision-subset** expresses a decision with multiple option selections. The expression states that the cost of bike localisation depends on the tracking mechanism. Zero cost is incurred if none of the tracking mechanism is selected. However, if the choice is to select only the GPS option, the cost is believed to be between £5m and £10m; if the choice is to select only the RFID option, the cost is believed to be between £2m and £7m. Finally, if the choice is to select both GPS and RFID options, the cost is the sum of the individual cost incurred by implementing each feature separately. We added the costs because of the '+' operator in the expression.

Lines 121 to 126 declare the constraint relationships in the problem using the keyword **Constraint** . The constraint expressions state that a bike of brand "A-Bike" does not support the GPS feature; the card dispenser option of the Kiosk registration and the key card option of the systems access management must be implemented together; and the key card option of the kiosk registration requires a key card reader.

3.5 Visualising AND/OR Refinements and Decision Dependencies

To help visualise the model structure, RADAR automatically generates the AND/OR refinement graph and decision dependency graphs from the model equations.

In RADAR's AND/OR refinement graphs, rectangles denote objectives; rounded rectangles denote random variables (i.e. variables characterised by a probability distribution rather than a single value), a black dot denotes an AND-refinement, an octagon denotes an XOR decisions, double octagon denotes OR decisions, arrows from a variable to an objective denotes that the objective refers to that variable. The leaf nodes in the refinement graphs are the model parameters. Their values are defined by probability distributions.

In the refactoring example, Fig. 3.1 shows that the objectives ENB and LP both refer to NB, that NB depends on Benefit *and* Cost (an AND-refinement), while Benefit depends on Benefit[As-is] *or* Benefit[Refactoring] based on which option is chosen (an

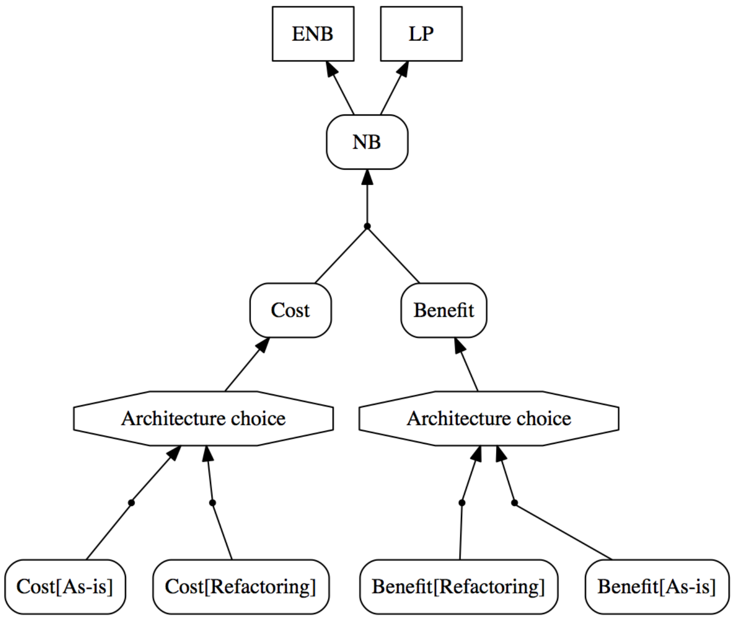


FIGURE 3.1: AND/OR refinement graph for the cost-benefit analysis example

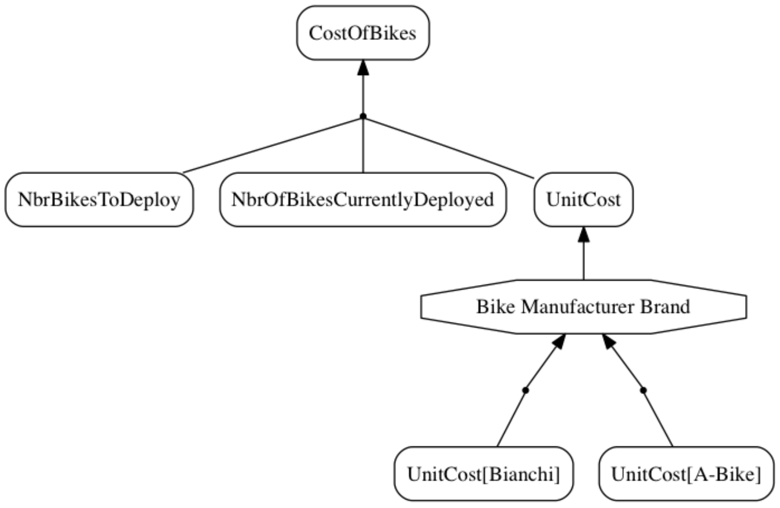


FIGURE 3.2: Partial AND/OR sub-graph for cost of bikes (CostOfBikes) in the bike sharing model. The rounded rectangles represent random variables, a black dot represents an AND-refinement, an octagon represents an OR-refinement. The leaf nodes in the graph represent the model parameter estimations.

OR-refinement). Similarly, in the bike sharing example, Fig. 3.2 represents a partial AND/OR graph of the bike sharing example that shows that CostOfBikes depends on NbrBikesToDeploy, NbrBikesCurrentlyDeployed *and* UnitCost which depends on UnitCost[Bianchi] *or* UnitCost[A-Bike].

RADAR decision graphs play a similar role to feature models in software product lines [236]; they help visualise the model decisions, their options and possible decision dependencies. In our refactoring model example, the decision graph presented in Fig.

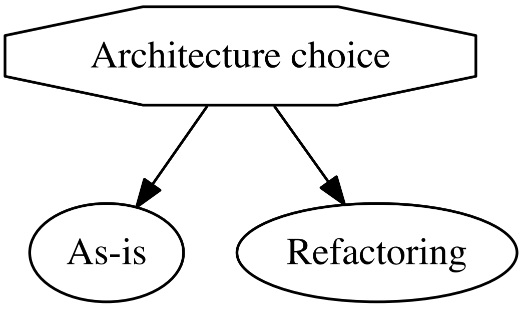


Figure 3.3: Decisions dependency graph for the cost-benefit analysis example

3.3 is extremely simple because the model includes a single decision and no decision dependency. Fig. 3.4 shows a more interesting decision graph for our bike sharing example. Octagons denote decisions with single option selection, double octagons represent decisions with multiple option selection, the ovals represent options, the arrows from decisions to options represent possible options for that decision; the arrows from option to a decisions state that such decisions have to be made only in situations where that option has been selected, the dotted arrows represent constraint relationships between options.

3.6 Analysing Decision Models

RADAR supports a decision analysis method that consists in first shortlisting a set of Pareto-optimal solutions through simulation and multi-objective optimisation, then computing the expected value of information to evaluate whether to seek additional information before making a decision between the shortlisted candidates [166]. This section provides a brief overview of this method and how RADAR supports it.

Fig. 3.5 shows the result of the analysis performed by RADAR on our small refactoring model. The first part shows the results of RADAR's optimisation analysis. It lists the optimisation objectives and the objective values for the two architecture choices: refactoring has an expected net benefit of £2m, but a loss probability of 23%, whereas keeping the current architecture has an expected net benefit of £1m but the loss probability is zero.

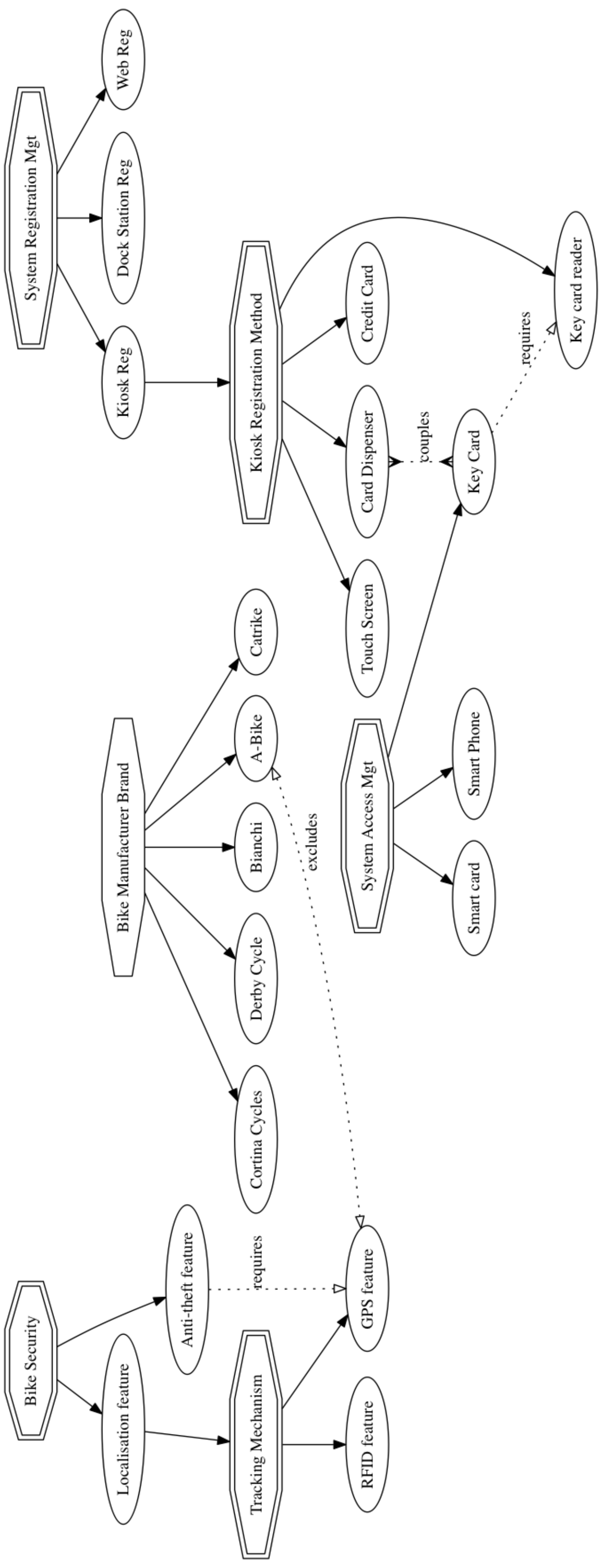


FIGURE 3.4: Partial decision dependency graph of the bike sharing model. Octagon represents XOR decisions, double octagons represent OR decisions, ovals represent options, arrows from decisions to options represent possible options for that decision; arrows from options to decisions state that such decisions have to be made only when that option has been selected, the dotted arrows represent constraint relationships between options.

Optimisation Analysis

Objective:	**Max** ENB
Objective:	**Min** LP

Architecture choice	ENB	LP
Refactoring	2	0.23
As-is	1	0

Information Value Analysis

Objective:	**Max** ENB
EVTPI:	0.64

Parameter	EVPPI
Benefit[Refactoring]	0.54
Cost[Refactoring]	0.14
Benefit[As-is]	0
Cost[As-is]	0

FIGURE 3.5: Analysis results for the cost-benefit analysis model

In this small refactoring example, we have only two solutions to choose from. Larger problems such as the fraud detection problem of Section 1.2 and the public bike sharing problem of Section 3.2 have a larger number of solutions. Before displaying the optimisation analysis results, RADAR shortlists the set of Pareto-optimal solutions and presents only those to the decision makers. A solution is Pareto-optimal if there is no other solution that is better on all objectives simultaneously [166]. In our small refactoring decision problem, a solution is thus Pareto-optimal if no other solution has both higher expected net benefit and lower loss probability. Here, both solutions are Pareto-optimal because none of them is better than the other on both objectives. For larger problems, shortlisting Pareto-optimal solutions can reduce a large set of solutions to a smaller set of candidates worthy of further investigation. For example, Fig. 3.6 shows the Pareto optimal solutions shortlisted through optimisation analysis of the public bike sharing model presented in Section 3.4.2. RADAR shortlists 35 candidate solutions out of a total of 15×2^{20} possible alternatives. The shortlisted solutions represent the trade-off between maximising expected net benefit and minimising risk.

The second part of Fig. 3.5 shows the result of information value analysis [133, 166]. In many decision situations, we might be able to perform additional data collection and analysis to reduce our uncertainty before making a decision. Additional data collection and analysis, however, are worthwhile only if their cost is lower than the value of the new information they will bring.

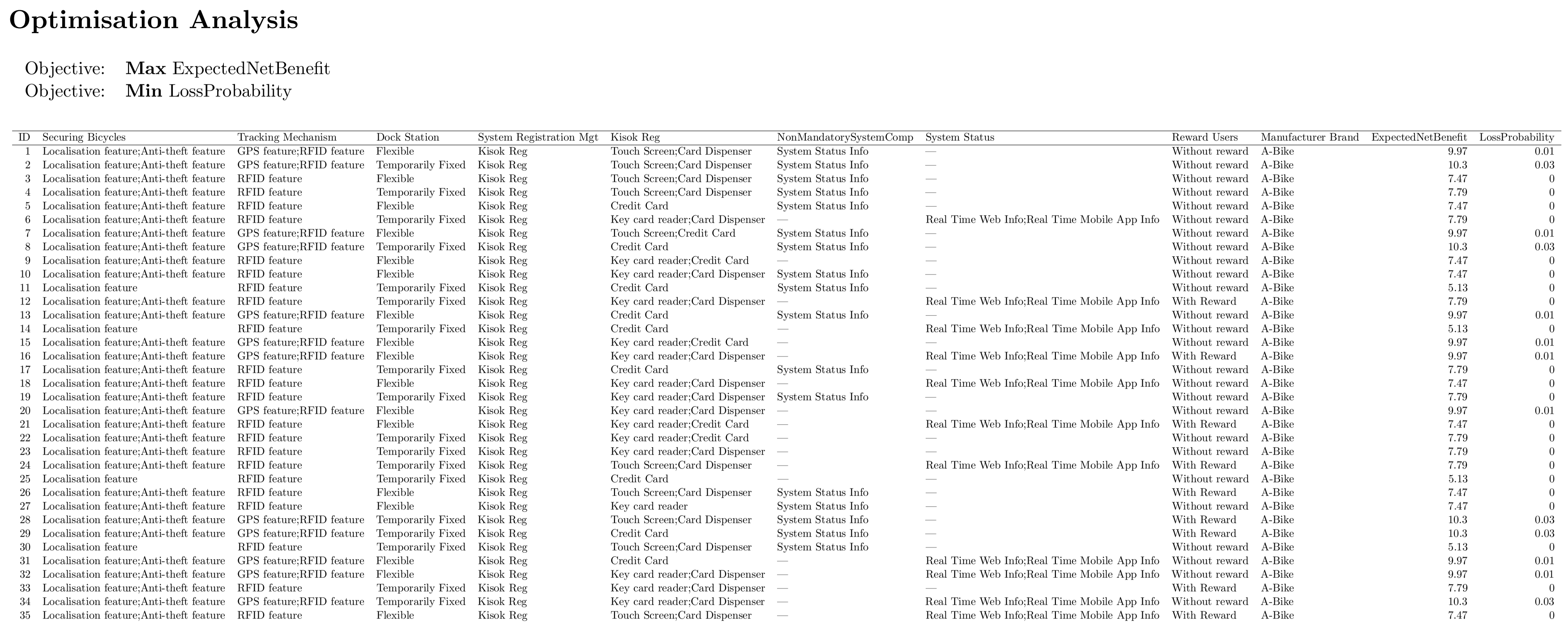
Evaluating the expected value of perfect information provides an upper bound on the value of additional data collection and analysis to our decision problem. The *expected value of total perfect information* (EVTPI) is a theoretical measure of the expected gain in some objective value (usually, maximising net benefit) that would result from having access to perfect information about all model parameters, that is from having access to an oracle who could tell us the exact values of all model parameters. The EVTPI gives an upper bound to the information that would result from additional data collection and analysis.

The expected value of partial perfect information (EVPPI) is the expected gain in some objective value resulting from having access to perfect information about a single model parameter Θ. It gives an upper bound to how much we should spend to reduce uncertainty about that model parameter. A more detailed explanation of these concepts can be found in previous publications [133, 134, 166, 224] and Chapter 5.

Analysing the expected value of information is important because it helps mitigate a measurement bias, known as measurement inversion, where decision makers would spend sometimes considerable efforts measuring quantities with low or even zero information values but disregard measuring quantities with high information value [136]. This bias has notably been observed in a study of 20 IT project business cases [135]. This study cites the effort spent by an organisation conducting detailed measurement of software development productivity as an example of measurement with very low information value, whereas quantities with high information value that are not measured at all are typically those related to benefits that are wrongly perceived to be intangible.

In the refactoring example, we evaluate information value with respect to maximising expected net benefit. The EVTPI is £0.64 million. Spending a small fraction of that amount on reducing uncertainty could have high value. The EVPPI analysis shows that reducing uncertainty about the benefits of refactoring has by far the highest value (£0.54m). By contrast, reducing uncertainty about the refactoring cost has little value and reducing uncertainty about benefits of the current architecture has no value.

One way to reduce uncertainty about the benefits of refactoring would be to elaborate a finer-grained decision model by refining the Benefit variable into lower-level variables (e.g. customers retention and acquisition rates, savings in software maintenance costs) and potentially identifying finer-grained architecture decisions corresponding to alternative ways to refactor the existing architecture. This would trigger a new decision analysis. The cycle of model refinement and analysis would eventually stop when the remaining expected value of perfect information is too low to justify further data collection and analysis.



Optimisation Analysis

Objective: **Max** ExpectedNetBenefit
Objective: **Min** LossProbability

ID	Securing Bicycles	Tracking Mechanism	Dock Station	System Registration Mgt	Kisok Reg	NonMandatorySystemComp	System Status	Reward Users	Manufacturer Brand	ExpectedNetBenefit	LossProbability
1	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	Without reward	A-Bike	9.97	0.01
2	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Temporarily Fixed	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	Without reward	A-Bike	10.3	0.03
3	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	Without reward	A-Bike	7.47	0
4	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	Without reward	A-Bike	7.79	0
5	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Credit Card	System Status Info	—	Without reward	A-Bike	7.47	0
6	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	7.79	0
7	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Touch Screen;Credit Card	System Status Info	—	Without reward	A-Bike	9.97	0.01
8	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	System Status Info	—	Without reward	A-Bike	10.3	0.03
9	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Key card reader;Credit Card	—	—	Without reward	A-Bike	7.47	0
10	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Key card reader;Card Dispenser	System Status Info	—	Without reward	A-Bike	7.47	0
11	Localisation feature	RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	System Status Info	—	Without reward	A-Bike	5.13	0
12	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	With Reward	A-Bike	7.79	0
13	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Credit Card	System Status Info	—	Without reward	A-Bike	9.97	0.01
14	Localisation feature	RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	5.13	0
15	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Key card reader;Credit Card	—	—	Without reward	A-Bike	9.97	0.01
16	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	With Reward	A-Bike	9.97	0.01
17	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	System Status Info	—	Without reward	A-Bike	7.79	0
18	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	7.47	0
19	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	System Status Info	—	Without reward	A-Bike	7.79	0
20	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Key card reader;Card Dispenser	—	—	Without reward	A-Bike	9.97	0.01
21	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Key card reader;Credit Card	—	Real Time Web Info;Real Time Mobile App Info	With Reward	A-Bike	7.47	0
22	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Credit Card	—	—	Without reward	A-Bike	7.79	0
23	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	—	—	Without reward	A-Bike	7.79	0
24	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Touch Screen;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	With Reward	A-Bike	7.79	0
25	Localisation feature	RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	—	—	Without reward	A-Bike	5.13	0
26	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	With Reward	A-Bike	7.47	0
27	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Key card reader	System Status Info	—	Without reward	A-Bike	7.47	0
28	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Temporarily Fixed	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	With Reward	A-Bike	10.3	0.03
29	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	System Status Info	—	With Reward	A-Bike	10.3	0.03
30	Localisation feature	RFID feature	Temporarily Fixed	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	Without reward	A-Bike	5.13	0
31	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Credit Card	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	9.97	0.01
32	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	9.97	0.01
33	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	—	—	With Reward	A-Bike	7.79	0
34	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	10.3	0.03
35	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Touch Screen;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	With Reward	A-Bike	7.47	0

FIGURE 3.6: Optimisation Analysis results for the public bike sharing model

Chapter 4

The RADAR Modelling Language

4.1 The Modelling Language

A RADAR model is defined as a tuple (Obj, Var, $Contr$), where Obj is set of objective definitions, Var is a set of variable definitions and $Contr$ is a set of constraint definitions.

4.1.1 Objective Definition

An *objective definition* has the form:

```
Objective  (Min | Max) Name = Statistic(X)
```

where Name is the objective name, **Min** or **Max** declares whether the objective function should be minimised or maximised, and Statistic(X) is a statistical measure on a single random variable X. Statistical measures include:

- **EV**(X) denoting the expected value of X.
- **Pr**(X $\sim$ x) denoting the probability that X $\sim$ x where $\sim$ is $\leq$, $<$, $=$, $>$, or $\geq$.
- x=**percentile**(X, i) denoting the i^{th} percentile of X, i.e. the value x such that $\mathbf{Pr}(X \leq x) = i$.

We saw examples of the use of first two types of statistics in the refactoring model and the bike sharing model in Chapter 3. Percentiles are useful statistics for measuring risk. For example, in the refactoring example, the Value at Risk (VaR) can be defined as:

Objective **Min** VaR = **percentile**(NB, 5);

4.1.2 Variable Definition

A *variable definition* is either an AND-refinement, an OR-refinement, or a parameter estimation.

An *And-refinement* has the form:

$$\text{X} = F(X_1, ..., X_n)$$

where X is a variable and $F(X_1, ..., X_n)$ is an arithmetic or Boolean expression involving variables $X_1, ..., X_n$.

The equation defining NB in bike sharing model in Section 3.2 is an example of And-Refinement where NB is defined as Benefit − Cost.

A *parameter estimation* has the form:

$$\text{X} = \textit{ProbabilityDistribution}$$

where *ProbabilityDisribution* defines a probability distribution for the variable X. Probability distributions include:

- **uniform**(min, max) denoting the uniform distribution between values min and max.
- **exponential**(x) denoting an exponential distribution that describes the time between events which occur at a constant average rate x.
- **triangular**(min, mode, max) denoting the triangular distribution with lower limit min, upper limit max and the most likely value mode.

- **normalCI**(lower, upper) denoting a normal distribution characterised by the lower and upper bounds of its 90% confidence interval (i.e. lower is the 5^{th} percentile and upper the 95^{th}) percentile.
- **geometric**(x, n) denotes the geometric distribution that describes the probability of occurrence of the first success (x) in n independent trials.
- **deterministic**(x) denoting that the variable has the certain value x.

Examples of parameter estimations in our motivating example are:

68 NbrBikesToDeploy = **triangular**(500, 550, 600);

69 NbrOfBikesCurrentlyDeployed = **deterministic**(500);

An *OR-refinement* is used to express alternative option choices at a decision point.*OR-refinements* can be characterised by exclusive-or (single option selection) or inclusive-or (multiple option selections).

An *OR-refinement* with exclusive-or (XOR) selection has the form:

X = **decision**(*label*){

  $Option_1$: $Expression_1$;

  ...

  $Option_n$: $Expression_n$;

}

where *label* is the decision name, $Option_i$ are option names, and $Expression_i$ is an AND-refinement or parameter estimation defining the value of X if $Option_i$ is selected. When $Expression_i$ is a parameter estimation, the value of a variable X is a parameter X[$Option_i$]. For example, in the cost-benefit refactoring model, the OR-refinement for Cost introduces the parameters Cost[As-is] and Cost[Refactoring].

The definitions of Cost and Benefit in the refactoring model are examples of OR-refinement with single option selection.

```
NbrFraudPerAccountBeforeBlocked = decision("blocking policy"){
        "block first" : NbrFraudBeforeDetection;
        "investigate first" : NbrFraudBeforeDetection + NbrFraudDuringInvestigation;
}
NbrFraudBeforeDetection = decision("processing type"){
        "continuous" : 1 / ContinuousTrueAlertRate;
        "batch" : NbrFraudsPerCompromisedAccountPerDay / BatchTrueAlertRate ;
}
ContinuousTrueAlertRate = decision("fraud detection method"){
        "classifier" : ContinuousAlertThreshold;
        "rule-based" : deterministic(0,75)
}
BatchTrueAlertRate = decision("fraud detection method"){
        "classifier" : BatchAlertThreshold;
        "rule-based" : deterministic(0,80);
}
ContinuousAlertThreshold = decision("alert threshold"){
        "high" : deterministic(0.9);
        "medium" : deterministic(0,8);
        "low" : deterministic(0,7);
}
BatchAlertThreshold = decision("alert threshold"){
        "high" : deterministic(0.95);
        "medium" : deterministic(0,85);
        "low" : deterministic(0,75);
}
```

FIGURE 4.1: Fragment of RADAR model showing all OR-refinements in the financial fraud detection system

Consider also Fig. 4.1 that shows all OR-refinements with single option selection of the fraud detection example introduced in Section 1.2. The first OR-refinement states that NbrFraudPerAccountBeforeBlocked depends on the blocking policy; the second that NbrFraudBeforeDetection depends on the processing type; etc. Justification for these equations can be found in the detailed model in Chapter 7.2.1

Multiple OR-refinements can refer to the same decision. For example, in Fig 4.1, the variables ContinuousTrueAlertRate and BatchTrueAlertRate depend both on the fraud detection method.

An *OR-refinement* with multiple option selection (inclusive-or) has the form:

```
X = decision-subset(op)(label){
    $Option_1$ : $Expression_1$;
    ...
    $Option_n$: $Expression_n$;
}
```

where *label* is the decision name, $Option_i$ denotes options' names, $Expression_i$ denotes AND-refinement or parameter estimation corresponding to the selection of $Option_i$, *op* is an arithmetic operator ('+' or '*') for combining multiple $Expression_i$ if multiple $Option_i$ are selected. The value of X, $Value(X)$, is defined as:

$$\mathbb{OP}_{o \in O(d)} selected(o) \times Expression(o) \tag{4.1}$$

where $\mathbb{OP}$ is either a $\sum$ or $\prod$, $O(d)$ is the set of options in decision d, $selected(o) = 1$ if option o is selected and 0 otherwise, $Expression(o)$ denotes AND-refinement or parameter estimation corresponding to the selection of option o. An *OR-refinement* with inclusive OR selection has a total of $2^{|O|}$ possible option combinations, where $|O|$ is the number of options of a particular decision.

An example of OR-refinements with multiple option selection from the bike sharing example is:

```
CostOfBikeLocalisation = decision-subset(+)("Tracking Mechanism"){
        "GPS feature" : normalCI(5,10);
        "RFID feature" : normalCI(2, 7);
}
```

The above example states that the cost of bike localisation depends on the tracking mechanism. Zero cost is incurred if none of the tracking mechanism is selected. However, if the choice is to select only the GPS option, the cost is believed to be between £5m and £10m; if the choice is to select only the RFID option, the cost is believed to be between

£2m and £7m. Finally, if the choice is to select both GPS and RFID options, the cost is the sum of the individual cost incurred by implementing each feature separately. The costs are added because of the '+' operator in the expression.

An inclusive-or refinement with a multiplication operator is useful for modelling reliability of system components. For example, in the bike sharing system example, the reliability of the system's localisation component is defined as:

```
ReliabilityOfSystemLocalisationComponent = 1 - FailureOfSystemLocalisationComponent
FailureOfSystemLocalisationComponent = decision-subset(*)("Tracking Mechanism"){
        "GPS feature" : FailureProbabilityOfGPSComponent;
        "RFID feature" : FailureProbabilityOfRFIDComponent;
}
FailureProbabilityOfGPSComponent = normalCI(0.01, 0.05);
FailureProbabilityOfRFIDComponent = normalCI(0.10, 0.20);
```

The above example states that the reliability of the system localisation components depends on the tracking mechanism. If none of the tracking mechanism is selected, the reliability is zero. However, if the choice is to select only the GPS option, the reliability is the failure probability of the GPS component deducted from one. Similarly, if the choice is to select only the RFID option, the reliability is the failure probability of the RFID component deducted from one. Finally, if the choice is to select both GPS and RFID options, the reliability is the product of the reliabilities of each component.

It is important to note that one can model inclusive-or option selections in the bike sharing example using the XOR construct of an *OR-refinement*. For example, we model the tracking mechanism decision that involves inclusive OR option selection between GPS and RFID as below:

```
 CostOfLocalisation=decision(“Tracking Mechanism”){
         “No feature” : deterministic(0);
         “GPS” :CostOfGPS;
         “RFID” : CostOfRFID;
         “GPS and RFID” : CostOfGPS + CostOfRFID;
}
CostOfGPS = normalCI(5,10);
CostOfRFID = normalCI(2, 7);
```

The limitation of using the XOR construct to model an OR option selection is that the modelling effort increases as the number of options for an OR decision increases. As a result, one would have to write expressions explicitly for all possible option combinations. This implies introducing AND-refinements (e.g. CostOfGPS and CostOfRFID) for each possible option combination. RADAR addresses this problem using the inclusive-or constructs.

4.1.3 Constraint Definition

RADAR provides constructs to model constraint relationships between options of decisions. Such *Constraints* have the syntax

Constraint $label_i : option_i \bowtie label_j : option_j$

where $option_i$ and $option_j$ are options' names, $label_i$ and $label_j$ are decisions names,and $\bowtie$ is a constraint relationship in the set {*requires*, *excludes*, *couples*}.

Definition (requires). $option_i : label_1$ *requires* $option_j : label_j$ means that the selection of $option_i$ implies the selection of $option_j$. However, $option_j$ can be selected without $option_i$ being selected.

Definition (excludes). $option_i : label_i$ *excludes* $option_j : label_j$ means that both $option_i$ and $option_j$ cannot be selected together.

Definition (couples). $option_i$: $label_i$ and $option_j$: $label_j$ are said to be coupled if none of $option_1$ or $option_2$ can be selected separately.

Examples of constraints relationships in the bike sharing model example of Section 3.4 are:

```
Constraint  "Bike Manufacturer Brand" : "A-Bike" excludes
        "Tracking Mechanism ": "GPS feature";
Constraint  "Kiosk Registration" : "Card Dispenser" couples
            "System Access Management": "Key Card";
Constraint  "System Access Management" : "Key Card" requires
            "Kiosk Registration" : "Key card reader";
```

The expressions above declare that a bike of brand "A-Bike" does not support the GPS feature; the card dispenser option of the Kiosk registration and the key card option of the systems access management must be implemented together; and the key card option of the kiosk registration requires a key card reader.

4.2 AND/OR Refinement Graph

The equations in a RADAR model create an acyclic AND/OR refinement graph between variables. An AND-refinement relates a variable to the set of variables involved in its definition. An OR-refinement relates a variable to the set of AND-refinement or parameter estimations involved in the OR-refinement definition. As an example, Fig. 4.2 shows the AND/OR refinement graphs for the fraud detection model fragments in Fig. 4.1.

The AND/OR refinement graph of a model must be acyclic. The tool generates an error if it detects a circular dependency between variables in the model.

By showing the variable dependencies, the AND/OR refinement graph helps the modellers to review and validate the model structure with other stakeholders. Such AND/OR graphs are commonly used in goal-oriented requirements engineering to communicate

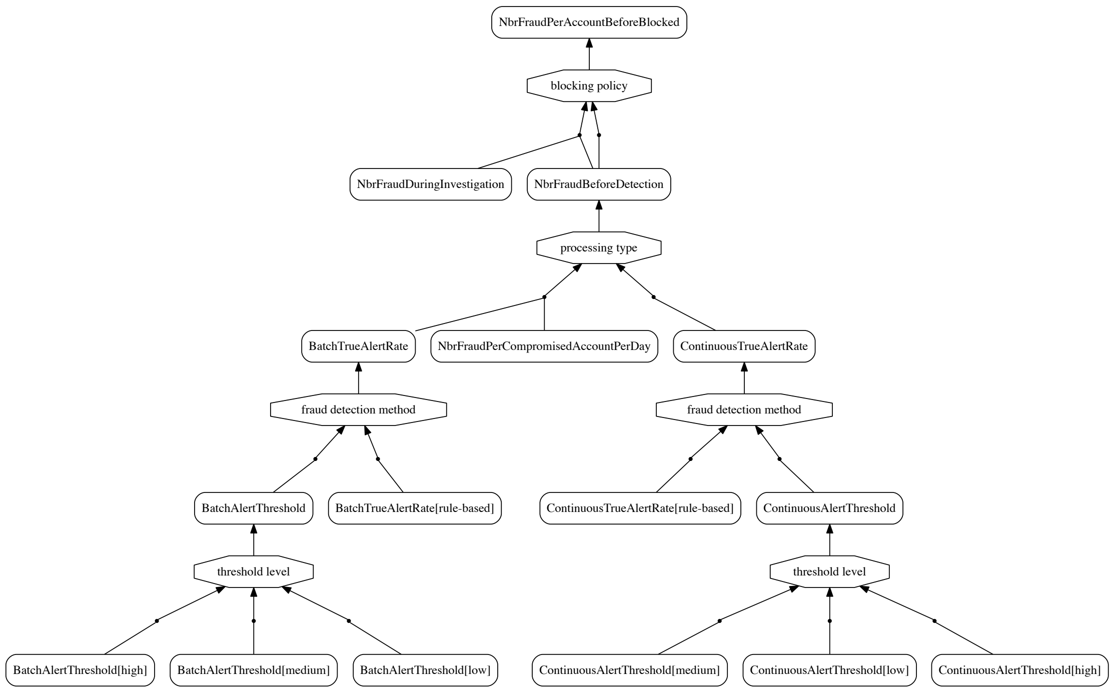

NbrFraudPerAccountBeforeBlocked
blocking policy
NbrFraudDuringInvestigation
NbrFraudBeforeDetection
processing type
BatchTrueAlertRate
NbrFraudPerCompromisedAccountPerDay
ContinuousTrueAlertRate
fraud detection method
fraud detection method
BatchAlertThreshold
BatchTrueAlertRate[rule-based]
ContinuousTrueAlertRate[rule-based]
ContinuousAlertThreshold
threshold level
threshold level
BatchAlertThreshold[high]
BatchAlertThreshold[medium]
BatchAlertThreshold[low]
ContinuousAlertThreshold[medium]
ContinuousAlertThreshold[low]
ContinuousAlertThreshold[high]

FIGURE 4.2: AND/OR Refinement Graph for the Financial Fraud Detection System

and validate traceability links between technical software characteristics (e.g. the classifier's true alert rate) and high-level stakeholders' concerns (e.g the financial loss due to fraud) [264].

4.3 The Design Space

A model's OR-refinement equations introduce a set of decisions and options. Selecting an option for a particular decision replaces all OR-refinements that refer to this decision by AND-refinements or parameters estimation corresponding to the selected option. Consider the fraud detection example, selecting the ”block first” option for the ”decision policy” option in Fig. 4.1 replaces the OR-refinement for NbrFraudPerAccountBeforeBlocked by the AND-refinement corresponding to the ”block first” option, i.e. NbrFraudPerAccountBeforeBlocked = NbrFraudBeforeDetection. Consider also the bike sharing system example, in the OR-refinement defined on the variable CostOfLocalisation between lines 79 to 84 of page 63, the selection of the “GPS” option of the “Tracking Mechanism” decision replaces the OR-refinement of CostOfLocalisation by the AND-refinement CostOfLocalisation = CostOfGPS. Similarly the selection of “GPS and RFID” option would replace the OR-refinement by CostOfLocalisation = CostOfGPS

+ CostOfRFID. This idea provides the basis for how we define the set of valid solutions for a RADAR decision model.

Definition (Solution). A RADAR solution, s, is defined as a mapping between decisions and the power set of options i.e. $s : D \to \mathbb{P}(O)$ such that $s(d) \subseteq O(d)$, where D is the set of all decisions in a model, $O(d)$ is the set of options in decision d, and $O = \cup_{d \in D} O(d)$ is the set of all options in a model.

For example, in the fraud detection model shown in Fig. 4.1, the solution $s_1 =$ {(**blocking policy**, block first), (**processing type**, {continuous}), (**fraud detection method**, {classifier}), (**alert threshold**, {high})} is an example of a RADAR solution with only XOR decisions. The solution $s_2 =$ {(**bike brand**, {A-bike}), (**Security Strategy**, {Localisation Feature}) (**Tracking Mechanism**, {GPS}), (**Dock station**, {flexible}), (**System Access**, {Smart Card}), (**System Registration**, {Kiosk}), (**Kiosk System Registration**, {Touch Screen}), (**Non Mandatory Component**, {(System Status, Bike Maintenance, Bike Redistribution}), (**System Status**, {Web Info}), (**Redistribution Reward**, {No Reward})} is a RADAR solution that contains both XOR and OR decisions.

Definition (Solution Space). Given any RADAR model, the size of the solution space is computed using $\prod_{d \in XOR-D} |O(d)| \times \prod_{d \in OR-D} 2^{|O(d)|}$, where the first factor gives the total solutions for XOR decisions and the second gives the total solutions for OR decisions. In the bike sharing example, which has 3 XOR decisions and 7 OR decisions, the solutions space is given as XOR$(5 \times 3 \times 2) \times$ OR$(2^2 \times 2^2 \times 2^3 \times 2^3 \times 2^4 \times 2^3 \times 2^2) = 15 \times 2^{20}$.

Definition (Design Space). Given any RADAR model, the design space is the set of *minimal* and *complete* solutions. A solution s is complete if applying s to a model replaces all OR-refinements with AND-refinements or parameter estimations that correspond to the selected option for each decision in s. A solution s is minimal if any subset of s is complete.

For example, in the bike sharing system, the solution $s_3 =$ {(**Bike Brand**, {A-bike}), (**Tracking Mechanism**, {GPS}), (**Dock station**, {flexible}), (**Redistribution Reward**, {Reward Users})} is minimal and complete. It is minimal because its subset s_4 = {(**Bike Brand**, {A-bike}), (**Dock station** {flexible}), (**Redistribution Reward**,

{Reward Users})} is complete. It is complete because applying the solution to the model results in no OR-refinements.

The *design space* of a RADAR model defines the set of all valid solutions to be considered during optimisation. RADAR generates the design space of a model by recursion over the acyclic AND/OR refinement graph by merging the set of solutions associated to a variable subgraph. For any model, the size of the design space is always smaller or equal to the size of the solution space. For the fraud detection model, the design space contains 16 solutions.

It is important to note that the decision-based solution encoding used in RADAR is different from the alternative option-based encoding commonly used in search-based software engineering, notably for the problem of selecting optimal designs in software product lines [111, 127, 234]. In the option-based encoding, solutions are encoded as a mapping $s : O \rightarrow Boolean$ such that for each option $o \in O$, $s(o)$ denotes whether o is selected or not. For problems with only XOR decisions, additional constraints must then be added to remove invalid solutions such as those that select two mutually exclusive options. With an option-based encoding, the solution space would include $2^{|O|}$ solutions against $\prod_{d \in D} |O(d)|$ for our decision-based encoding. In the fraud detection model, an option-based encoding would have resulted in $2^9 = 512$ total solutions instead of 24. For a slightly larger model including 10 decisions with 3 options each, an option-based encoding would include $2^{3*10} \approx 10^9$ total solutions, whereas the decision-based encoding would have only $3^{10} \approx 59000$ solutions, i.e. 0.005% of the number of solutions in the option-based encoding. The benefits of a decision-based encoding over an option-based encoding are thus enormous: the solution space is much smaller and it does not need additional constraints to remove invalid solutions.

4.4 The Decision Graph

The equations in a RADAR model may create dependencies between decisions. For example, in the fraud detection model, the "alert threshold" decision is dependent on the selection of the "classifier" option in the "fraud detection method" decision.

Definition (Decision Dependency). A decision d_1 is *dependent* on the selection of option x in decision d_0 if, and only if, for all solutions s in the design space, if d_1 is defined then the selected option for decision d_0 is x; formally: $\forall s \in \mathit{DesignSpace} \mid d_1 \in \mathit{dom}(s) \Rightarrow s(d_0) = x$, where $\mathit{dom}(s)$ denotes the domain of the function s, i.e. the set of decisions that have a mapping in s.

RADAR infers decision dependencies by first generating the design space, then checking for dependency between every pair of decisions. To visualise such dependencies, the tool generates a decision diagram showing all decisions, their options, and dependencies between decisions and options. The decision diagram for the fraud detection model is shown in Fig. 4.3. These diagrams play a similar role to that of feature diagrams in software product lines [236]: they help us visualise a potentially large design space in terms of a smaller set of decisions and options.

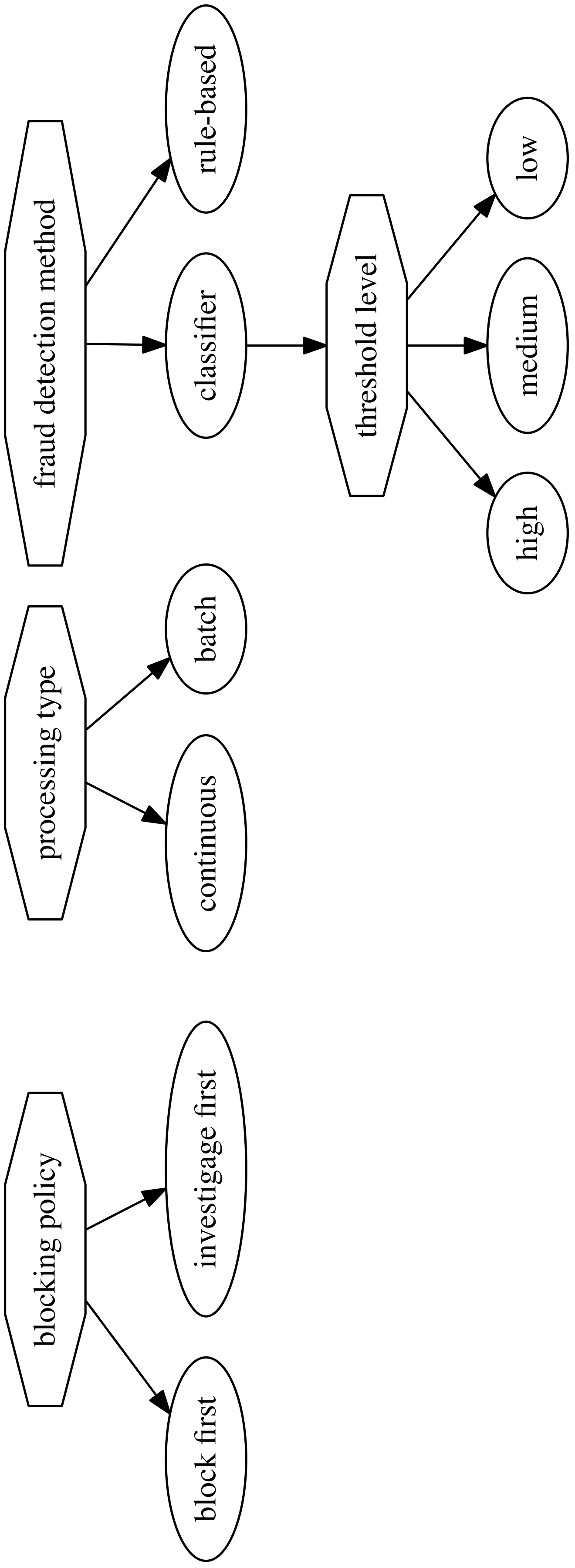


Figure 4.3: Decision Graph for the Financial Fraud Detection System

Chapter 5

RADAR Decision Analysis

In Chapter 4, we have formally presented RADAR's modelling language. RADAR's modelling language extends and simplifies quantitative AND/OR goal models used in requirements engineering, and has relations to feature diagrams used in software product lines. This chapter describes RADAR's decision analysis technique which is based on the statistical decision analysis steps presented in Chapter 2.4.8.

Once a decision-maker, such as software architects or requirements engineers, develops a RADAR model that defines the problem objectives, refinement equations, decisions and parameter uncertainty, then RADAR performs two main types of analysis: the optimisation analysis and information value analysis.

5.1 Optimisation Analysis

Given a model's abstract syntax tree (AST), the optimisation analysis consists of three activities: generating the design space, simulating a design solution and shortlisting the Pareto-optimal solutions using exhaustive search or evolutionary multi-objective algorithms.

5.1.1 Generating Design Space

The design space is generated through a single traversal on RADAR's semantic model (an acyclic AND/OR refinement graph that consist of different model elements, such as objectives, variables, decisions, expressions and operators) created from a model's abstract syntax tree. The design space consist of all valid solutions to be used for the simulation and optimisation process.

Algorithm 1 presents a pseudo-code for generating the design space of a RADAR model. The algorithm takes as input the RADAR model and for each objective (line 2), it gets the variable ($ObjVar$) associated to the objective definition (line 3) and then generates the solution set of the objective variable (line 4), by invoking the *getAllSolutions* method (described in Algorithm 2) starting from the variable's definition and then recursively traversing the model until reaching the leaf-variables, i.e., $ParameterEstimations$. The solution set obtained for each objective are then merged using Algorithm 3 (line 5).

In Algorithm 2, the solution set (S) returned depends on whether the calling variable's definition is a parameter estimation, an AND-Refinement, or an OR-refinement. If the variable's definition is a parameter estimation (line 3), S is always an empty set (line 4), since a parameter estimation has no decision. If the variable's definition is an AND-Refinement (line 7), S is obtained by merging the solution set obtained (using Algorithm 3) for each Boolean or arithmetic expression that define the AND-Refinement. If the variable's definition is an OR-Refinement (line 13); first, we retrieve the solution set for each option declared in the OR-Refinement by invoking the *getSolutionSetPerOption* method (line 14). Next, the decision (d) associated with OR-Refinement is obtained (line 15): if the decision type is mutually exclusive (XOR), we combine all solution set obtained for each AND-Refinement corresponding to each option in the OR-Refinement (lines 18-21). If the decision type is inclusive-or (OR), we obtain all possible option combinations ($optionCombinations$) for the OR-decision (line 24), and for each option combination (oc), the algorithm gets the combined solution set (lines 27-29) and update each solution s in the solution set ss with decision d and a list of option (lines 30-33).

In Algorithm 3, the merging of two solution sets S_1 and S_2 requires a pair-wise comparison of all solutions $s_1 \in S_1$ and $s_2 \in S_2$, and if they do not conflict (line 10), i.e. if they do not disagree on the selected option for the same decisions, then s_1 and s_2 are combined (line 11).

The time and space complexity of generating the design space is $\mathcal{O}(m)$, where m is the model length measured as number of nodes in the model's abstract syntax tree .

```
Input  : Model $m = \{Obj, Var, Constr\}$
Output: SolutionSet $S$
1 $S \leftarrow \emptyset$
2 foreach Objective obj in $m.Obj$ do
3     Variable $objVariable \leftarrow obj$.getObjectiveVariable()
4     SolutionSet $objSolutionSet \leftarrow objVariable$.getAllSolutions(m)
5     $S$.mergeSolutionSet($objSolutionSet$)
6 end
7 return $S$
```

Algorithm 1: GenerateDesignSpace

5.1.2 Simulating A Design

This is the most computationally expensive step. It returns simulation values of all objectives for a candidate design s, by evaluating the impact of s on the model's objectives through Monte-Carlo simulation (MCS) [113] –a mathematical technique for exploring the range of possible outcomes when analysing a model and the chance that the outcome will occur. Given a RADAR semantic model generated from a model's abstract syntax tree (AST), RADAR's simulation algorithm traverses the semantic model (acyclic AND/OR refinement graph) by recursion while sampling and evaluating large number of possible scenarios. Scenarios are generated through probability distributions of the model's parameter estimations.

Algorithm 6 describes the simulation of a single design: it takes a RADAR model and a candidate design s as inputs, then simulates s (lines 2-9). The algorithm loops through each model objective (line 2), then gets the variable associated to the objective definition (line 4), and simulates solution s through recursion over the variable's sub-graph (line 5) by invoking the *Simulate* method in Algorithm 7. *Simulate* returns an array of simulated objective values. If the variable's definition is a parameter estimation (line 3), the algorithm returns N samples of the probability distribution (e.g. **normal**, **triangular** and **normalCI**) associated to the variable's definition. If the variable's definition is an ANDRefinement (line 7), the algorithm determines whether the ANDRefinement's definition is a *BinaryExpression* or a *UnaryExpression*: if the definition is a *BinaryExpression* (line 8), the algorithm simulates the left and right expressions and returns the combined expressions using a binary operator, which may be an arithmetic operator (+, -, /, $\times$) or Boolean operator(!, &&); if the definition is a *UnaryExpression* (line 13), the algorithm simulates the expression and returns the simulated values with the unary

```
Input   : Model, $m = \{Obj, Var, Constr\}$
Output: SolutionSet $S$
$S \leftarrow \emptyset$
Expression $expr \leftarrow this$.getDefinition
if $expr$ == $ParameterEstimation$ then
    Solution $newSol \leftarrow \emptyset$
    $S$.add($newSol$)
end
if $expr$ == $ANDRefinement$ then
    foreach $Expression\ childVar\ in\ expr.Children$ do
        SolutionSet $andSolSet \leftarrow childVar$.getAllSolutions($m$)
        $S$.mergeSolutionSet($andSolSet$)
    end
end
if $expr$ == $OR\_Refinement$ then
    Map<String, SolutionSet> $spo \leftarrow$ getSolutionSetPerOption($m$,$expr$)
    Decision $d \leftarrow expr$.getDecisions
    if $d.decisionType$ = $ExclusiveOR\_Refinement$ then
        $i \leftarrow 0$
        foreach $Map<String, SolutionSet>\ entry : spo.EntrySet()$ do
            $S$.addAll($entry.Value$[i])
            $i++$
        end
    end
    if $d.decisionType$ == $InclusiveOR\_Refinement$ then
        List<List<String>> $optionCombinations \leftarrow$
          getAllOptionCombinations($d$)
        foreach $List<String>\ oc\ in\ optionCombinations$ do
            SolutionSet $ss \leftarrow \emptyset$
            foreach $String\ option\ in\ oc$ do
                $ss$.addAll($spo$[$option$])
            end
            foreach $Solution\ s\ in\ ss$ do
                $s$.addDecision($d$, $oc$)
                $S$.add($s$)
            end
        end
    end
end
return $S$
```

Algorithm 2: GetAllSolutions: algorithm to recursively retrieve all solutions staring from a specified variable to the leaf variables in the goal graph.

```
Input  : SolutionSet $S_1$
Output: SolutionSet $S$
1  $S \leftarrow \emptyset$
2  if $S_1$.empty() then
3      return this
4  end
5  if this.empty() then
6      return $S_1$
7  end
8  foreach solution $s_1$ in $S_1$ do
9      foreach solution $s_2$ in this do
10         if $!conflicting(s_1, s_2)$ then
11             Solution $newSolution = s_1$.union($s_2$)
12             $S$.add($newSolution$)
13         end
14     end
15 end
16 return $S$
```

Algorithm 3: MergeSolutionSets: algorithm to merge two solutions if they do not conflict in selected options.

```
Input  : Model, $m = \{Obj, Var, Constr\}$, Expression $expr$
Output: Map<String, SolutionSet> $S$
1  $S \leftarrow \emptyset$
2  Decision $d \leftarrow expr$.getDecisions
3  foreach option in d.getOptions do
4      AND_Refinement $andRef \leftarrow expr$.get($option$)
5      SolutionSet $andRefSolutions \leftarrow andRef$.getAllSolutions($m$)
6      foreach solution soln in $andRefSolutions$ do
7          $soln$.addDecision($d, option$)
8      end
9      $S$.put($option$, $andRefSolutions$)
10 end
11 return $S$
```

Algorithm 4: GetSolutionSetPerOption

```
Input  : Solution, $s_1, s_2$
Output: Boolean $true/false$
1  foreach decision d in $s_1$.getDecisions do
2      if $s_2.selection(d) \neq null$ and $s_2.selection(d) \neq s_1.selection(d)$
        then
3          return true
4      end
5  end
6  return false
```

Algorithm 5: ConflictingSolutions: algorithm to check if two solutions conflict in their selections.

operator (e.g. -, !) applied on them. If the variable's definition is an OR_Refinement with single option selection (exclusive-or), the algorithm simulates the AND_Refinement of the selected option o in decision d; if the variable's definition is an OR_Refinement with multiple option selection (inclusive-or), the algorithm combines (adds or multiplies) the simulated values of each AND_Refinement corresponding to a selected option o in decision d.

```
Input  : Model, $m = \{Obj, Var, D, O\}$, Design ($s$)
Output: SimulatedObjectiveValues $SOV$
1 Double [ ]$SOV \leftarrow \emptyset$
2 foreach $Objective\ obj\ in\ m.Objectives$ do
3     Statistic $st \leftarrow obj$.getDefinition()
4     Variable $objVariable \leftarrow obj$.getObjectiveVariable()
5     Double [ ]$objectiveValueArray \leftarrow st$.Simulate($s$, $objVariable$)
6     Double$objectiveValue \leftarrow$ Mean($objectiveValueArray$)
7     $SOV$.add($objectiveValue$)
8 end
9 return $SOV$
```

Algorithm 6: Simulating A Design

Simulating a design involves a single recursive traversal of the semantic model. Thus the time and space complexity is $\mathcal{O}(m)$, where m is the number of nodes in the model's AST. Generating N simulations for all objectives and a set of solutions thus has a time and space complexity of $\mathcal{O}(|Obj| \times |SS| \times N \times m)$, where Obj is the model objectives, SS is the set of solutions considered during simulation, N is the number of simulations.

5.1.3 Shortlisting Pareto Optimal Solution using Exhaustive Search

RADAR shortlists Pareto optimal solutions together with their objective values through multi-objective optimisation technique presented in Chapter 2.1. By default, RADAR finds optimal solutions through exhaustive search of the design space. As shown in Algorithm 8, its exhaustive strategy implementation involves comparing pairs of solutions and selecting the one with higher objective value(s). This algorithm has a complexity of $\mathcal{O}(|S|^2)$, where S is the number of simulated solutions the algorithm takes as input.

```
Input  : Solution ($s$), Variable ($Var$)
Output: Array of Simulated Objective Values ($SOV$)
Double [ ] $SOV \leftarrow \emptyset$
Expression $expr \leftarrow Var$.getDefinition()
if $expr == ParameterEstimation$ then
    Distribution $distr = expr$.getDefinition()
    return $distr.getProbabilityDistribution()$
end
if $expr == ANDRefinement$ then
    if $expr.Definition == BinaryExpression$ then
        $leftSim \leftarrow$ Simulate($s$, $expr.leftExpr$)
        $rightSim \leftarrow$ Simulate($s$,$expr.rightExpr$)
        return $leftSim \oplus rightSim$
    end
    if $expr.Definition == UnaryExpression$ then
        return $\oplus Simulate(s,expr)$
    end
end
if $expr == OR\_Refinement$ then
    Decision $d \leftarrow expr$.getDecisions()
    List<Option> options $\leftarrow s[d]$
    foreach $Option\ o\ in\ options$ do
        ANDRefinement $andRef \leftarrow expr$[o]
        Double [ ] $simValue \leftarrow$ Simulate($s$, $andRef$)
        if $d.decisionType = ExclusiveOR\_Refinement$ then
            $SOV \leftarrow simValue$
            break
        end
        if $d.decisionType == InclusiveOR\_Refinement$ then
            Integer $i \leftarrow 0$
            foreach $Double\ sv\ in\ simValue$ do
                if $d.op == Addition$ then
                    $SOV[i] \leftarrow SOV[i] + sv$
                end
                if $d.op == Multiplication$ then
                    $SOV[i] \leftarrow SOV[i] \times sv$
                end
                $i$++
            end
        end
    end
end
return $SOV$
```

Algorithm 7: RADAR's Simulate method: $\oplus$ is a binary operator (+, -, /, $\times$) or a Boolean operator (!, &&)

```
Input  : SimulatedSolutions ($SimS$), Model, $m = \{Obj, Var, D, O\}$
Output: Pareto Set $S$
1  $S \leftarrow \emptyset$
2  SolutionSet $SS$ = $SimS$.getEvaluatedSolutions()
3  $solutionSize \leftarrow SS$.size()
4  $ObjectiveValues$ = $SimS$.getObjectiveValues()
5  Boolean[ ] isPareto = new Boolean[solutionSize]
6  ArrayFill(isPareto, true)
7  int $i = 0$
8  while $i < ObjectiveValues.size()-1$ do
9      int $j = i + 1$
10     while $isPareto[i]$ AND $j < ObjectiveValues.size()$ do
11         if $ObjectiveValues[i].dominates(ObjectiveValues[j])$ then
12             isPareto[$j$] $\leftarrow$ false
13         else
14             isPareto[$i$] $\leftarrow$ false
15         end
16         $j$++
17     end
18     $i$++
19 end
20 int $k = 0$
21 foreach $Boolean\ b\ in\ isPareto$ do
22     if $b == true$ then
23         $S$.add($S[k]$, $ObjectiveValues[k]$)
24     end
25     $k$++
26 end
27 return $S$
```

Algorithm 8: RADAR exhaustive strategy to find Pareto optimal solutions.

5.1.4 Shortlisting Pareto Optimal Solution using Evolutionary Algorithms

In a situation where a RADAR model contains OR decisions with multiple options selections, the size of the solution space increases, and therefore may make the use of exhaustive search infeasible because it requires enormous memory resources to store matrix results *Result* of dimension $|DS| \times |Obj|$, where DS is the model's design space and Obj the model's objectives [43]. To overcome this problem, RADAR implements alternative search-based approaches, such as evolutionary multi-objective optimisation algorithms [287] to explore solution space in shortlisting Pareto optimal solutions.

Evolutionary Multi-objective Optimisation Algorithms (EMOAs), such as NSGAII [69], use the principles of natural selection and evolution to iteratively evolve a set of solutions (chromosomes), called a population, towards Pareto-optimality [287]. Applying EMOAs requires (i) selecting the population size, the number of generations, and the mutation and crossover probabilities that will determine how the best solutions (parents) from one generation will be combined to form new solutions (offsprings) for the next generation; (ii) determining a termination criterion, which could be either a fixed number of fitness function evaluations, or a fixed amount of time the algorithm is allowed to run [122]. Guidelines on how to use EMOAs have been proposed in the field of search-based software engineering [118]—a field that reformulates software engineering problems as optimisation problems, and then search for optimal or near optimal solutions in a solution space guided by the fitness functions.

RADAR uses elitist EMOAs, such as NSGAII [69], SPEAII [289], MoCell [184] and IBEA [288], implemented in JMetal5 [185]— a JAVA-based optimisation framework. These algorithms are elitist because they maintain their best solutions throughout the generations. Apart from IBEA, the other algorithms are similar in dominance criteria: they use the Boolean Pareto dominance (see definition 2.2) and give priority to solutions that are more spread out in the Pareto front. IBEA, however, incorporates decision-maker's preference information when estimating its dominance values between two solutions, i.e., it assign weight values to each solution based on quality indicator, typically hypervolume [288].

Next, we detail RADAR's approaches for handling the key ingredients of search-based optimisation approaches, i.e., the solution encoding, the fitness function evaluation, a set of variation operators (i.e. crossover and mutation), and the optimisation approach used in handling constraints.

Solution Encoding. Search-based approaches generally encode solutions using bit strings i.e. a vector of bits [122, 281]. For RADAR's EMOAs to explore the solution space of a model, we encoded the solutions using an "array of bit strings", where the size of the array is the number of decisions in the model, $|D|$, and each element of the array, i.e. the bit strings, represents the set of options, $O(d)$, corresponding to decision $d \in D$, and are used to encode mutually exclusive option (XOR) and inclusive option (OR) selections. A bit value of 1 (respectively 0) denotes an option represented by the bit is selected (respectively not selected).

For example, in the bike sharing model example, the solution s = {(**Bike Manufacturer Brand**, {A-bike}), (**Bike Security**, {localisation feature}), (**Tracking Mechanism**, {GPS}), (**Dock station**, {flexible}), (**System Access**, {smart card}), (**System Registration**, {kiosk}), (**Kiosk System Registration**, {touch screen}), (**Non Mandatory Component**, {(system status, bike maintenance, bike redistribution)}), (**System Status**, {web info}), (**Redistribution Reward**, {no reward})} has the equivalent binary representation of {[10000],[10],[10],[001],[100],[100],[1000],[111],[10],[10]}. In this representation, each bit string encodes option selection of decisions in the bike sharing model. The bit strings in this representation have an order that corresponds to the order in which the options of each decision are presented in the RADAR model. For example, the bit string "10000" in which only the A-bike option is selected encodes all options in the XOR decision 'Bike Manufacturer Brand' of solution s below:

```
UnitCost = decision ("Bike Manufacturer Brand"){
          "A-Bike": normalCI(80 ,100);
          "Bianchi" : normalCI(200 ,300);
          "Cortina Cyclese": normalCI(100 ,150);
          "Derby Cycle": normalCI(140, 200);
          "Catrike" : normalCI(250 ,350);
  }
```

Also, the bit string "111" in which the system status, bike maintenance and bike reward options are selected encodes options in the OR decision 'Non Mandatory Component' of solution s below:

```
CostOfOtherComponents = decision-subset(+)("Non Mandatory Component"){
          "System Status Info" : CostOfStatusInfo;
          "Bike Maintenance" : triangular(8,10, 12);
          "Bike Redistribution" : CostOfRedistribution;
  }
```

Fitness Evaluation. RADAR performs a decoding process in order to evaluate the fitness function of candidate solutions. The decoding process involves a conversion of EMOA solutions (encoded as array of bit strings) to a RADAR solution (represented as a mapping from decisions to power set of options). Such decoding is necessary for evaluating the impact of alternative solutions on the objectives through a simulation of model parameters.

Crossover and Mutation. RADAR performs the crossover and mutation operations ensuring that the offspring produced by combining two parent solutions have valid set of decision choices in the model. Currently, RADAR implements single point crossover, but can be extended to use other crossover methods available in JMetal5 [185], such as uniform crossover and two-point crossover. Given a RADAR solution encoded using an array of bit strings of size n, the crossover point, $cp(0 < cp < n)$, is randomly chosen at the boundaries of adjacent bit strings in the array. The offspring are generated through a copy of array contents from the start of the crossover point from the first parent and the remaining contents from the second parent.

For the mutation operation, RADAR implements bit flip mutation such that the decisions characteristics are preserved, i.e., a decision with a single option selection is not transformed to decisions with multiple option selection, and vice versa.

Constraint Handling. When a RADAR model has constraint(s), there is the possibility of generating and evolving invalid solutions, and consequently shortlisting these solutions in the Pareto front approximation. For instance, in our bike sharing example, the solution s = {(**Bike brand**, A-bike), (**Bikes Security**, Anti-theft feature), (**Tracking**, RFID), (**Dock station**, flexible), (**Redistribution Reward**, Reward Users)} is invalid. This is because it violates the constraint that the Anti-theft feature of Bike security requires the selection of GPS option of Tracking mechanism. In order to guide the search towards valid solutions in the Pareto front, an approach is to consider an extra objective, which is the number of violated constraints which is minimised [130, 235].

RADAR adopts two established optimisation approaches to handle constraint relationships in a decision model [130]. These are the (λ+1) approach and 1+λ approach, where λ is the number of optimisation objectives declared in the RADAR model.

The (λ+1) approach is a classical approach used in optimal product selection based on feature models [235]. In this approach, all optimisation objectives, including the "number of violated constraints", are treated equally during the optimisation process. The parenthesis around "λ+1" gives the notion that the model objectives and the objective "number of constraints violated" are considered simultaneously.

The 1+λ approach, introduced by Hierons et al. [130], treats the objective "number of violated constraints" as the primary and first objective during optimisation and other objectives declared in the model are treated equally as the secondary objectives. This approach gives priority to a solution with fewer constraint violations when estimating dominance value between two solutions, and if the number of violated constraints of the compared solutions are the same, then, the solution with better fitness is preferred.

Although the EMOAs implemented in RADAR have better scalability than the exhaustive strategy. These algorithms do not guarantee finding the Pareto optimal solutions. This is because they are designed to explore a subset of the problem solution space, i.e., some design decisions may not be considered during the solution encoding step of an EMOA. As a consequence, when these algorithms are used to analyse a decision model, RADAR may generate a partial decision dependency graph that would not contain all model decisions, options and relationships between decisions and options. Such partial decision dependency graph, for example in Figure 3.4 of Section 3.4, can be used to provide information about how much of the solution space was explored by running an EMOA on a problem.

5.2 Information Value Analysis

Uncertainty complicates requirements engineering and software architecture decisions. Uncertainty is the lack of total knowledge about the actual consequences of alternative decision choices on stakeholders' goals [166]. Despite the inherent uncertainty in requirements and architecture decisions, requirements engineers and software architects have to make decisions. Thus, to aid requirements and architecture decisions under uncertainty, RADAR automatically computes the expected value of information [133, 166, 224] which is defined as the "expected gain in net benefit between the selected alternatives with and without additional information" [166]. The expected value of information helps decision makers answer the question of whether reducing uncertainty about model parameters may lead to the selection of an alternative with a higher net benefit (see section 3.6 for illustrative example).

RADAR estimates the financial value of reducing uncertainty in model parameters with respect to a given set of optimal solutions and a single model objective (e.g. **Max** ExpectedNetBenefit). Based on the estimated financial value, decision-makers can make a decision or seek additional information or even perform further analysis through requirements elicitation, prototyping and modelling. Then the decision model is updated to trigger a new modelling and analysis iteration. Such iteration will stop once the financial value of additional information is insufficient to justify the effort required.

The expected value of information about an objective can be computed given perfect and imperfect information. RADAR, however, computes the expected value of information given only perfect information about model parameter(s), as expected value of information with imperfect information may not produce significant benefit over simple information value analysis [166]. Specifically, RADAR computes the expected value of total perfect information (EVTPI) and expected value of partial perfect information (EVPPI) [133, 166].

The EVTPI about all non-deterministic model parameter estimations, Ω, is a theoretical measure of the upper bound to the value of reducing uncertainty through additional data collection or analysis. The EVTPI for an objective variable $\hat{X}$ can be computed using the formular below [166]:

$$EVTPI = \frac{1}{N}\sum_{i=1}^{N} \max_{j:1..M} \hat{X}[i,j] - \max_{1:1..M} \frac{1}{N}\sum_{i:1}^{N} \hat{X}[i,j]$$

where $\hat{X}[i,j]$ represents the objective value of X in a simulation $i = \{1 \ldots N\}$ for a solution $j = \{1 \ldots M\}$. The first term in the expression above refers to the expectation of the highest value of objective $\hat{X}$ over all possible value of the model parameters. The second term in the expression is the highest expected $\hat{X}$ in the presence of the current uncertainty about model parameters. The EVTPI is always positive or zero, and the time complexity of the formular is $\mathcal{O}(N \times |S|)$, where N is the number of simulations and S is the shortlisted solutions.

The EVPPI of a single model parameter with respect to an objective gives us information about whether reducing uncertainty about a single model parameter is worthwhile. In other words, EVPPI gives the expected gain in the objective $\hat{X}$ given perfect information about a single model parameter Θ. RADAR estimates EVPPI using a recent efficient algorithm that computes EVPPI for a model parameter Θ from $\hat{X}$ and the vector $\vec{\Theta}$ containing the simulations of parameter Θ [224]. The EVPPI is always positive or zero.

In the bike sharing model example presented in Section 3.2, RADAR estimates the EVTPI as £0.81m. It also estimates the EVPPI about the unit cost of A-Bike brand to be £0.03m, and the EVPPI of the number of bikes to deploy to be £0.70m. This means that reducing uncertainty about the number of bikes to deploy has a higher value than reducing uncertainty about the cost of the A-Bike brand.

Chapter 6

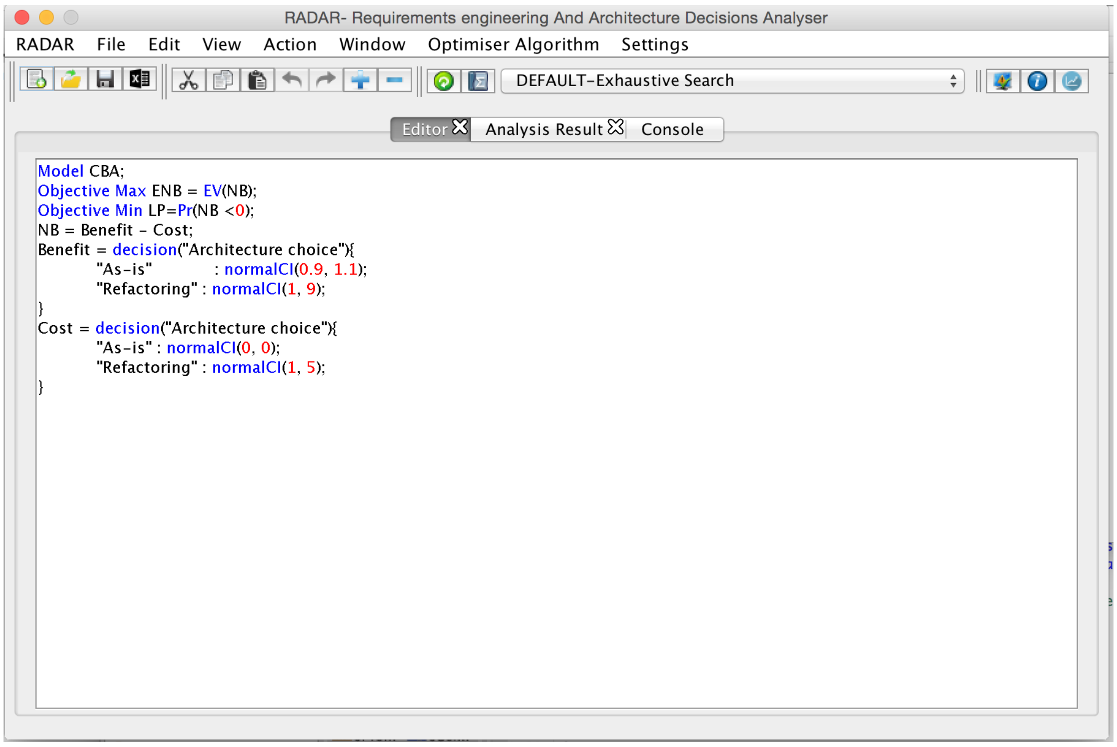
The RADAR Tool

This chapter describes the design of the RADAR tool. We first present an overview of the RADAR tool by describing how to make decisions with the tool. This is followed by a presentation of the detailed tool design which entails an overview of RADAR's system architecture; RADAR's implementation details; the description of RADAR's context free grammar, the tool's semantic model used during decision analysis.

6.1 Running RADAR

The RADAR tool is a self-contained jar file. Figure 6.1 shows RADAR's graphical user interface (GUI) that appears once the tool is launched. The GUI consists of three main tabs. These include: (i) An *editor tab* that modellers can use to write decision models and load existing models. The decision model in the editor tab of Figure 6.1 is for the refactoring decision problem introduced in Section 3.4. Once the model is completely written, modellers can analyse the model by clicking the "solve" button under the "action" menu. (ii) *Analysis result tab* which displays the results of the optimisation and information value analysis. Figure 6.2 shows the analysis results of refactoring decision problem. These results are also saved in a directory where the jar file is located. (iii) *Console tab*, shown in Fig. 6.3, for logging RADAR's analysis status, such as the time taken for each decision analysis steps.

Additional tabs are displayed on completion of RADAR decision analysis. These include: the AND/OR goal graph of a decision model (Figure 6.5); the decision dependency graph


RADAR- Requirements engineering And Architecture Decisions Analyser
RADAR File Edit View Action Window Optimiser Algorithm Settings
DEFAULT-Exhaustive Search
Editor
Analysis Result
Console

```
Model CBA;
Objective Max ENB = EV(NB);
Objective Min LP=Pr(NB <0);
NB = Benefit - Cost;
Benefit = decision("Architecture choice"){
        "As-is"         : normalCI(0.9, 1.1);
        "Refactoring" : normalCI(1, 9);
}
Cost = decision("Architecture choice"){
        "As-is" : normalCI(0, 0);
        "Refactoring" : normalCI(1, 5);
}
```

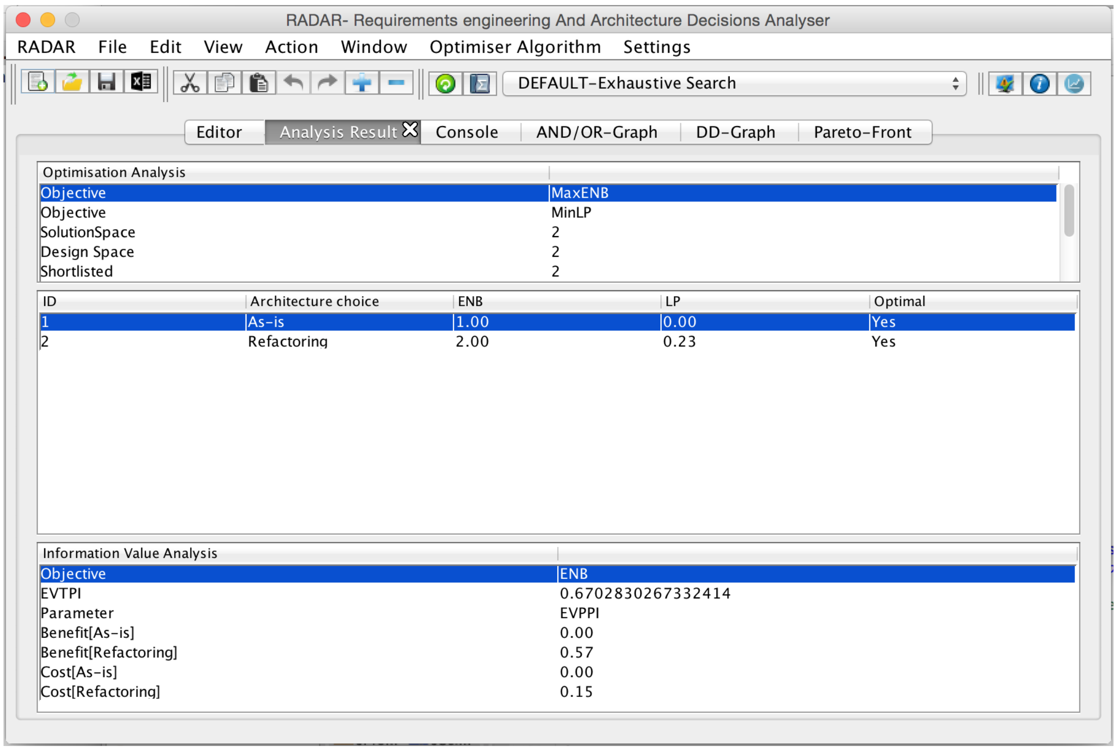
FIGURE 6.1: RADAR refactoring model example.

(Figure 6.6); and the Pareto-front, which shows the Pareto optimal solutions (Figure 6.7).

Another tab that is important during the analysis of a RADAR model is *analysis settings tab*. This tab, shown in Figure 6.8, is used to specify parameters for model analysis. Examples of these parameters include: (i) the number of Monte-Carlo simulation run which is default to 10^4; (ii) the objective variable, for example NB of the refactoring example, to be used for computing the expected value of total and partial perfect information (EVTPI and EVPPI); and the model variable (e.g., NB) for which the generation of the AND/OR goal graph starts from.

In Figure 6.9, RADAR provides modellers with an option to specify evolutionary algorithm parameters such as the population size, maximum number of fitness function evaluations, crossover, mutation probabilities and optimisation approach. RADAR validates these input parameters to ensure users enter valid values and displays an error message when the values are invalid.

RADAR Command-line Tools. The RADAR tool also provides command-line interface. The command **java -jar radar_console_app.jar - -help** will display a list


RADAR- Requirements engineering And Architecture Decisions Analyser
RADAR File Edit View Action Window Optimiser Algorithm Settings
Editor
Analysis Result
Console
AND/OR-Graph
DD-Graph
Pareto-Front

DEFAULT-Exhaustive Search

Optimisation Analysis	
Objective	MaxENB
Objective	MinLP
SolutionSpace	2
Design Space	2
Shortlisted	2

ID	Architecture choice	ENB	LP	Optimal
1	As-is	1.00	0.00	Yes
2	Refactoring	2.00	0.23	Yes

Information Value Analysis	
Objective	ENB
EVTPI	0.6702830267332414
Parameter	EVPPI
Benefit[As-is]	0.00
Benefit[Refactoring]	0.57
Cost[As-is]	0.00
Cost[Refactoring]	0.15

FIGURE 6.2: RADAR refactoring model analysis result.

of RADAR commands for decision analysis. These commands and their functions are enumerated below:

1. **- -EVPI:** computes EVTPI and EVPPI. Input to this command option is an objective name. An example usage of this command is **- -EVPI 'ENB'** (used by default).

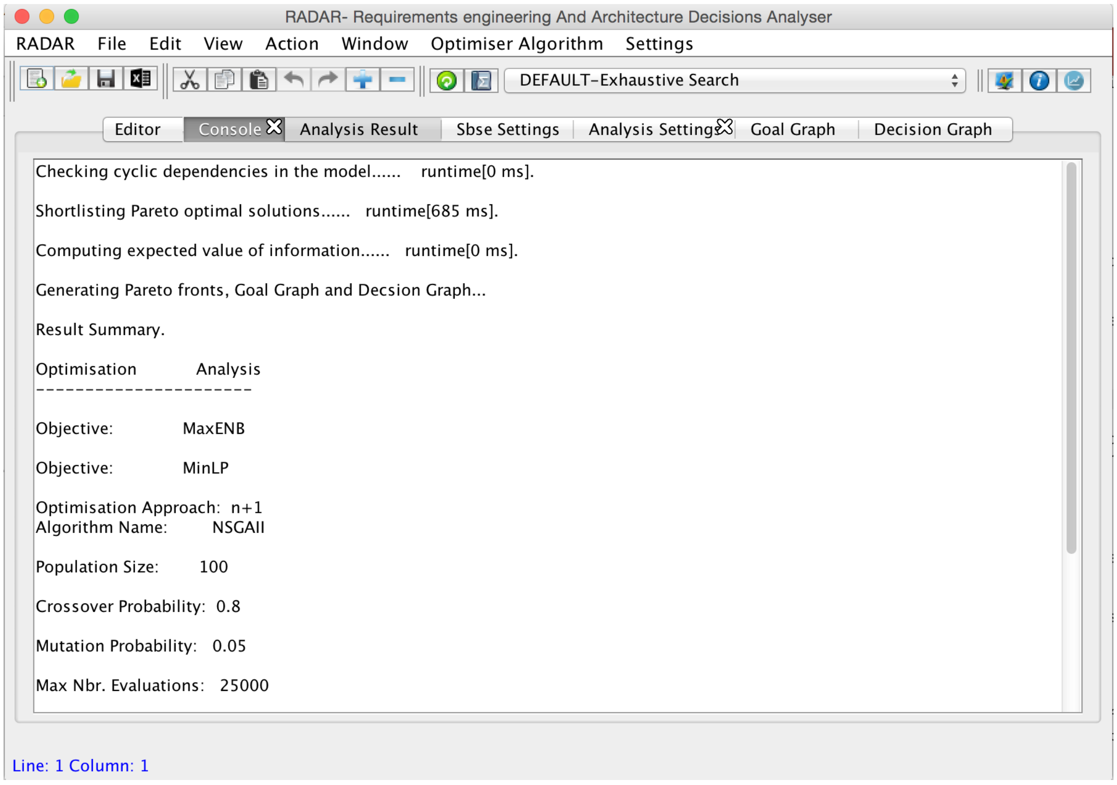
2. **- -model:** Specifies the path where the decision model is stored. Input to the command option is the file that contains the decision model <file path>.

3. **- -nbr_sim:** Number of Monte-carlo simulation run. Input to the command option is <sample size> (default: 10000).

4. **- -opt:** Optimisation approach for handling RADAR's constraints defined in a model. Example usage is **- -opt 'n+1'** (default: n+1).

5. **- -output:** Output folder where the results are saved. Input to the command option is <file path>.

6. **- -param:** Specify the approximate algorithm parameters. Inputs to this command option <population size> (datatype: integer), <crossover rate> (datatype:


RADAR- Requirements engineering And Architecture Decisions Analyser
RADAR File Edit View Action Window Optimiser Algorithm Settings
DEFAULT–Exhaustive Search
Editor Console Analysis Result Sbse Settings Analysis Setting Goal Graph Decision Graph
Line: 1 Column: 1

```
Checking cyclic dependencies in the model......    runtime[0 ms].

Shortlisting Pareto optimal solutions......   runtime[685 ms].

Computing expected value of information......   runtime[0 ms].

Generating Pareto fronts, Goal Graph and Decsion Graph...

Result Summary.

Optimisation          Analysis
----------------------

Objective:          MaxENB

Objective:          MinLP

Optimisation Approach:  n+1
Algorithm Name:        NSGAII

Population Size:      100

Crossover Probability:  0.8

Mutation Probability:   0.05

Max Nbr. Evaluations:   25000
```

Figure 6.3: RADAR refactoring model output log in a console tab.

double), <mutation rate> (datatype: double), <maximum evaluation> (datatype: integer) and <runs> (datatype: integer) in this order.

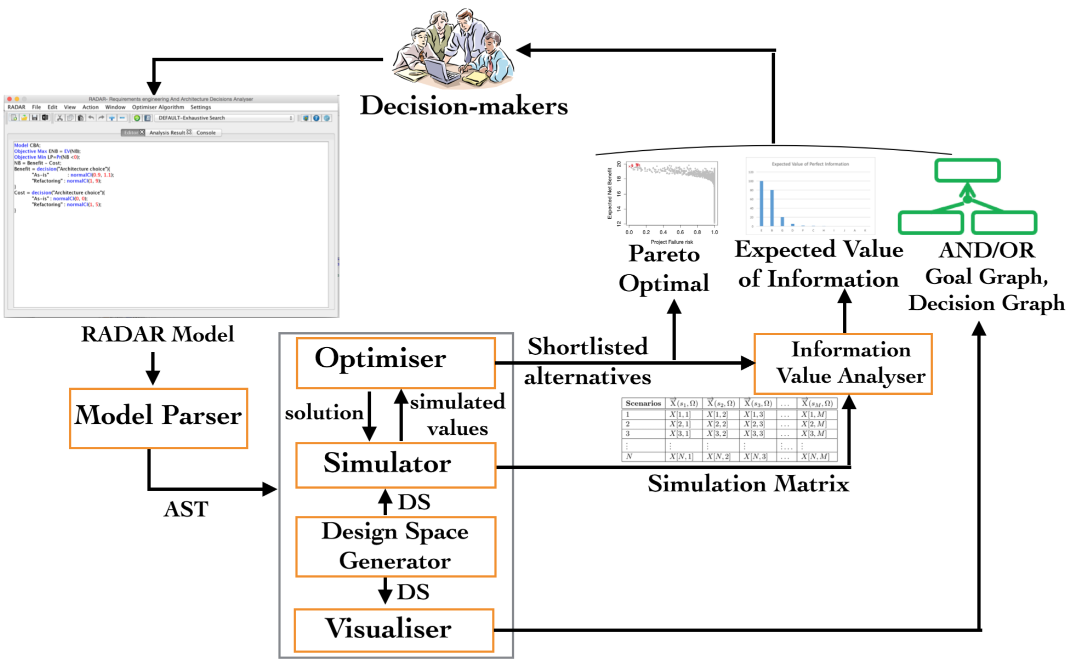
7. **- -param-default:** Tells the approximate algorithm to use default parameters for <population size> (100), <crossover rate> (0.8), <mutation rate> (1/options), <maximum evaluation> (1000) and <runs> (10).

8. **- -solve-using:** Solves the decision model. Input to this command option is <algorithm name> e.g. ExhaustiveSearch (default), NSGAII, SPEA2, MOCell, IBEA, RandomSearch. Specify 'SbseAlgs' to use all the EMOAs.

9. **- -subGraphObj:** Generates AND/OR sub-graph for the specified objective only. Input is an objective name. An example usage of this command is: **- - subGraphObjective 'InvestigationCost'**.

An example of a full command to analyse a decision model in RADAR is thus: **java -jar radar_console_app.jar - -model ../models/SAS.rdr - -param 100 0.8 0.1 50000 30 - -solve-using NSGAII - -EVPI 'ENB' - -opt 'n+1'** . In the above command:

- The argument after **- -model** specifies the path where the decision model is stored.


RADAR- Requirements engineering And Architecture Decisions Analyser
RADAR File Edit View Action Window Optimiser Algorithm Settings
DEFAULT-Exhaustive Search
Editor Analysis Result Console
Decision-makers
RADAR Model
Model Parser
AST
Optimiser
solution
simulated values
Simulator
DS
Design Space Generator
DS
Visualiser
Shortlisted alternatives
Pareto Optimal
Expected Value of Information
AND/OR Goal Graph, Decision Graph
Information Value Analyser
Simulation Matrix

```
Model CBA;
Objective Max ENB = EV(NB);
Objective Min LP=Pr(NB <0);
NB = Benefit - Cost;
Benefit = decision("Architecture choice"){
          "As-is"           : normalCI(0.9, 1.1);
          "Refactoring" : normalCI(1, 9);
}
Cost = decision("Architecture choice"){
          "As-is" : normalCI(0, 0);
          "Refactoring" : normalCI(1, 5);
}
```

Scenarios	$\vec{X}(s_1,\Omega)$	$\vec{X}(s_2,\Omega)$	$\vec{X}(s_3,\Omega)$	…	$\vec{X}(s_M,\Omega)$
1	$X[1,1]$	$X[1,2]$	$X[1,3]$	…	$X[1,M]$
2	$X[2,1]$	$X[2,2]$	$X[2,3]$	…	$X[2,M]$
3	$X[3,1]$	$X[3,2]$	$X[3,3]$	…	$X[3,M]$
⋮	⋮	⋮	⋮	⋮…	⋮
N	$X[N,1]$	$X[N,2]$	$X[N,3]$	…	$X[N,M]$

FIGURE 6.4: High-level system architecture of RADAR.

- The arguments after **- -param** specifies the SBSE parameters: 100 is the population size, 0.8 is the crossover rate, 0.1 is the mutation rate, 50000 is the maximum number of fitness evaluation, 30 is the number of independent runs.

- The argument after **- -solve-using** specifies the algorithm to be used in analysing the model. In this case NSGAII was specified. We can specify other algorithms like Exhaustive search, SPEA2, MOCell, IBEA, Random search.

- The argument **- -EVPI** specifies the objective for which to compute the expected value of information, i.e, Expected Net Benefit (ENB).

- The argument **- -opt** specifies the optimisation approach to be used in handling constraints. In this case, we used n+1 (i.e. lambda plus one approach).

6.2 RADAR Tool Design

RADAR's design follows an object-oriented architecture which encourages the addition of new system components and the re-use of existing components. The tool has a set of JAVA classes that a user can extend to implement new model constructs and decision analysis approaches. The tool also uses the state-of-the-art object-oriented java-based

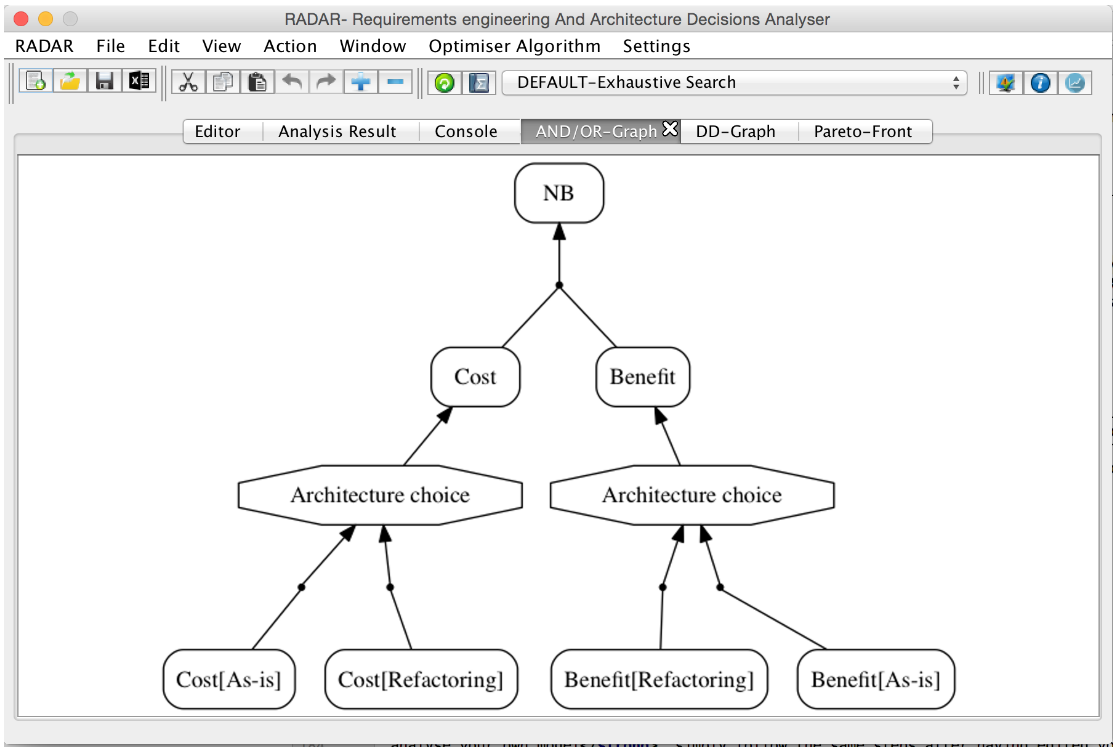


FIGURE 6.5: RADAR refactoring model AND/OR graph.

multi-objective optimisation framework with different implementations of meta-heuristic algorithms, such as the evolutionary algorithms discussed in Chapter 5.1.4.

To describe RADAR's tool design, we start by presenting the architecture of the tool, followed by the implementation details; grammar of RADAR's modelling language and its semantic model.

6.2.1 RADAR System Architecture

Figure 6.4 presents the high-level system architecture of the RADAR tool. The RADAR system is made up of six components namely: model parser, design space generator, simulator, optimiser, visualiser and information value analyser.

Model Parser. This takes a RADAR model as input and checks for syntax correctness. If the model is valid, the parser generates the model's abstract syntax tree, *AST*, and then populates the semantic model described in Section 6.2.4.

Design Space Generator. This component performs a single traversal on RADAR semantic model to generate all valid solutions to be used for the simulation and optimisation process.

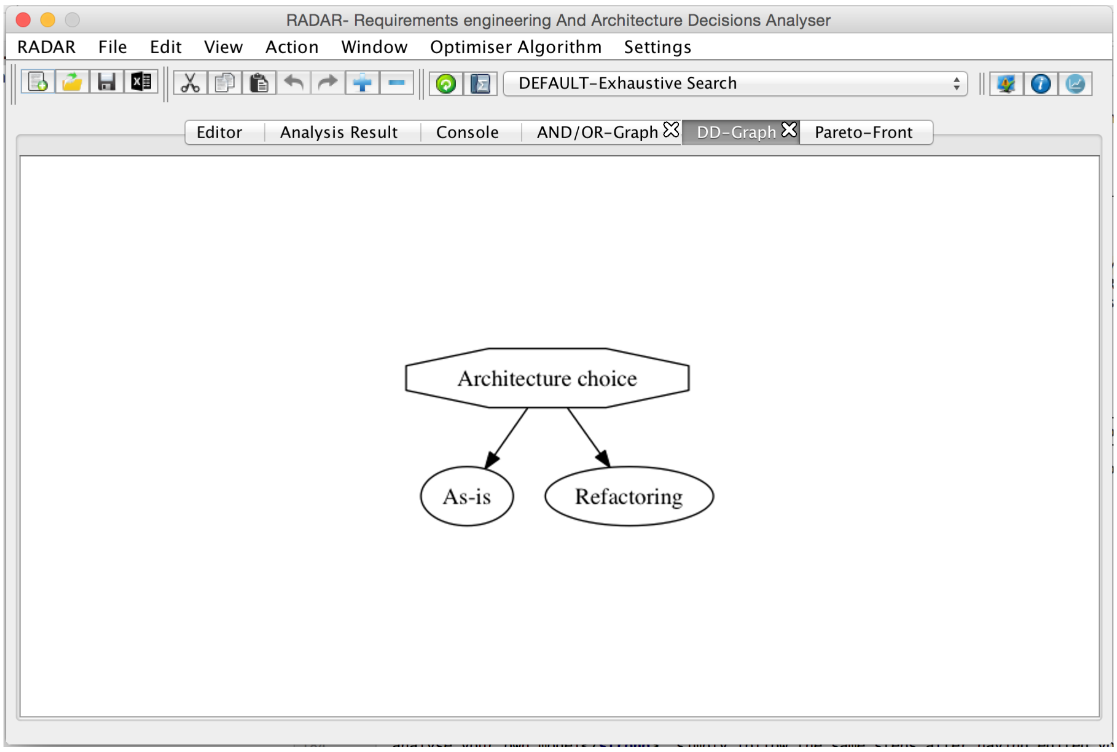

RADAR- Requirements engineering And Architecture Decisions Analyser
RADAR File Edit View Action Window Optimiser Algorithm Settings
DEFAULT–Exhaustive Search
Editor
Analysis Result
Console
AND/OR–Graph
DD–Graph
Pareto–Front
Architecture choice
As-is
Refactoring

FIGURE 6.6: RADAR refactoring model Decision graph.

Simulator. This component analyses model uncertainty through Monte-Carlo simulation (MCS) [113]. The simulator performs two main task: (i) given a candidate solution s, it returns simulation values of all objectives for s; (ii) given a set S of candidate solutions and a variable X, e.g., the ExpectedNetBenefit variable of the fraud detection example, it returns a simulation vector $\overline{X}$ that contains simulations of X and a simulation matrix $\overline{\overline{P}}$ that contains the simulations for all model parameters used to compute $\overline{X}$. The first function is used when shortlisting Pareto optimal solutions. The second function is used for Information Value Analysis. Internally, both functions use the same Monte-Carlo simulation where simulations of variables are generated by recursion through the AND/OR refinement equations by selecting the appropriate decision options through OR-refinements. The number N of simulations is set by the user and has a default value of 10^4. To ensure correctness in a given simulation run, all solutions are evaluated using the same parameter simulation data.

Optimiser. This component shortlists the Pareto-optimal solutions. It implements an exhaustive search strategy and alternative search-based approaches, such as evolutionary multi-objective optimisation algorithms [287].

Information Value Analyser. This component returns the expected value of total

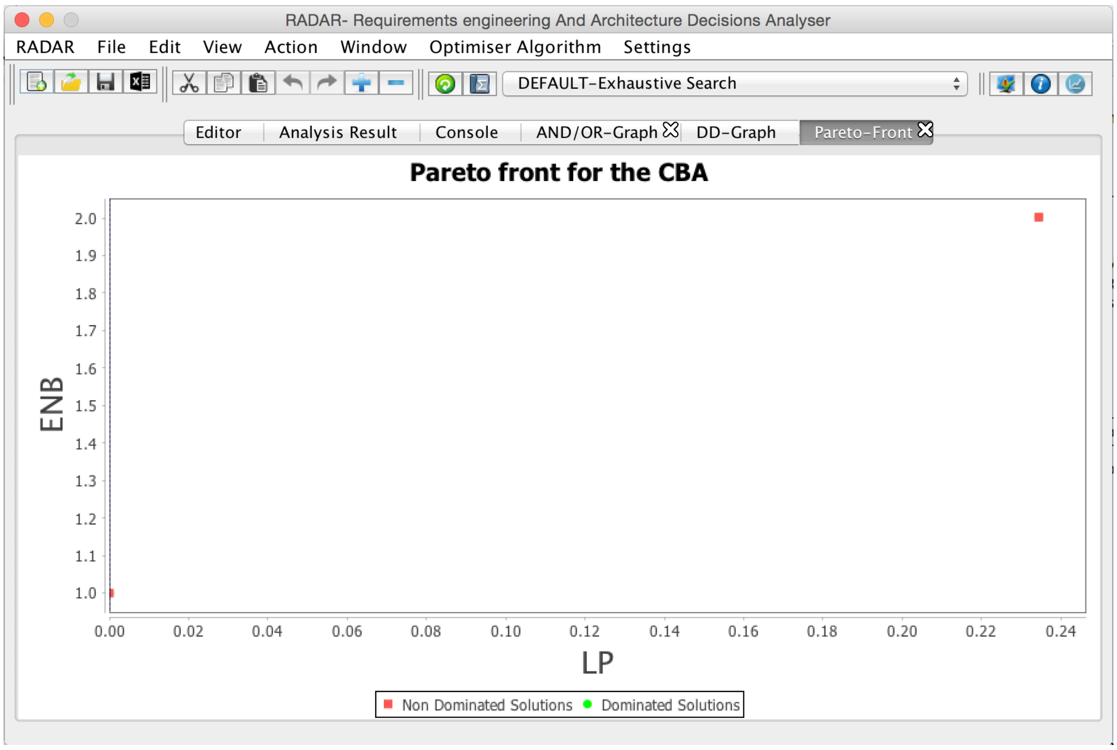


FIGURE 6.7: Pareto front for the RADAR refactoring model example.

and partial perfect information, i.e. EVTPI and EVPPI, for all non-deterministic model parameters with respect to a given objective, $\hat{X}$, and set of candidate solutions.

Visualiser. This component generates a visualisation of a RADAR model and the analysis results. Given RADAR's semantic model, the visualiser creates: (i) a variable dependency graph similar to the AND/OR goal refinement graph; and (ii) the decision dependency graph similar to feature diagrams in software product lines. This graph contains all decisions, their corresponding options and possible dependencies that exist between decisions.

For the model analysis results, the visualiser generates a tabular representation of the optimisation analysis and information value analysis results. Information about the optimisation analysis presented include: the optimisation objectives, size of the solutions space, size of design space, number of shortlisted solutions, number of model variables, number of model parameters, number of model decisions, the analysis run-time, and a tabular presentation of the shortlisted decisions and their corresponding options. For problems that have constraints, only the solutions without constraint violations are displayed with other solutions that violate constraints written to a file for further considerations. Information about the information value analysis presented are the EVTPI

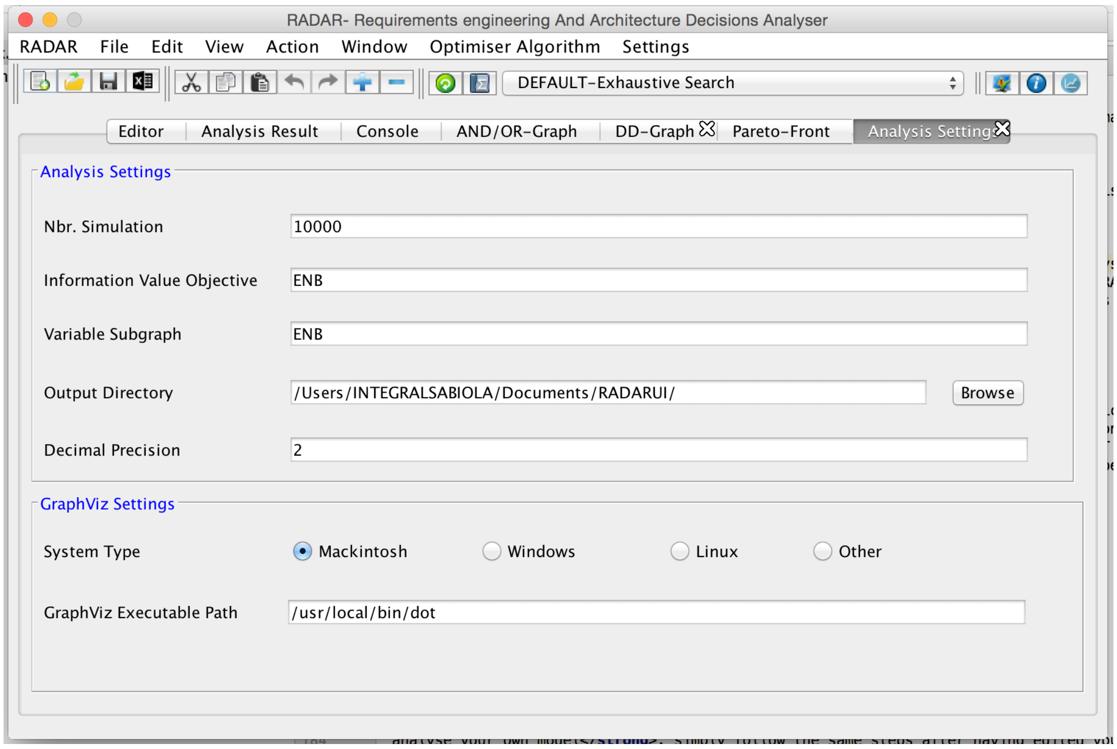
Context free grammar for the RADAR modelling language			
1	model	:	**Model** VAR **‘;’** NL* model-element*
2	model-element	:	objective-def+ | variable-def+ | constraint-def+
3	objective-def	:	**Objective (Max** | **Min)** VAR (= statistic)? **‘;’** NL*
4	statistic	:	**EV ‘(’** VAR **‘)’** | **Pr ‘(’** VAR **‘)’** | **Pr ‘(’** comparison **‘)’** | **percentile ‘(’** (+|-)? VAR **‘,’** NUMBER **‘)’**
5	variable-def	:	VAR **‘=’** variable-def NL*
6	variable-def	:	or-refinement | and-refinement | param-def
7	or-refinement	:	**decision ‘(’** decision-name **‘)’ ‘{’** option-name **‘:’** option-def **‘;’ ‘}’**
8		|	**decision-subset ‘(’ ‘*’** | **‘+’ ‘)** (’ decision-name **‘)’ ‘{’** option-name **‘:’** option-def **‘;’ ‘}’**
9	option-def	:	and-refinement | param-def
10	and-refinement	:	expr
11	constraint-def	:	**Constraint** constraint-arg (**‘requires’**|**‘excludes’**|**‘couples’**) constraint-arg **‘;’** NL*
12	constraint-arg	:	decision-name ‘:’ option-name
13	expr	:	**‘(’** expr **‘)’**
14		|	expr (‘ˆ’ | ‘/’ | **‘*’** | ‘+’ | **‘-’** ‘||’ | ‘&&’) expr
15		|	‘+’ | **‘-’** expr
16		|	expr **‘%’**
17		|	**‘!’** expr
18		|	comparison
19		|	NUMBER
20		|	VAR
21	param-def	:	distribution **‘(’** (expr (**‘,’** expr)* **‘)’**
22	comparison	:	VAR (‘>’ | ‘>=’ | ‘<’ | ‘<=’ | ‘==’ ‘! =’) expr
23	distribution	:	(**‘normal’**|**‘normalCI’**|**‘geometric’**|**‘exponential’**|**‘random’** |**‘triangular’**|**‘deterministic’**)
24	decision-name	:	STRING
25	option-name	:	STRING
26	STRING	:	‘”’ (‘”’ | ‘ ’ | .)*? ‘”’
27	VAR	:	(LETTER | ‘_‘) (LETTER | DIGIT | ‘_‘)*
28	LETTER	:	[**a-zA-Z**] NL*
29	NUMBER	:	DIGIT+ **‘.‘** DIGIT* | DIGIT+ | ‘.‘ DIGIT+
30	DIGIT	:	[**0-9**]

TABLE 6.1: RADAR grammar: Implemented using ANTLR 4 [198]: rule start (**:**), subrule (**(...)**), termination (**;**), alternative (|), optional (**?**), repetition-one or more (**+**), repetition-zero or more (*****) concatenation (**,**)

and EVPPI for all non-deterministic model parameters. The visualiser also generates a plot of the dominated and non-dominated (i.e. Pareto-optimal) solutions.

6.2.2 Implementation Details

RADAR is implemented entirely in JAVA. This is because JAVA has proven to be reliable, expressive, has rich libraries and above all platform independent. RADAR uses the ANTLR4 [197] tool to generate the model parser and traverse the abstract syntax tree to populate the semantic model. RADAR generates the AND/OR refinement graphs and decision dependency graphs in DOT format [97] that can be visualised and converted to other formats (e.g., GIF, PNG, SVG, PDF, or PostScript) using Graphviz [82]. The multi-objective evolutionary algorithms, such as NSGAII [69], SPEAII [289], MoCell [184] and IBEA [288], implemented in the tool's optimiser component uses JMetal5 [185]— a JAVA-based optimisation framework. The tool's GUI is implemented using Java Swing Windows builder. The open source RADAR tool and models discussed in the thesis can be downloaded from the tool's webpage (`https://ucl-badass.github.io/radar/`).


RADAR- Requirements engineering And Architecture Decisions Analyser
RADAR File Edit View Action Window Optimiser Algorithm Settings
DEFAULT-Exhaustive Search
Editor Analysis Result Console AND/OR-Graph DD-Graph Pareto-Front Analysis Settings
Browse

Analysis Settings

Nbr. Simulation	10000
Information Value Objective	ENB
Variable Subgraph	ENB
Output Directory	/Users/INTEGRALSABIOLA/Documents/RADARUI/
Decimal Precision	2

GraphViz Settings

System Type	[✓ Mackintosh] Windows Linux Other
GraphViz Executable Path	/usr/local/bin/dot

FIGURE 6.8: RADAR analysis settings page.

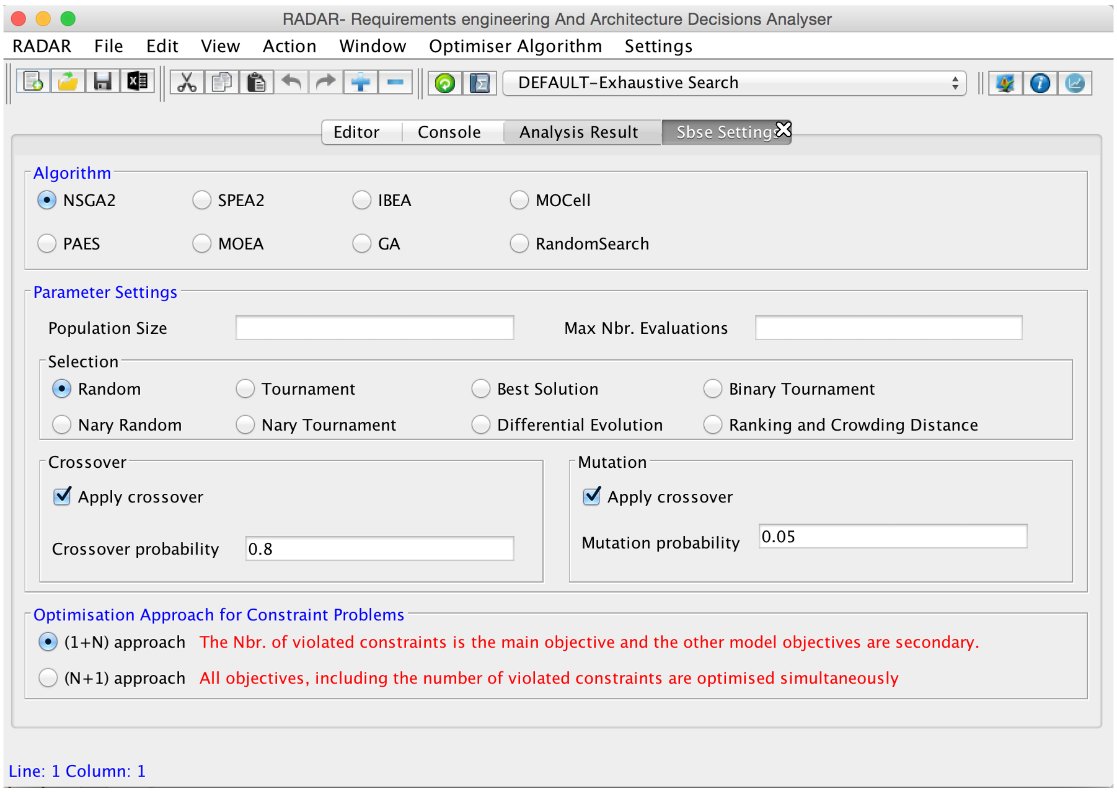
6.2.3 Language Grammar

RADAR is a domain specific language (DSL) [94]. One of the first and key steps in designing such language is to define grammar rules used by the language parser to convert the DSL to abstract syntax tree or parse tree. Table 6.1 presents the context free grammar of RADAR's modelling language. The grammar defines the exact syntax of RADAR's language and can be extended to cater for new language constructs. The grammar contains terminals, non-terminals, production rules[1] and a starting symbol.

Terminals are alphabet characters in form of strings. They are in bold or in single quote and cannot be changed by a production rule. Examples of terminals in Table 6.1 are the operators ('/', '*', '+', '>' and '=='), digits (**0-9**), alphabets (**a-zA-Z**) and the bold keywords (**Model**, **Objective**, **Max**, **Min** and **decision**).

Non-terminals are placeholder for sequence of terminal symbols. Such terminal symbols are created by recursively decomposing non-terminals using production rules until they are replaced by terminals only. Non-terminals are defined in terms of terminals and

[1]Production rules are set of rules for replacing non-terminal symbols (on the left side of the rule definition) with other terminal and non-terminal symbols (on the right side of the rule definition)


RADAR- Requirements engineering And Architecture Decisions Analyser

RADAR File Edit View Action Window Optimiser Algorithm Settings

DEFAULT–Exhaustive Search

Editor Console Analysis Result Sbse Setting

Algorithm

[✓ NSGA2] SPEA2 IBEA MOCell PAES MOEA GA RandomSearch

Parameter Settings

Population Size:

Max Nbr. Evaluations:

Selection

[✓ Random] Tournament Best Solution Binary Tournament Nary Random Nary Tournament Differential Evolution Ranking and Crowding Distance

Crossover

[✓ Apply crossover]

Crossover probability: 0.8

Mutation

[✓ Apply crossover]

Mutation probability: 0.05

Optimisation Approach for Constraint Problems

[✓ (1+N) approach] The Nbr. of violated constraints is the main objective and the other model objectives are secondary.

(N+1) approach All objectives, including the number of violated constraints are optimised simultaneously

Line: 1 Column: 1

FIGURE 6.9: RADAR SBSE parameter settings page.

other non terminals declared at some other place in the grammar. In table 6.1, observe how the grammar builds the non-terminal "model element" (row 2) by combining the objective definition (row 3: objective-def), variable definition (row 5: variable-def) and constraint definition (row 11: constraint-def). A model's objective(s) is declared as either a maximisation or a minimisation, whose statistic could be an expectation, probability, Boolean probability or percentile defined over a random variable (row 4). The variable definition can either be a OR-refinement (row 7), AND-refinement (row 10) and parameter estimation (row 21: param-def). The OR-refinement defines a variable in terms of AND-refinement or parameter estimations. The AND-refinement is an arithmetic or Boolean expression relating model variables (row 13). The parameter estimation is used to define a probability distribution (row 21: distribution) of a model variable. These distributions capture uncertainty in domain quantities and they include: the normal, normalCI, triangular, exponential and geometric distributions. A special type of non-terminal is the start symbol at the beginning of the grammar definition. For example, the non-terminal "model", which is defined in terms of **Model** and "model-element".

6.2.4 Semantic Model

RADAR's semantic model is a fundamental component and input in RADAR's automated decision analysis. It is an acyclic graph-based data structure that consists of different model elements, such as a set of objectives; a set model variables; model parameters; model decisions and their corresponding options and a set of model constraints.

Figure 6.10 depicts a UML diagram showing the semantic model (*Model*) class which is composed of a set of class objects, i.e. *Decision*, *QualityVariable*, *Parameter*, *Objective* and *Constraint*, that are populated by a *ModelConstructor* class when the language parser traverses the generated RADAR's abstract syntax tree (AST) and visits nodes (terminals and non-terminals) of the AST using the visitor design pattern.

The *Model* class also has a set of methods to (i) evaluate model objectives given a candidate RADAR solution (see Chapter 4.3 for the definition of a RADAR solution) through Monte-Carlo simulation; (ii) get candidate RADAR solutions for all model objectives; (iii) check non-existence of cyclic dependency between model variables; (iv) check dependency between model decisions given a set of candidate RADAR solutions; (v) generate AND/OR graph and decision dependency graph; (vi) get all non-deterministic model parameters for information value analysis; (vii) perform information value analysis (i.e. estimate EVTPI and EVPPI). Following the principle of data encapsulation, the *Model* class has "setter" and "getter" methods for adding and accessing class fields.

6.2.4.1 Decision

The *Decision* class has a label, i.e. the name of the decision; a field that captures the list of option tied to *Decision*; and a field to store the type of decision (i.e. exclusive-OR or inclusive-OR). Though not shown in Figure 6.10, the *Decision* class has a "setter" and "getter" methods for including and accessing each of the fields.

6.2.4.2 QualityVariable

The *QualityVariable* class has a label, i.e. the name of the variable, and has an *Expression* object that stores the definition of the *QualityVariable* object. Figure 6.11 is a UML diagram showing the *Expression* class. It is an abstract class extended by four class objects , i.e. *ArithmeticExpression*, *Distribution*, *AND_Refinement*, and *OR_Refinement*. The *Expression* class has four abstract methods implemented by its derived classes and each method traverses the calling *Expression* object instance recursively until reaching the leaf *Expression* in the *Model*. The abstract methods are (i) the "simulate" method which takes a RADAR solution (see Chapter 4.3) and returns an array of simulation values, where the array size is N, the number of possible scenarios default to 10^4; (ii) the "getAllSolutions" method that takes *Model* as input and returns a set of solutions; (iii) the "checkCyclicDependency" method that takes *Model* as input and checks for cyclic dependencies between variables within an expression and throws an exception if one is found; and (iv) an "accept" method that takes a *ModelVisitor* and *Model* as inputs and visits different model construct classes (e.g., *Distribution*, *And-Refinement*, *OR-Refinement* and *Objective*) to generate AND/OR goal models.

The *ArithmeticExpression* class is an abstract class which is extended by four classes, such as *UnaryExpression*, *BinaryExpression*, *QualityVariable* and *Number*. The *UnaryExpression* defines an expression with a *UnaryOperator* (e.g., percentage and negation). *BinaryExpression* defines an expression with a *BinaryOperator* (e.g., addition, subtraction, multiplication, division, logical AND, logical OR) and has a left and right operands that are *ArithmeticExpression* object instances. *QualityVariable* has an *Expression* object that stores its definition. Such definition points to an *Expression* object and can be a *Distribution*, *AND_Refinement* or *OR_Refinement*.

The *Distribution* class is an abstract class extended by different probability distribution classes. It has a *RandomGenerator* object that generates random numbers with the option of using a seed. It also has methods that generate an array of simulation values for each probability distribution class, such as the *TriangularDistribution*, *NormalDistribution*, *NormalCIDistribution*, *GeometricDistribution* and *ExponentialDistribution*.

The *AND_Refinement* class is a concrete implementation of the *Expression* class. It has an *Expression* object that stores the definition of the *AND_Refinement* object. Such definitions points to an *ArithmeticExpression*.

The *OR_Refinement* class is a concrete implementation of the *Expression* class. It has a *Decision* object that represents the model decision the *OR_Refinement* refers to; it has a Boolean field that is set to determine if the *OR_Refinement* is an exclusive-OR selection or inclusive-OR selection; and a field to store an arithmetic operator (addition or multiplication) if the *OR_Refinement* is an inclusive-OR. The *OR_Refinement* stores its definition in a data structure with a key-value pair, where the key is the name of the option of *Decision* object and the value is an *AND_Refinement* corresponding to a decision option.

6.2.4.3 Objective

The *Objective* class, shown in Figure 6.10, has a label, i.e. the name of the objective; a *QualityVariable* it refers to, an optimisation direction which can be either a maximisation (*Max*) or minimisation (*Min*); and a *Statistic* definition. *Statistic* is an abstract class that is extended by the different RADAR's statistical measures, such as *Expectation* (used by default), *Probability*, *BooleanProbability* and *Percentile*. The *Statistic* class has an abstract "evaluate" method, which takes a RADAR solution and a *Variable* that a particular *Objective* refers to as input and returns simulation objective value.

The *Objective* class has an overloaded "evaluate" method: one that takes a RADAR solution as input and the other takes a set of RADAR solutions as input. Internally, these two "evaluate" methods invoke the "evaluate" method of the *Statistic* class to return simulation objective value(s). The solution and corresponding simulation objective value are stored in a field *Value* –a map with "*Solution*" as the key and "objective value" as the value. *Objective* also has a method called "get All Solutions" that returns a set of solutions by traversing *Model* recursively starting from the *Variable* associated to the definition of *Objective* (see Chapter 5.1.1 for the algorithm description).

6.2.4.4 Constraint

The *Constraint* class in Figure 6.10 is an abstract class that has a left and right *ConstraintArgument* object (stores a decision name and option name) that defines a constraint. The *Constraint* class is extended by its derived classes, such as *RequireConstraint*, *ExcludeConstraint* and *CoupleConstraint*. These derived classes implement an *IConstraint* interface that has a method "isSolutionValid" which takes a candidate *Solution* and a *Model* as inputs and then checks that the solution does not violate the constraint definition.

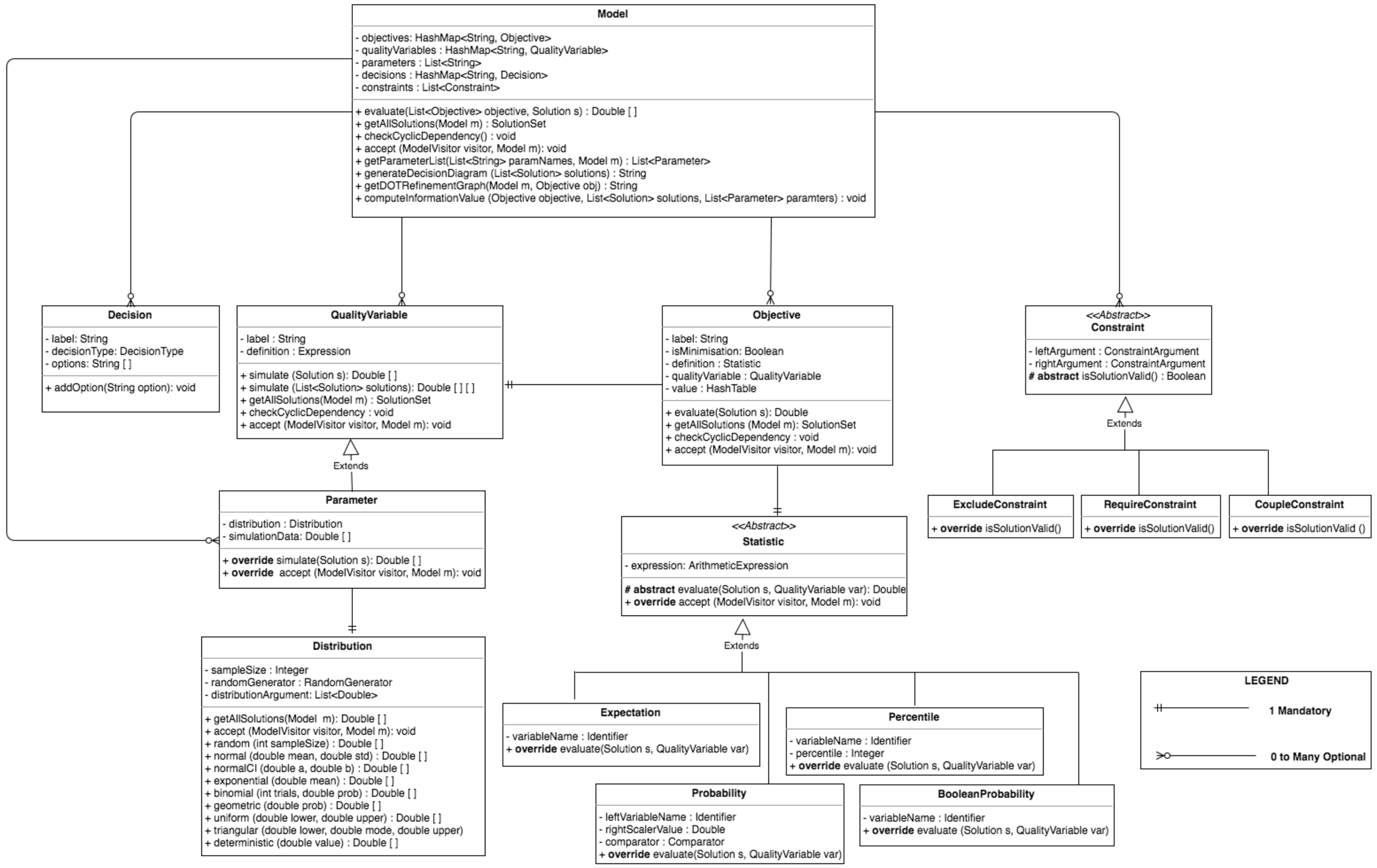
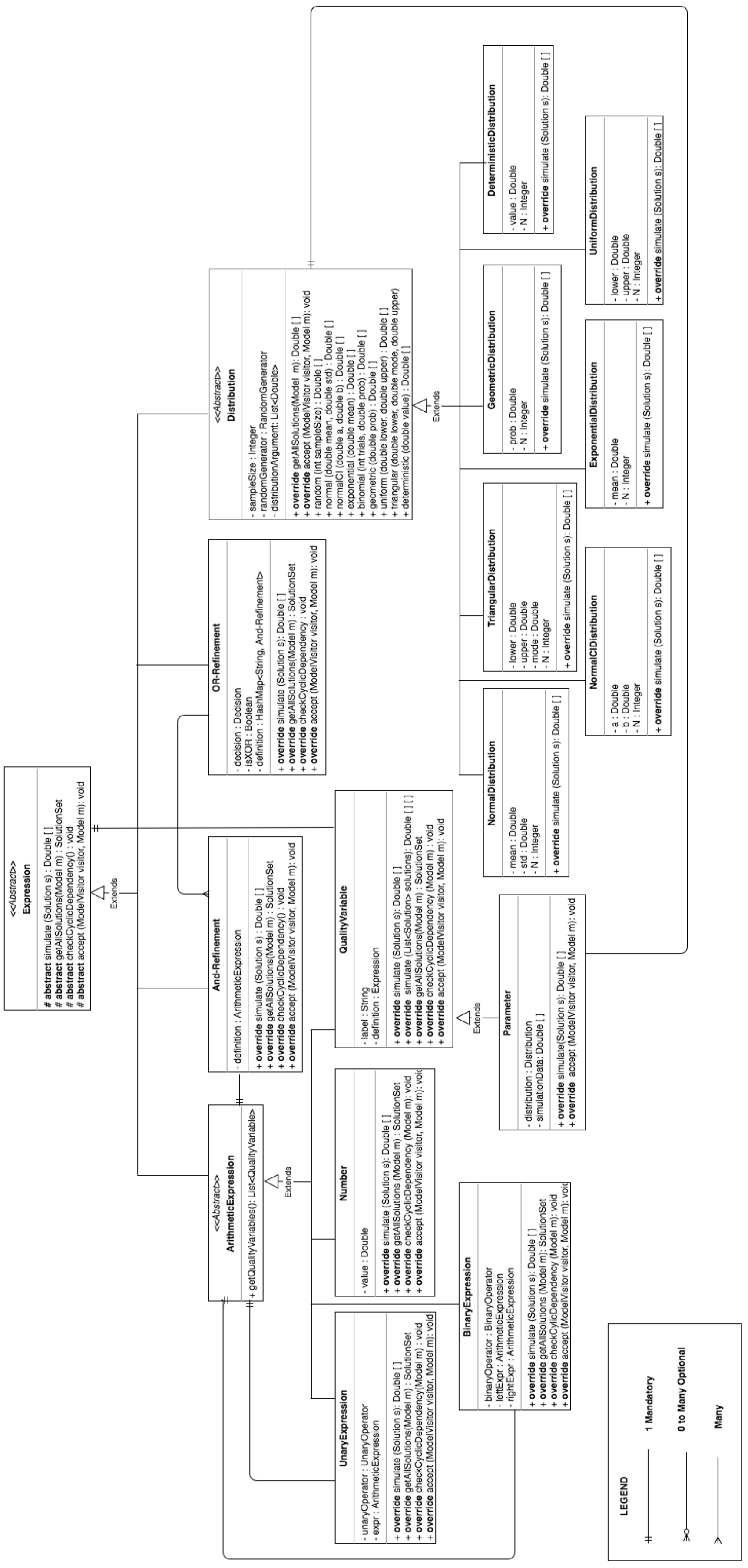

Model
- objectives: HashMap<String, Objective>
- qualityVariables : HashMap<String, QualityVariable>
- parameters : List<String>
- decisions : HashMap<String, Decision>
- constraints : List<Constraint>
+ evaluate(List<Objective> objective, Solution s) : Double [ ]
+ getAllSolutions(Model m) : SolutionSet
+ checkCyclicDependency() : void
+ accept (ModelVisitor visitor, Model m): void
+ getParameterList(List<String> paramNames, Model m) : List<Parameter>
+ generateDecisionDiagram (List<Solution> solutions) : String
+ getDOTRefinementGraph(Model m, Objective obj) : String
+ computeInformationValue (Objective objective, List<Solution> solutions, List<Parameter> paramters) : void
Decision
- label: String
- decisionType: DecisionType
- options: String [ ]
+ addOption(String option): void
QualityVariable
- label : String
- definition : Expression
+ simulate (Solution s): Double [ ]
+ simulate (List<Solution> solutions): Double [ ] [ ]
+ getAllSolutions(Model m) : SolutionSet
+ checkCyclicDependency : void
+ accept (ModelVisitor visitor, Model m): void
Objective
- label: String
- isMinimisation: Boolean
- definition : Statistic
- qualityVariable : QualityVariable
- value : HashTable
+ evaluate(Solution s): Double
+ getAllSolutions (Model m): SolutionSet
+ checkCyclicDependency : void
+ accept (ModelVisitor visitor, Model m): void
<<Abstract>>
Constraint
- leftArgument : ConstraintArgument
- rightArgument : ConstraintArgument
# abstract isSolutionValid() : Boolean
Extends
ExcludeConstraint
+ override isSolutionValid()
RequireConstraint
+ override isSolutionValid()
CoupleConstraint
+ override isSolutionValid ()
Extends
Parameter
- distribution : Distribution
- simulationData: Double [ ]
+ override simulate(Solution s): Double [ ]
+ override accept (ModelVisitor visitor, Model m): void
<<Abstract>>
Statistic
- expression: ArithmeticExpression
# abstract evaluate(Solution s, QualityVariable var): Double
+ override accept (ModelVisitor visitor, Model m): void
Distribution
- sampleSize : Integer
- randomGenerator : RandomGenerator
- distributionArgument: List<Double>
+ getAllSolutions(Model m): Double [ ]
+ accept (ModelVisitor visitor, Model m): void
+ random (int sampleSize) : Double [ ]
+ normal (double mean, double std) : Double [ ]
+ normalCI (double a, double b) : Double [ ]
+ exponential (double mean) : Double [ ]
+ binomial (int trials, double prob) : Double [ ]
+ geometric (double prob) : Double [ ]
+ uniform (double lower, double upper) : Double [ ]
+ triangular (double lower, double mode, double upper)
+ deterministic (double value) : Double [ ]
Extends
Expectation
- variableName : Identifier
+ override evaluate(Solution s, QualityVariable var)
Percentile
- variableName : Identifier
- percentile : Integer
+ override evaluate (Solution s, QualityVariable var)
Probability
- leftVariableName : Identifier
- rightScalerValue : Double
- comparator : Comparator
+ override evaluate(Solution s, QualityVariable var)
BooleanProbability
- variableName : Identifier
+ override evaluate (Solution s, QualityVariable var)
LEGEND
1 Mandatory
0 to Many Optional

FIGURE 6.10: UML diagram showing main classes of the semantic model.



<<Abstract>> Expression
# abstract simulate (Solution s) : Double [ ]
# abstract getAllSolutions(Model m) : SolutionSet
# abstract checkCyclicDependency() : void
# abstract accept (ModelVisitor visitor, Model m): void
Extends
<<Abstract>> ArithmeticExpression
+ getQualityVariables(): List<QualityVariable>
And-Refinement
- definition : ArithmeticExpression
+ override simulate (Solution s) : Double [ ]
+ override getAllSolutions(Model m) : SolutionSet
+ override checkCyclicDependency() : void
+ override accept (ModelVisitor visitor, Model m): void
OR-Refinement
- decision : Decision
- isXOR : Boolean
- definition : HashMap<String, And-Refinement>
+ override simulate (Solution s): Double [ ]
+ override getAllSolutions(Model m) : SolutionSet
+ override checkCyclicDependency : void
+ override accept (ModelVisitor visitor, Model m): void
<<Abstract>> Distribution
- sampleSize : Integer
- randomGenerator : RandomGenerator
- distributionArgument: List<Double>
+ override getAllSolutions(Model m): Double [ ]
+ override accept (ModelVisitor visitor, Model m): void
+ random (int sampleSize) : Double [ ]
+ normal (double mean, double std) : Double [ ]
+ normalCI (double a, double b) : Double [ ]
+ exponential (double mean) : Double [ ]
+ binomial (int trials, double prob) : Double [ ]
+ geometric (double prob) : Double [ ]
+ uniform (double lower, double upper) : Double [ ]
+ triangular (double lower, double mode, double upper)
+ deterministic (double value) : Double [ ]
QualityVariable
- label : String
- definition : Expression
+ override simulate (Solution s): Double [ ]
+ override simulate (List<Solution> solutions): Double [ ] [ ]
+ override getAllSolutions (Model m) : SolutionSet
+ override checkCyclicDependency (Model m) : void
+ override accept (ModelVisitor visitor, Model m): void
Parameter
- distribution : Distribution
- simulationData: Double [ ]
+ override simulate(Solution s): Double [ ]
+ override accept (ModelVisitor visitor, Model m): void
Number
- value : Double
+ override simulate (Solution s): Double [ ]
+ override getAllSolutions (Model m) : SolutionSet
+ override checkCyclicDependency (Model m): void
+ override accept (ModelVisitor visitor, Model m): void
UnaryExpression
- unaryOperator : UnaryOperator
- expr : ArithmeticExpression
+ override simulate (Solution s): Double [ ]
+ override getAllSolutions(Model m) : SolutionSet
+ override checkCyclicDependency(Model m) : void
+ override accept (ModelVisitor visitor, Model m): void
BinaryExpression
- binaryOperator : BinaryOperator
- leftExpr : ArithmeticExpression
- rightExpr : ArithmeticExpression
+ override simulate (Solution s): Double [ ]
+ override getAllSolutions (Model m): SolutionSet
+ override checkCyclicDependency (Model m): void
+ override accept (ModelVisitor visitor, Model m): void
NormalDistribution
- mean : Double
- std : Double
- N : Integer
+ override simulate (Solution s): Double [ ]
NormalCIDistribution
- a : Double
- b : Double
- N : Integer
+ override simulate (Solution s): Double [ ]
TriangularDistribution
- lower : Double
- upper : Double
- mode : Double
- N : Integer
+ override simulate (Solution s): Double [ ]
ExponentialDistribution
- mean : Double
- N : Integer
+ override simulate (Solution s): Double [ ]
GeometricDistribution
- prob : Double
- N : Integer
+ override simulate (Solution s): Double [ ]
UniformDistribution
- lower : Double
- upper : Double
- N : Integer
+ override simulate (Solution s): Double [ ]
DeterministicDistribution
- value : Double
- N : Integer
+ override simulate (Solution s): Double [ ]
LEGEND
1 Mandatory
0 to Many Optional
Many

Figure 6.11: UML diagram showing the base Expression class and its derived classes.

Chapter 7

Evaluating RADAR's Applicablity and Usefulness

7.1 Research Questions

This chapter demonstrates RADAR in action and evaluates its applicability and usefulness on a range of requirements and architecture decision problems. We thus aim to answer the following research questions:

RQ1 (Applicability): Is the RADAR tool applicable to real-world requirements and architectural decision problems? We use this research question to provide insight about the applicability of RADAR's modelling language and automated decision analysis technique on different requirements and architecture decision problems in different application domains.

RQ2 (Usefulness): Does RADAR's decision analysis technique provide useful improvements to real-world requirements and architecture decisions? This research question gives insight into the usefulness of the tool to guide decision-makers in making better decisions in the presence of uncertainty. The performance and scalability of RADAR will be evaluated in Chapter 8.

7.2 Experiments

To answer these questions, we have applied RADAR on 12 requirements and architecture decisions problems based on real design decisions problems described in the literature. These problems cover five important categories of requirements and architecture decisions problems:

1. **Decisions in Goal Models.** To evaluate RADAR's ability to analyse decisions in goal-oriented requirements engineering models, we have applied it to two problems that had previously been used to illustrate quantitative goal modelling techniques. The first is a financial fraud detection system [67, 74, 75]; the second is an ambulance dispatching system [90, 126].
2. **Decisions in Architecture Models.** We have also applied RADAR to two typical architecture decision problems. The first is the design of an emergency response system to coordinate the deployment of emergency response teams [85, 166]; the second is the NASA satellite processing system designed to collect and process satellite images [151, 179].
3. **Security Policy Decisions.** We have also applied RADAR to guide decisions about an organisation security policy regarding building access and sharing of electronic documents [50, 51].
4. **Next Release Problem (NRP).** We show how RADAR can be used to support decisions about what features to implement for the next release of a commercial release planning system and a word processor system [142].
5. **Feature selection in product line engineering.** We have also applied RADAR to a series of problems dealing with selecting an optimal set of features in a product family. These includes: the public Bike Sharing System [177, 250]; Drupal system —a PHP-based framework for web content management [228]; an e-commerce System [176]; Amazon Web Service (AWS) elastic compute cloud [98]; Berkeley Relational Database Management System [238].

Table 7.1 summarises these problems. They are characterised by different number of objectives; different number of decisions; different number of options per decision; different expressions (e.g., arithmetic, Boolean, AND-Refinements, OR-Refinements with single option selection and OR-refinement with multiple option selections); different number of constraints (e.g., excludes, requires and couples) between options of decisions; and different design space sizes between 6 and 2^{50} alternative designs. For each model, we assigned values to domain quantities and parameters as prescribed in previous works.

To show that RADAR's modelling language and automated decision analysis technique are applicable to real-world requirements and architecture decision problems, we have applied RADAR to five real-world examples, one from each problem category described in Section 7.2. The RADAR analysis of the other 7 problems can be found in appendix. We also compare RADAR's analysis results presented in this section with previous work where such problems have been analysed. The problems considered in this section are enumerated below:

1. The design of a financial fraud detection system [73, 75, 115].
2. The design of a system to coordinate the deployment of emergency response teams [85, 166, 173].
3. The decisions about system security policies on the leak of confidential information [50, 51].
4. Requirements subset selection for the future release of a commercial decision support system [142].
5. Optimal feature selection in a public Bike Sharing System [177, 250].

7.2.1 Plastic Card Fraud Detection System

7.2.1.1 Problem Statement

According to the financial fraud figures published in the first half of 2015 by the Financial Fraud Action UK (FFA-UK), financial fraud losses on cards in the UK totalled £250m, which represent 1% increase from 2014 [89]. The FFA-UK figures reveal increasing attempts of criminals employing sophisticated techniques to target cardholders despite the

Model	**Problem Description**	**Objectives**	**#Decisions (#XOR/#OR)**	**#Options Ranges**	**#Variables**	**#Constraints**	**Solution Space**
FDM	Design of a plastic card fraud detection system [73, 75, 115]	Max Fraud Detection Benefit Min Investigation Load	4/0	2-3	50	0	648
LAS	Cost-benefit analysis of automating the London Ambulance System [41, 90, 126]	Max Expected Net Benefit Min Risk	5/0	3-6	53	0	24
BSP	Security policy decisions on leaks of confidential information [50, 51]	Min Expected Cost of Disclosures Min Risk of Catastrophic Disclosures	2/0	2-3	34	0	6
BSS	Optimal feature selection of a public Bike Sharing System [177, 250]	Max Net Benefit Min Loss Probability	2/8	2-5	74	4	15×2^{20}
SAS	Software architecture evaluation of an emergency response system to coordinate teams in emergency situations [85, 166]	Max Net Benefit Min Risk	10/0	2-4	254	0	6912
ECS	Software architecture evaluation of a satellite Image Processor for collecting and managing satellite data [151, 179]	Max Project Utility Min Cost	10/0	2	86	0	2^{10}
MSM	Requirements subset selection for the future release of a Microsoft Word Processor [142]	Max Net Benefit Min Risk	0/1	50	251	39	2^{50}
RPM	Requirements subset selection for the future release of a commercial decision support tool for release planning [142]	Max Project Benefit Max Frequency Of Use Min Risk Min Stakeholder Dissatisfaction Min Requirements Volatilty	0/1	25	267	15	2^{25}
BDM	Optimal feature selection of the Berkeley Relational Database Management System [238]	Max Net Benefit Min Resource Utilisation	1/2	2-6	27	0	2^{9}
AWM	Amazon Web Service Elastic Compute Cloud optimal configuration [98]	Max Feature Richness Max Instance ECU Max EC2 Cores Max Instance RAM Max SSD Backed Min CostHour Min CostMonth	12/2	2-5	68	0	405×2^{10}
WPM	Optimal feature selection of a Web Portal System [176]	Max Feature Richness Max Feature Reuse Min Defect Count Min Cost	6/7	2-5	135	21	3×2^{26}
DPM	Optimal feature selection in Drupal— a PHP-based framework for web content management [228]	Max Net Benefit Min Risk	4/6	2-24	27	0	2^{40}

TABLE 7.1: RADAR application on real-world requirements and architecture decision problems with different complexities and different application domains. The model elaborations and their analysis are detailed in Chapter 7.

fraud detection methods deployed by banks and other financial institutions. Undoubtedly, the heterogeneous nature of card transactions on most accounts contributes greatly to the low accuracy and performance of most fraud detection methods [37].

Plastic card fraud detection systems (PCFDS) are deployed in banks to detect fraudulent transactions on plastic cards [115]. PCFDS process authorised card transactions and generate an alert when a transaction is suspected to be fraudulent so that further investigation can be carried out on the card . Card transactions are either processed in batches or real-time, and a card is blocked if the corresponding transaction performed on it is confirmed fraudulent.

In Chapter 1, we already introduced the plastic card fraud detection system, its optimisation objective and the design decisions. The optimisation of plastic card fraud detection systems typically include two conflicting objectives [115]: *minimise* the financial loss due to fraud and to *minimise* the fraud investigation costs. The design decisions include: the *transaction processing type* with the option of using continuous (real-time) or batch processing; the *fraud detection method* which can be two-class supervised classification method or a non-statistical rule-based method; If the classifier fraud detection method is chosen, the *alert threshold* needs to be decided on; and the *blocking policy* with the option of blocking an account once a fraud detection method flags a transaction as suspected fraud, or only blocking the account after the suspected fraud has been confirmed by human investigators.

7.2.1.2 RADAR Model

Modelling the Optimisation Objectives

The design of the plastic card fraud detection system has two key concerns, namely: *minimise* the financial loss due to fraud and to *minimise* the manual fraud investigation load.

```
**Objective** **Max** FraudDetectionBenefit= **EV**(Benefit)
**Objective** **Min** InvestigationLoad = **percentile**(NbrAlerts, 95)
Benefit = BaseLineFinancialLoss - FinancialLoss
FinancialLoss = NbrCompromisedAccounts × AverageFraudValue × NbrFraudPerAccountBe-
foreBlocked
BaseLineFinancialLoss = **deterministic**(500000)
AverageFraudValue = **normalCI**(100, 1000)
NbrFraudPerAccountBeforeBlocked = **decision**("blocking policy"){
    "block first" : NbrFraudBeforeDetection
    "investigate first" : NbrFraudBeforeDetection + NbrFraudDuringInvestigation
}
```

The first objective of minimising financial loss is equivalent to maximising the benefits of using a fraud detection system, where the benefits are defined as the reduction in financial loss with respect to the current system's baseline. The formulation of the objective *FraudDetectionBenefit* as a maximisation of expected benefits instead of minimisation of expected loss is more convenient for RADAR's information value analysis.

We model the second objective as minimising the alert investigation load defined here as the 95th percentile of the number of alerts generated by the fraud detection system:

```
**Objective** **Min** InvestigationLoad = **percentile**(NbrAlerts, 95);
```

The percentile means that in 95 days out of 100, the number of alerts will be below the investigation load.

Modelling Financial Loss

This plastic card fraud detection system analyses transactions after they have been authorised by the bank. Therefore, if the fraud detection system detects a transaction as fraudulent, the bank will still lose the fraudulent transaction amount (unless the bank can prove the fraud is due to negligence from the cardholder or vendor, a concern we will not consider in our model). The purpose of the fraud detection system is to block compromised card accounts as quickly as possible so as to prevent further fraud. Plastic card fraud detection systems are thus evaluated by their ability to minimise future financial losses.

We model the future financial loss as the product of the number of compromised accounts, the average number of fraudulent transactions that will be authorised on an account before it is blocked, and the average value of a fraudulent transaction:

```
FinancialLoss = NbrCompromisedAccounts × AverageFraudValue × NbrFraudPerAccountBe-
foreBlocked
```

The *FinancialLoss* is measured in £ per day, *NbrCompromisedAccounts* in number accounts per day, and *AverageFraudValue* in £.

The number of compromised accounts is the total number of accounts multiplied by the percentage of compromised accounts:

```
NbrCompromisedAccounts = NbrAccounts × CompromisedAccountRatio;
```

Both *NbrAccounts* and *CompromisedAccountRatio* are domain parameters that can be estimated from past data. For our example, we asume the following:

```
NbrAccounts = normalCI(0.9×10^6, 1.1×10^6);
CompromisedAccountRatio = triangular(0, 0.0001, 0.0003);
```

The average fraudulent transaction value can also be estimated from past data. For example:

```
AverageFraudValue = normalCI(100, 1000);
```

The average number of frauds on an account before it is blocked, *NbrFraudPerAccountBeforeBlocked*, depends on the blocking policy. If accounts are blocked as soon as the fraud detection system suspects a fraud, the number of fraud before the account is blocked is the number of fraud before detection. If accounts are blocked only after suspected frauds are confirmed by a fraud investigation, further frauds might occur during investigation:

```
NbrFraudPerAccountBeforeBlocked = decision("blocking policy"){
    "block first" : NbrFraudBeforeDetection;
    "investigate first" : NbrFraudBeforeDetection + NbrFraudDuringInvestigation;
}
```

The average number of frauds before detection, *NbrFraudBeforeDetection*, depends on the processing type. For continuous processing, the mean number of fraud before detection is the infinite series:

```
1 * probability(fraud is detected after 1 fraudulent transactions)
+ 2 * probability(fraud is detected after 2 fraudulent transactions)
+ 3 * probability(fraud is detected after 3 fraudulent transactions)
+ ...
```

The probability that a fraudulent transaction is detected is the true alert rate (the ratio of the number of detected fraud over the number of fraud). Factoring the above series yield that for continuous processing:

```
NbrFraudBeforeDetection = 1/TrueAlertRate;
```

In batch processing, transactions are analysed at the end of every day. Batch processing thus introduces a delay between a fraudulent transaction and its detections, a delay during which additional fraud might occur, but because transactions are analysed in groups rather than individually, the batch processing may have a better true alert rate than the continuous processing. Assuming that batch processing adds on average a delay of a day to fraud detection, our models assumes that:

```
NbrFraudBeforeDetection = NbrFraudsPerCompromisedAccountPerDay × 1/BatchTrueAlertRate;
```

Thus, the final equation for estimating *NbrFraudBeforeDetection* is:

```
NbrFraudBeforeDetection = decision("processing type") {
    "continuous" : 1/ContinuousTrueAlertRate;
    "batch" : NbrFraudPerCompromisedAccountPerDay/BatchTrueAlertRate;
}
```

The average number of frauds per day per compromised account is a domain parameter that could be estimated from past data. For example:

```
NbrFraudsPerCompromisedAccountPerDay = normalCI(1, 20);
```

The average number of fraud per account committed during the investigation period is also proportional to the number of frauds per compromised account per day.

```
NbrFraudDuringInvestigation = NbrFraudPerCompromisedAccountPerDay×InvestigationDelay;
InvestigationDelay = triangular(1/24, 1/3, 1);
```

The true alerts rates depend on the fraud detection methods and their parameters. True alerts rates are typically estimated by analysing the performance of the fraud detection method on past data.

To keep the model simple, we assume the classifier has three settings high, medium, and low that generates high, medium, or low number of alerts; and the rule-based approach has a single fixed true alert rate.

For the continuous true alert rate:

```
ContinuousTrueAlertRate = decision(“fraud detection method”){
    “classifier” : ContinuousAlertThreshold;
    “rule-based” : deterministic(0.75);
}
ContinuousAlertThreshold = decision(“alert threshold”){
    “low” : triangular(0.75, 0.85, 0.95);
    “medium” : triangular(0.65, 0.75, 0.85);
    “high” : triangular(0.55, 0.65, 0.75);
}
```

For the batch true alert rate:

```
BatchTrueAlertRate = decision(“fraud detection method”){
    “classifier” : BatchAlertThreshold;
    “rule-based” : deterministic(0.80);
}
BatchAlertThreshold = decision(“alert threshold”){
    “low” : triangular(0.75, 0.85, 0.95);
    “medium” : triangular(0.65, 0.75, 0.85);
    “high” : triangular(0.55, 0.65, 0.75);
}
```

Modelling Fraud Investigation Load

The number of generated alerts is the sum of the number of true alerts and false alerts:

```
NbrAlerts = NbrTrueAlerts + NbrFalseAlerts;
```

The number of true and false alerts are functions of the number of accounts, the percentage of compromised accounts, and the true and false alert rates:

```
NbrTrueAlerts = NbrFraud * TrueAlertRate;
NbrFalseAlerts = NbrLegitTransactions * (1 – TrueNegativeRate);
```

The true alert rate (the ratio of the number true alert over the number of fraud, a.k.a. sensitivity) and true negative rate (the ratio of the number of un-flagged legitimate transactions over the total number of legitimate transactions, a.k.a. specificity) vary with the processing type:

```
TrueAlertRate = decision("processing type"){
    "continuous" : ContinuousTrueAlertRate;
    "batch" : BatchTrueAlertRate;
}
TrueNegativeRate = decision("processing type"){
    "continuous" : ContinuousTrueNegativeRate;
    "batch" : BatchTrueNegativeRate;
}
```

Models of the continuous and batch true alert rates have already been defined. The models for the continuous and batch true negative rate follow the same structure:

```
ContinuousTrueNegativeRate = decision("fraud detection method"){
    "classifier" : ContinuousClassierTrueNegativeRate;
    "rule-based" : deterministic(0,99);
}
ContinuousClassierTrueNegativeRate = decision("continuous classifier threshold level"){
    "low" : triangular(0.95, 0.99, 0.995);
    "medium" : triangular(0.99, 0.995, 0.999);
    "high" : triangular(0.995, 0.999, 0.9999);
}
BatchTrueNegativeRate = decision("fraud detection method"){
    "classifier" : BatchClassifierTrueNegativeRate;
    "rule-based" : deterministic(0,995);
}
BatchClassifierTrueNegativeRate = decision("batch classifier threshold level"){
    "low" : triangular(0.9, 0.99, 0.995);
    "medium" : triangular(0.99, 0.995, 0.999);
    "high" : triangular(0.995, 0.999, 0.9999);
}
```

Finally, the number of fraudulent and legitimate transactions depends on the number of accounts and average number of transactions per account:

```
NbrFraud = NbrAccounts * CompromisedAccountRatio * NbrFraudPerCompromisedAccount-
PerDay;
NbrLegitTransactions = NbrAccounts * NbrLegitTransactionsPerAccountPerDay;
```

The average number of accounts and compromised account ratio are model parameters that we have already estimated above. The average number of legitimate and fraudulent transactions per compromised account per day are also model parameters that could be estimated from past data. For example:

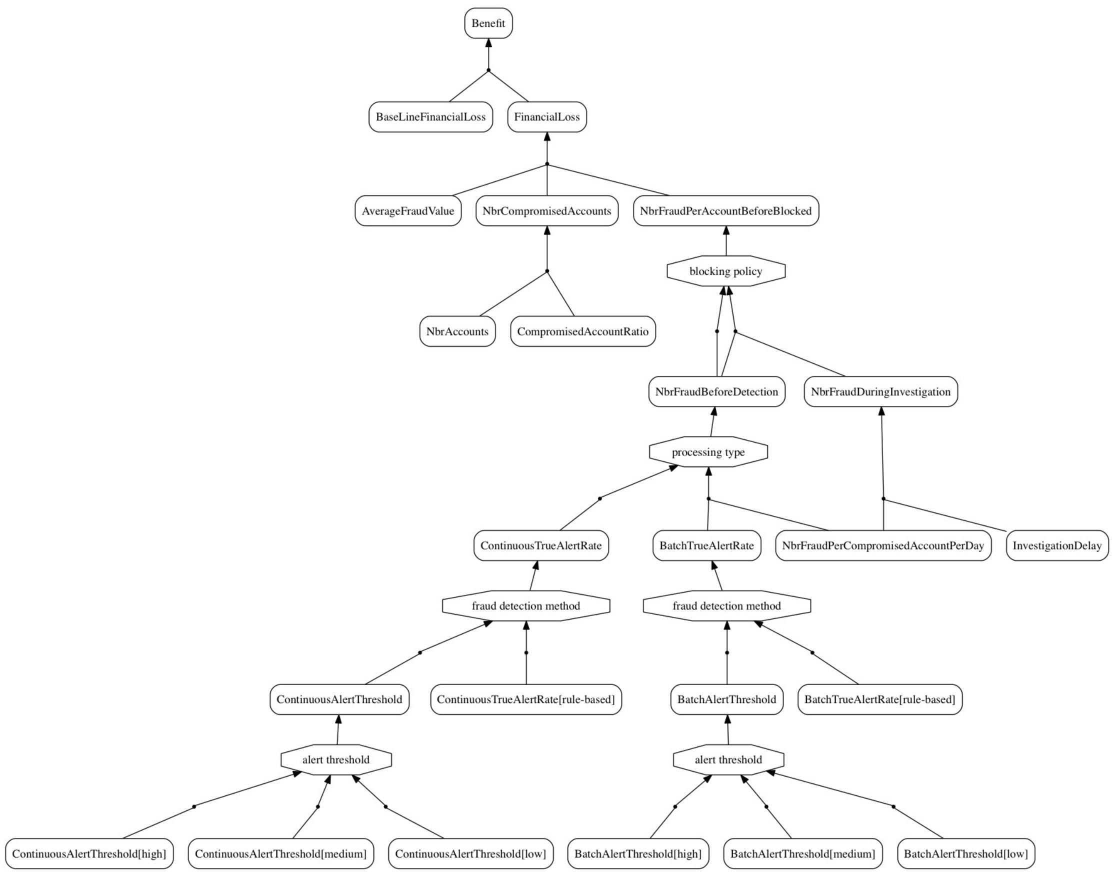


FIGURE 7.1: AND/OR refinement graph for the model variable Benefit.

```
NbrFraudPerCompromisedAccountPerDay = triangular(0, 3, 10);

NbrLegitTransactionsPerAccountPerDay = triangular(0, 3, 10);
```

To help visualise the model structure, RADAR automatically generates the AND/OR refinement graph and decision dependency graphs from the fraud detection model equations. Fig. 7.1 shows the partial AND/OR refinement graphs for the fraud detection model starting from the model variable Benefit. Fig. 4.3 in Chapter 3 shows the decisions; their corresponding options; and the dependency between a decision (threshold level) and an option of another decision (classifier).

7.2.1.3 Analysis Results

The RADAR analysis of the plastic card fraud detection model is presented in Table 7.2, which shows the results of the optimisation and information value analysis on the model.

The first part of Table 7.2 is the optimisation analysis results, which shows that all shortlisted solutions include the "block first" policy and "continuous processing" type. This means that, in our model, these two options outperform the "investigate first" policy and "batch processing" on both objectives. However, once these two options are selected, the shortlist includes all possible combinations of fraud detection methods and alert thresholds; each combination representing different tradeoffs between maximising fraud detection benefit and minimising investigation load. To visualise such tradeoffs, RADAR generates the graph in Fig. 7.2, plotting the objective values for the shortlised solutions (shown squares at the top of the graph) and all other non shortlisted ones (shown as circles).

The second part of Table 7.2 is the information value analysis results, which show that the EVTPI for this problem is 220 and EVPPI for AverageFraudValue is 122. All other parameters have EVTPI below 2. This means that in this model, the only parameter worth investigating further before deciding between the shortlisted solutions is average fraud value.

7.2.1.4 Comparison To Previous Analysis Approaches

Previous analyses on (plastic card) fraud detection systems have mainly focused on developing various fraud detection methods and tools. Some of the widely used methods include: (i) Expert Systems that represent domain-specific knowledge in order to solve detection problem by encoding fraud attacks as IF-THEN rules [15]; (ii) Artificial Neural Networks which uses previously observed fraud patterns to detect future unseen abnormal transactions [102]; (iii) Model-Based Reasoning technique in which frauds are detected through observable attack activities using a database of attacks scenarios [99]; (iv) Data mining techniques where learning algorithms are used to construct detection models from a large audit of transaction data [244]; (v) State Transition Analysis techniques where attacks are modelled as series of state transactions of the software systems. Fraud scenarios are represented by state transition diagram and actions that precedes or initiates a fraud represent transitions between states [138]; (vi) Genetic Algorithms technique which separates legitimate and fraudulent transactions using mathematical models whose candidate solutions represent possible fraud scenarios [54].

Optimisation Analysis

Objective:	**Max** FraudDetectionBenefit
Objective:	**Min** InvestigationLoad
SolutionSpace:	24
Minimal SolutionSet:	16
Shortlisted:	4
Nbr. Variables:	31
Nbr. Parameters:	19
Nbr. Decisions:	4
Runtime(s) :	0

ID	blocking policy	processing type	fraud detection method	alert threshold	FraudDetectionBenefit	InvestigationLoad
1	block first	continuous	rule-based	—	402799	82709
2	block first	continuous	classifier	medium	402516	52139
3	block first	continuous	classifier	high	387394	22467
4	block first	continuous	classifier	low	414087	232479

Information Value Analysis

Objective:	**Max** ENB
EVTPI:	220

Parameter	EVPPI
AverageFraudValue	122
ContinuousAlertThreshold[medium]	1.56
ContinuousAlertThreshold[low]	0
ContinuousClassierTrueNegativeRate[low]	0
ContinuousAlertThreshold[high]	1
BatchClassifierTrueNegativeRate[low]	0
BatchClassifierTrueNegativeRate[medium]	0
NbrAccounts	0
BatchAlertThreshold[medium]	0
NbrFraudPerCompromisedAccountPerDay	0
BatchAlertThreshold[low]	0
CompromisedAccountRatio	0
BatchAlertThreshold[high]	0
ContinuousClassierTrueNegativeRate[high]	0
NbrLegitTransactionsPerAccountPerDay	0
BatchClassifierTrueNegativeRate[high]	0
ContinuousClassierTrueNegativeRate[medium]	0
InvestigationDelay	0

Table 7.2: Optimisation analysis and information value analysis results for the plastic card fraud detection model.

Recently, Hand et al. [116] proposed a mathematical model for defining optimisation objectives and introduced performance evaluation criteria for plastic card fraud detection systems. Duboc et al. [67, 74, 75] proposed a goal-oriented requirements engineering technique to analyse the scalability of a commercial financial fraud detection system, i.e., Searchspace's Intelligent Enterprise Framework (IEF). The authors used KAOS

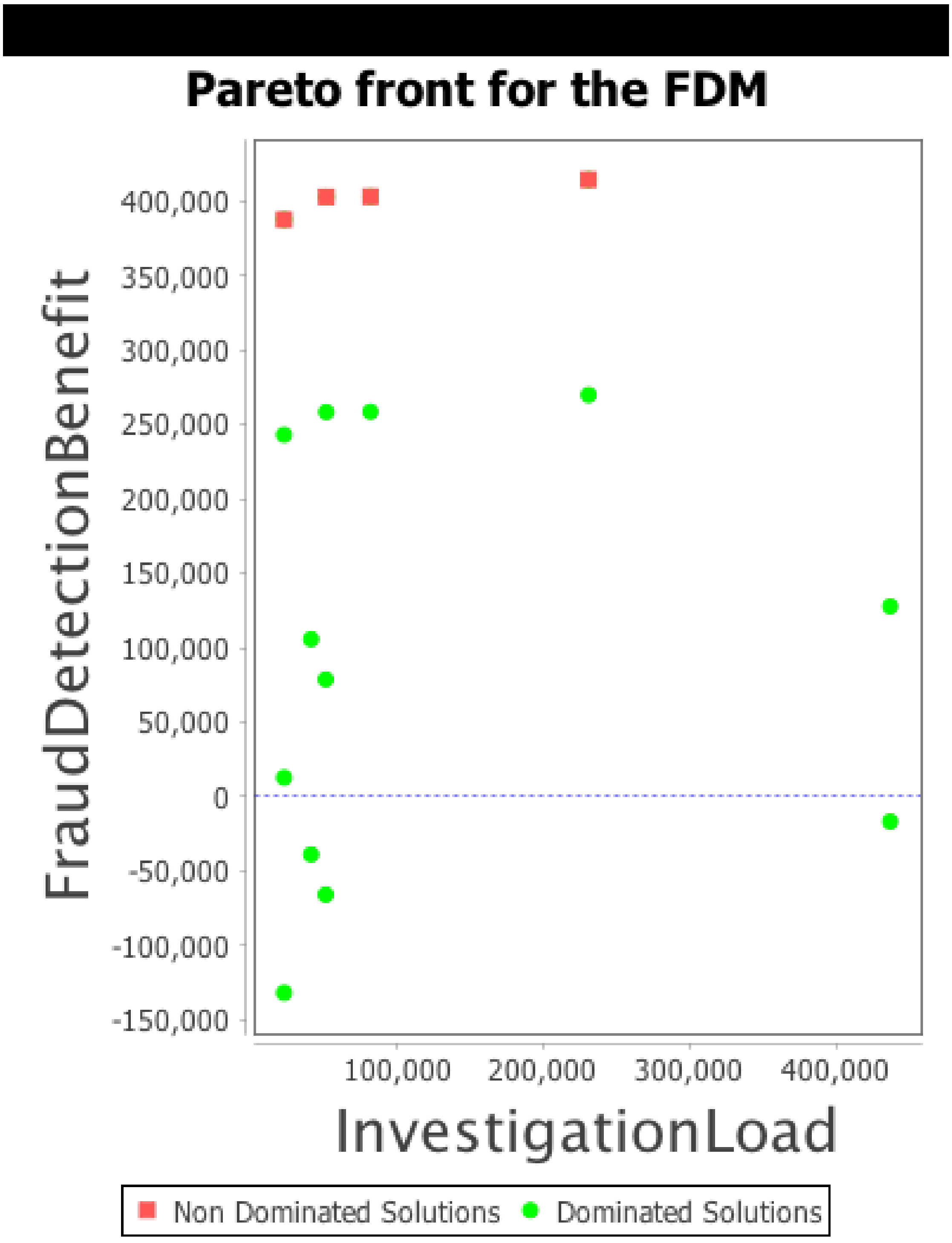


FIGURE 7.2: Pareto front of the the plastic card fraud detection model analysis.

approach to elaborate the plastic card fraud detection model and to capture conflicts between system goals. They defined the degrees of satisfaction of each goal using objective functions defined in terms of quality variables, which are random variables, defined over probability space. Each goal was specified using a natural language definition and a formal specification in linear temporal logic. Finally, alternative designs are selected using utility functions defined based on stakeholders' preferences.

In this section, we have defined our plastic card fraud detection model following the quantitative goal models presented by Duboc et al. [67, 75]. While Duboc's approach focused on analysing scalability of the fraud detection system, our approach seeks to aid stakeholders in selecting the best alternative fraud detection system designs that gives the best trade-off such that the optimisation objectives (i.e., maximising the fraud detection benefit and minimsing the investigation load resulting from investigating a fraud) are satisfied. In addition, the RADAR approach allows modellers to define different model objectives and elaborate the objective definition using simple mathematical equations. Unlike Duboc's approach, RADAR provides automated tool support for decision analysis and allows capturing uncertainties in domain quantities, a feature not included in Duboc's approach and analysis.

7.2.2 Emergency Response System

7.2.2.1 Problem Statement

The Emergency Response System, which is also known as the Situation Awareness System (SAS), was originally introduced by Naeem et al. [85], in their research presented at the International Conference on Software Engineering (ICSE) in 2013, on the early architecture selection problem under uncertainty. The SAS is a mobile software application originally developed to deploy emergency staff in cases of emergencies, and allows deployed individuals to share information about the status of an emergency situation.

Due to the proliferation of mobile technologies, standards, and platforms, the SAS stakeholders decided to improve the system using some of the latest technologies. An example of such improvements includes allowing deployed personnel to send and receive real-time information about the status of an emergency situation (e.g., interactive overlay on maps) and coordinate with one another (e.g., send reports, chat, and share video streams).

The SAS project team, which consisted of academics and engineers from a government agency, identified and described the design decisions and their corresponding alternative options as presented in Table 7.3. These decisions impacts the system's response time,

Decisions	**Alternative Options**
Location Finding	GPS
	Radio Triangulation
File Sharing Package	OpenIntents
	In house
Report Synchronisation	Explicit
	Implicit
Chat Protocol	XMPP (Open Fire)
	In house
Map Access	On demand (Google)
	Cached on Server
	Preloaded (ESRI)
Hardware Platform	Nexus 1 (HTC)
	Droid (Motorola)
Connectivity	Wi-Fi
	3G on Nexus 1
	3G on Droid
	Bluetooth
Database	MySQL
	sqlLite
Architectural Pattern	Facade
	Peer-to-peer
	Push-based
Data Exchange format	XML
	Compressed XML
	Unformatted data

Optimisation Objectives
Battery Usage
Response Time
Reliability
Ramp up Time
Cost
Development Time
Deployment Time

TABLE 7.3: Overview of SAS Design decisions and optimisation goals [84]

reliability, ramp up time, cost, development time and deployment time, which ultimately impacts the utility derived when the application delivers the intended goals and total cost incurred in improving the SAS application. The team is faced with the decision of selecting design options to be implemented in the new system that gives maximum net benefit to the stakeholders at a reduced risk.

7.2.2.2 RADAR Model

Modelling the Optimisation Objectives

The primary decision objectives of the SAS model include two objectives: maximise the expected net benefit and minimise the risk associated to each alternative architecture:

Objective **Max** ENB = **EV**(NB);

Objective **Min** Risk = **EV**(ProjectRisk);

Modelling the Net Benefit

The Net Benefit, NB, derived from each alternative architecture is defined as:

NB = BatteryUsageWeight × BatteryUsagePreference +
    ResponseTimeWeight × ResponseTimePreference +
    ReliabilityWeight × ReliabilityPreference +
    RampUpTimeWeight × RampUpTimePreference +
    CostWeight × CostPreference +
    DevelopmentTimeWeight × DevelopmentTimePreference +
    DeploymentTimeWeight × DeploymentTimePreference ;

BatteryUsageWeight = **deterministic**(9);

ResponseTimeWeight = **deterministic**(7);

ReliabilityWeight = **deterministic**(3);

RampUpTimeWeight = **deterministic**(2);

CostWeight = **deterministic**(1);

DevelopmentTimeWeight = **deterministic**(2);

DeploymentTimeWeight = **deterministic**(2);

Modelling the Project Risk

ProjectRisk = 1 − ((1 − BatteryUsageRisk) ×

(1 − ResponseTimeRisk) ×

(1 − ReliabilityRisk) ×

(1 − RampUpTimeRisk) ×

(1 − CostRisk) ×

(1 − DevelopmentTimeRisk) ×

(1 − DeploymentTimeRisk)

) ;

Modelling the Battery Usage

```
BatteryUsagePreference = (BatteryUsage - BatteryUsageWorst)/
                         (BatteryUsageBest - BatteryUsageWorst);
BatteryUsageRisk = Pr(BatteryUsage < BatteryUsageMust);
BatteryUsageBest = deterministic(24) ;
BatteryUsageWorst = deterministic(111) ;
BatteryUsageMust = deterministic(52);
BatteryUsage = BatteryUsageLocationFinding +
        BatteryUsageFileSharing +
        BatteryUsageReportSyncing +
        BatteryUsageChatProtocol +
        BatteryUsageMapAccess +
        BatteryUsageHardwarePlatform +
        BatteryUsageConnectivity +
        BatteryUsageDataBase +
        BatteryUsageArchitecturalPattern +
        BatteryUsageDataExchangeFormat;
BatteryUsageLocationFinding = decision(”Location Finding”){
    ”GPS” : triangular(10, 10, 14);
    ”radio triangulation” : triangular(4, 5, 6);
}
BatteryUsageFileSharing = decision(”File Sharing”){
    ”OpenIntent” : triangular(5, 5, 6);
    ”In house” : triangular(0, 0, 0);
}
```

```
BatteryUsageReportSyncing = decision("Report Syncing"){
    "Explicit" : triangular(1, 3, 4);
    "Implicit" : triangular(7, 8, 10);
}
BatteryUsageChatProtocol = decision("Chat Protocol"){
    "XMPP (Open Fire)" : triangular(4, 5, 6);
    "In house" : triangular(2, 3, 12);
}
BatteryUsageMapAccess = decision("Map Access"){
    "On Demand (Google)" : triangular(4, 4, 12);
    "Cache on server" : triangular(4, 5, 12);
    "Preloaded (ESRI)" : triangular(5, 7, 7);
}
BatteryUsageHardwarePlatform = decision("Hardware Platform"){
    "Nexus I (HTC)" : triangular(3, 5, 5);
    "Droid (Motorola)" : triangular(4, 5, 14);
}
BatteryUsageConnectivity = decision("Connectivity"){
    "Wifi" : triangular(3, 4, 5);
    "3G on Nexus I" : triangular(1, 2, 3);
    "3G on Droid" : triangular(2, 4, 5);
    "Bluetooth" : triangular(2, 3, 15);
}
BatteryUsageDataBase = decision("Database"){
    "MySQL" : triangular(3, 6, 7);
    "sqLite" : triangular(5, 5, 10);
}
BatteryUsageArchitecturalPattern = decision("Architectural Pattern"){     "Peer-to-peer" :
triangular(7, 8, 10);
    "Client-Server" : triangular(5, 6, 7);
    "Push-based" : triangular(2, 4, 4);
}
BatteryUsageDataExchangeFormat = decision("Data Exchange Format"){     "XML" : tri-
angular(3, 4, 6);
    "Compressed XML" : triangular(5, 5, 7);
    "Unformatted data" : triangular(1, 1, 3);
}
```

Modelling the Performance

```
ResponseTimePreference = (ResponseTime - ResponseTimeWorst)
                        /(ResponseTimeBest - ResponseTimeWorst);

ResponseTimeRisk = Pr(ResponseTime < ResponseTimeMust);
ResponseTimeBest = deterministic(203);
ResponseTimeWorst = deterministic(2850);
ResponseTimeMust = deterministic(882);

ResponseTime = ResponseTimeLocationFinding +
        ResponseTimeFileSharing +
        ResponseTimeReportSyncing +
        ResponseTimeChatProtocol +
        ResponseTimeMapAccess +
        ResponseTimeHardwarePlatform +
        ResponseTimeConnectivity +
        ResponseTimeDataBase +
        ResponseTimeArchitecturalPattern +
        ResponseTimeDataExchangeFormat;
ResponseTimeLocationFinding = decision(”Location Finding”){
    ”GPS” : triangular(480, 500, 990);
    ”radio triangulation” : triangular(50, 100, 600);
}
ResponseTimeFileSharing = decision(”File Sharing”){
    ”OpenIntent” : triangular(50, 65, 70);
    ”In house” : triangular(40, 60, 100);
}
ResponseTimeReportSyncing = decision(”Report Syncing”){
    ”Explicit” : triangular(20, 30, 50);
    ”Implicit” : triangular(1, 4, 10);
}
```

```
ResponseTimeChatProtocol = decision(”Chat Protocol”){
    ”XMPP (Open Fire)” : triangular(40, 60, 70);
    ”In house” : triangular(30, 40, 200);
}
ResponseTimeMapAccess = decision(”Map Access”){
    ”On Demand (Google)” : triangular(700, 800, 900);
    ”Cache on server” : triangular(1, 4, 500);
    ”Preloaded (ESRI)” : triangular(1, 2, 3);
}
ResponseTimeHardwarePlatform = decision(”Hardware Platform”){
    ”Nexus I (HTC)” : triangular(40, 60, 65);
    ”Droid (Motorola)” : triangular(50, 55, 200);
}
ResponseTimeConnectivity = decision(”Connectivity”){
    ”Wifi” : triangular(30, 35, 40);
    ”3G on Nexus I” : triangular(20, 25, 40);
    ”3G on Droid” : triangular(20, 25, 40);
    ”Bluetooth” : triangular(25, 30, 200);
}
ResponseTimeDataBase =decision(”Database”){
    ”MySQL” : triangular(20, 25, 30);
    ”sqLite” : triangular(8, 10, 50);
}
ResponseTimeArchitecturalPattern = decision(”Architectural Pattern”){
    ”Peer-to-peer” : triangular(10, 20, 30);
    ”Client-Server” : triangular(25, 30, 80);
    ”Push-based” : triangular(15, 25, 40);
}
ResponseTimeDataExchangeFormat = decision(”Data Exchange Format”){
    ”XML” : triangular(20, 35, 80);
    ”Compressed XML” : triangular(12, 20, 35);
    ”Unformatted data” : triangular(3, 10, 15);
}
```

Modelling the Ramp Up Time

```
RampUpTimePreference = (RampUpTime - RampUpTimeWorst)
                       /(RampUpTimeBest - RampUpTimeWorst);

RampUpTimeRisk = Pr(RampUpTime < RampUpTimeMust);

RampUpTimeBest = deterministic(31);
RampUpTimeWorst = deterministic(83);
RampUpTimeMust = deterministic(58);

RampUpTime = RampUpTimeLocationFinding +
        RampUpTimeFileSharing +
        RampUpTimeReportSyncing +
        RampUpTimeChatProtocol +
        RampUpTimeMapAccess +
        RampUpTimeHardwarePlatform +
        RampUpTimeConnectivity +
        RampUpTimeDataBase +
        RampUpTimeArchitecturalPattern +
        RampUpTimeDataExchangeFormat

RampUpTimeLocationFinding = decision("Location Finding"){
    "GPS" : triangular(5, 6, 7);
    "radio triangulation" : triangular(7, 8, 9);
}
RampUpTimeFileSharing = decision("File Sharing"){
    "OpenIntent" : triangular(8, 9, 10);
    "In house" : triangular(5, 8, 12);
}
RampUpTimeReportSyncing = decision("Report Syncing"){
    "Explicit" : triangular(2, 2, 3);
    "Implicit" : triangular(1, 2, 2);
}
```

```
RampUpTimeChatProtocol = decision(”Chat Protocol”){
    ”XMPP (Open Fire)” : triangular(5, 6, 7);
    ”In house” : triangular(3, 4, 14);
}
RampUpTimeMapAccess = decision(”Map Access”){
    ”On Demand (Google)” : triangular(7, 9, 10);
    ”Cache on server” : triangular(7, 9, 10);
    ”Preloaded (ESRI)” : triangular(10, 13, 14);
}
RampUpTimeHardwarePlatform = decision(”Hardware Platform”){
    ”Nexus I (HTC)” : deterministic(0);
    ”Droid (Motorola)” : deterministic(0);
}
RampUpTimeConnectivity = decision(”Connectivity”){
    ”Wifi” : triangular(1, 3, 4);
    ”3G on Nexus I” : triangular(1, 2, 3);
    ”3G on Droid” : triangular(1, 2, 3);
    ”Bluetooth” : triangular(1, 2, 8);
}
RampUpTimeDataBase = decision(”Database”){
    ”MySQL” : triangular(1, 2, 3);
    ”sqLite” : triangular(3, 4, 5);
}
RampUpTimeArchitecturalPattern = decision(”Architectural Pattern”){
    ”Peer-to-peer” : triangular(10, 11, 13);
    ”Client-Server” : triangular(7, 8, 10);
    ”Push-based” : triangular(9, 10, 12);
}
RampUpTimeDataExchangeFormat = decision(”Data Exchange Format”){
    ”XML” : triangular(2, 3, 4);
    ”Compressed XML” : triangular(4, 5, 6);
    ”Unformatted data” : triangular(1, 2, 3);
}
```

Modelling the Cost

```
CostPreference = (Cost - CostWorst)/(CostBest - CostWorst);
CostRisk = Pr(Cost < CostMust);

CostBest = deterministic(550);
CostWorst = deterministic(2250);
CostMust = deterministic(1290);

Cost = CostLocationFinding +
        CostFileSharing +
        CostReportSyncing +
        CostChatProtocol +
        CostMapAccess +
        CostHardwarePlatform +
        CostConnectivity +
        CostDataBase +
        CostArchitecturalPattern +
        CostDataExchangeFormat;
CostLocationFinding = decision("Location Finding"){
   "GPS" : triangular(50, 80 100);
   "radio triangulation" : deterministic(0);
}
CostFileSharing = decision("File Sharing"){
   "OpenIntent" : deterministic(0);
   "In house" : deterministic(0);
}
CostReportSyncing = decision("Report Syncing"){
   "Explicit" : deterministic(0);
   "Implicit" : deterministic(0);
}
CostChatProtocol = decision("Chat Protocol"){
   "XMPP (Open Fire)" : deterministic(0);
   "In house" : deterministic(0);
}
```

```
CostMapAccess = decision(”Map Access”){
    ”On Demand (Google)” : deterministic(0);
    ”Cache on server” : triangular(700, 900, 950);
    ”Preloaded (ESRI)” : triangular(100, 170, 200);
}
CostHardwarePlatform = decision(”Hardware Platform”){
    ”Nexus I (HTC)” : triangular(500, 525, 530);
    ”Droid (Motorola)” : triangular(520, 520, 600);
}
CostConnectivity = decision(”Connectivity”){
    ”Wifi” : triangular(70, 80, 85);
    ”3G on Nexus I” : triangular(360, 400, 600);
    ”3G on Droid” : triangular(360, 400, 600);
    ”Bluetooth” : triangular(50, 70, 200);
}
CostDataBase = decision(”Database”){
    ”MySQL” : deterministic(0);
    ”sqLite” : deterministic(0);
}
CostArchitecturalPattern = decision(”Architectural Pattern”){
    ”Peer-to-peer” : deterministic(0);
    ”Client-Server” : deterministic(0);
    ”Push-based” : deterministic(0);
}
CostDataExchangeFormat = decision(”Data Exchange Format”){
    ”XML” : deterministic(0);
    ”Compressed XML” : deterministic(0);
    ”Unformatted data” : deterministic(0);
}
```

Modelling the Development Time

```
DevelopmentTimePreference = (DevelopmentTime - DevelopmentTimeWorst)
                            /(DevelopmentTimeBest - DevelopmentTimeWorst);

DevelopmentTimeRisk = Pr(DevelopmentTime < DevelopmentTimeMust);
DevelopmentTimeBest = deterministic(61);
DevelopmentTimeWorst = deterministic(149);
DevelopmentTimeMust = deterministic(111);

DevelopmentTime = DevelopmentTimeLocationFinding +
        DevelopmentTimeFileSharing +
        DevelopmentTimeReportSyncing +
        DevelopmentTimeChatProtocol +
        DevelopmentTimeMapAccess +
        DevelopmentTimeDataBase +
        DevelopmentTimeArchitecturalPattern +
        DevelopmentTimeDataExchangeFormat;
DevelopmentTimeLocationFinding = decision("Location Finding"){
    "GPS" : triangular(3, 4, 5);
    "radio triangulation" : triangular(10, 14, 15);
}
DevelopmentTimeFileSharing = decision("File Sharing"){
    "OpenIntent" : triangular(3, 4, 6);
    "In house" : triangular(5, 6, 15);
}
DevelopmentTimeReportSyncing = decision("Report Syncing"){
    "Explicit" : triangular(5, 6, 7);
    "Implicit" : triangular(3, 4, 4);
}
```

```
DevelopmentTimeChatProtocol = decision("Chat Protocol"){
    "XMPP (Open Fire)" : triangular(5, 6, 8);
    "In house" : triangular(7, 8, 20);
}
DevelopmentTimeMapAccess = decision("Map Access"){
    "On Demand (Google)" : triangular(14, 18, 21);
    "Cache on server" : triangular(14, 18, 21);
    "Preloaded (ESRI)" : triangular(20, 27, 30);
}
DevelopmentTimeDataBase = decision("Database"){
    "MySQL" : triangular(15, 17, 18);
    "sqLite" :triangular(15, 16, 22);
}
DevelopmentTimeArchitecturalPattern = decision("Architectural Pattern"){
    "Peer-to-peer" : triangular(20, 26, 30);
    "Client-Server" : triangular(15, 16, 20);
    "Push-based" : triangular(20, 24, 25);
}
DevelopmentTimeDataExchangeFormat = decision("Data Exchange Format"){
    "XML" : triangular(6, 7, 8);
    "Compressed XML" : triangular(7, 9, 10);
    "Unformatted data" : triangular(3, 4, 5);
}
```

Modelling the Deployment Time

```
DeploymentTimePreference = (DeploymentTime - DeploymentTimeWorst)
                          /(DeploymentTimeBest - DeploymentTimeWorst);
DeploymentTimeRisk = Pr(DeploymentTime < DeploymentTimeMust);
DeploymentTimeBest = deterministic(21) ;
DeploymentTimeWorst = deterministic(72) ;
DeploymentTimeMust = deterministic(38);

DeploymentTime = DeploymentTimeLocationFinding +
        DeploymentTimeFileSharing +
        DeploymentTimeReportSyncing +
        DeploymentTimeChatProtocol +
        DeploymentTimeMapAccess +
        DeploymentTimeConnectivity +
        DeploymentTimeDataBase +
        DeploymentTimeArchitecturalPattern;
DeploymentTimeLocationFinding = decision(”Location Finding”){
    ”GPS” : triangular(2, 3, 3);
    ”radio triangulation” : triangular(1, 1, 2);
}
DeploymentTimeFileSharing = decision(”File Sharing”){
    ”OpenIntent” : triangular(1, 1, 2);
    ”In house” : deterministic(0);
}
DeploymentTimeReportSyncing = decision(”Report Syncing”){
    ”Explicit” : triangular(1, 2, 2);
    ”Implicit” : deterministic(1);
}
DeploymentTimeChatProtocol = decision(”Chat Protocol”){
    ”XMPP (Open Fire)” : triangular(1, 1, 2);
    ”In house” : deterministic(0);
}
```

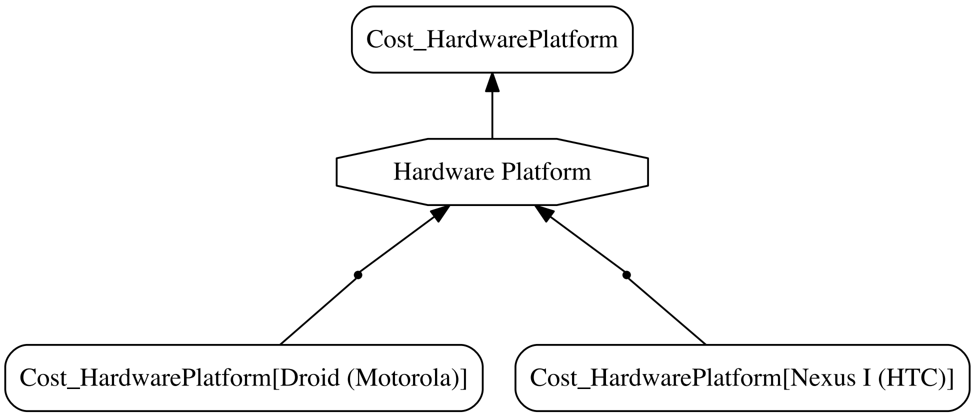
```
DeploymentTimeMapAccess = decision("Map Access"){
    "On Demand (Google)" : deterministic(0);
    "Cache on server" : triangular(3, 4, 5);
    "Preloaded (ESRI)" : triangular(3, 4, 5);
}
DeploymentTimeConnectivity = decision("Connectivity"){
    "Wifi" : triangular(5, 6, 7);
    "3G on Nexus I" : triangular(2, 3, 4);
    "3G on Droid" : triangular(2, 3, 4);
    "Bluetooth" : triangular(4, 5, 15);
}
DeploymentTimeDataBase = decision("Database"){
    "MySQL" : triangular(10, 15, 16);
    "sqLite" : triangular(13, 14, 22);
}
DeploymentTimeArchitecturalPattern = decision("Architectural Pattern"){
    "Peer-to-peer" : triangular(14, 18, 21);
    "Client-Server" : triangular(7, 9, 10);
    "Push-based" : triangular(8, 9, 12);
}
```

RADAR generates the AND/OR refinement graph and decision dependency graphs from the SAS model equations to aid visualisation of model structure. Fig.7.3a shows the partial AND/OR refinement graphs for the SAS model starting from the model variable CostHardwarePlatform. Fig.7.3b shows the partial decision graph for the model which consists of the decisions Map Access, Hardware Platform, Connectivity and Database.

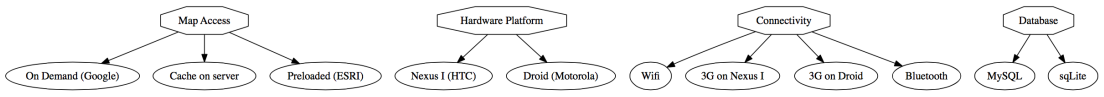
7.2.2.3 Analysis Results

The RADAR analysis of the Situation Awareness System Decision Model (SAS) is presented in Table 7.4 and Table 7.5, which show the results of the optimisation and information value analysis on the model, respectively.

Table 7.4 shows that all shortlisted solutions include options "radio triangulation" , "Open Intent" , "XMPP (Open Fire)" , "Preloaded (ESRI)" and "MySQL" . This means that, in our model, the objective values of these five options, respectively, outperform the "GPS" option for decision *Location Finding*, the "In house" option for


Cost_HardwarePlatform
Hardware Platform
Cost_HardwarePlatform[Droid (Motorola)]
Cost_HardwarePlatform[Nexus I (HTC)]

(A) Partial AND/OR refinement graph for the model variable CostHardwarePlatform of the SAS decision model.



(B) Partial decision dependency graph for the SAS decision model.

FIGURE 7.3: Partial AND/OR refinement graph and decision dependency graph for the SAS model.

decision *File Sharing*, "In house" option for decision *Chat protocol*, "Cache on Server" and "On Demand (Google)" options for decision *Map Access* and "SqLite" option for decision *Database* on both objectives. However, once these five options are selected, the shortlist includes all possible combinations of the decisions "Report Syncing", "Hardware Platform", "Connectivity", "Architectural pattern" and "Data Exchange Format". Each combination represents different tradeoffs between maximising Expected Net Benefit (ENB) and minimising Risk. To visualise such tradeoffs, RADAR generates the graph in Fig.7.4, plotting the objective values for the shortlised solutions (shown squares at the top of the graph) and all other non shortlisted ones (shown as circles).

The information value analysis results is presented in Table 7.5, which shows that the EVTPI for this problem is 0.04 and EVPPI for all the model parameters is approximately 0. This means that in this model, there is no parameter worth investigating further before deciding between the shortlisted solutions to be selected for implementation. Reducing uncertainty about any of the parameters would bring no value to the decision.

Optimisation Analysis	
Objective:	**Max** ENB
Objective:	**Min** Risk
SolutionSpace	6912
Minimal SolutionSet	6912
Shortlisted	15
Nbr. Variables	117
Nbr. Parameters	137
Nbr. Decisions	10
Runtime(s)	111

ID	Location Finding	File Sharing	Report Syncing	Chat Protocol	Map Access	Hardware Platform	Connectivity	Database	Architectural Pattern	Data Exchange Format	ENB	Risk
1	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Droid (Motorola)	Wifi	MySQL	Client-Server	Compressed XML	16.37529833	0.9997
2	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Nexus I (HTC)	3G on Droid	MySQL	Client-Server	Unformatted data	15.96290754	0.5634
3	radio triangulation	OpenIntent	Explicit	XMPP (Open Fire)	Preloaded (ESRI)	Nexus I (HTC)	3G on Droid	MySQL	Client-Server	Unformatted data	15.1679976	0.039
4	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Droid (Motorola)	3G on Droid	MySQL	Client-Server	XML	16.24388763	0.9863
5	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Droid (Motorola)	3G on Droid	MySQL	Client-Server	Compressed XML	16.32688836	0.9983
6	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Droid (Motorola)	Wifi	MySQL	Client-Server	XML	16.2922976	0.9961
7	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Nexus I (HTC)	3G on Nexus	MySQL	Client-Server	Unformatted data	15.79036997	0.4258
8	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Nexus I (HTC)	3G on Droid	MySQL	Push-based	Unformatted data	15.50738273	0.417
9	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Droid (Motorola)	3G on Nexus	MySQL	Client-Server	Unformatted data	15.99582951	0.7475
10	radio triangulation	OpenIntent	Explicit	XMPP (Open Fire)	Preloaded (ESRI)	Nexus (HTC)	3G on Droid	MySQL	Client-Server	XML	15.24351815	0.06
11	radio triangulation	OpenIntent	Explicit	XMPP (Open Fire)	Preloaded (ESRI)	Nexus I (HTC)	3G on Droid	MySQL	Client-Server	Compressed XML	15.32651889	0.1079
12	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Droid (Motorola)	Wifi	MySQL	Client-Server	Unformatted data	16.21677705	0.9429
13	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Droid (Motorola)	3G on Droid	MySQL	Client-Server	Unformatted data	16.16836707	0.8782
14	radio triangulation	OpenIntent	Implicit	XMPP (Open Fire)	Preloaded (ESRI)	Nexus I (HTC)	Wifi	MySQL	Client-Server	Unformatted data	16.01131751	0.7632
15	radio triangulation	OpenIntent	Explicit	XMPP (Open Fire)	Preloaded (ESRI)	Droid (Motorola)	3G on Droid	MySQL	Client-Server	Unformatted data	15.37345714	0.1928

Table 7.4: Optimisation analysis results for the Situation Awareness System analysis model

Information Value Analysis

Objective: **Max** ENB
EVTPI: 0.04

Parameter	EVPPI	Parameter	EVPPI	Parameter	EVPPI
BatteryUsageLocationFinding[GPS]	0	ReliabilityArchitecturalPattern[Push-based]	0	ReliabilityLocationFinding[GPS]	0
BatteryUsageLocationFinding[radio triangulation]	0	RampUpTimeLocationFinding[GPS]	0	DevelopmentTimeDataExchangeFormat[Compressed XML]	0
BatteryUsageFileSharing[OpenIntent]	0	RampUpTimeLocationFinding[radio triangulation]	0	ReliabilityLocationFinding[radio triangulation]	0
BatteryUsageFileSharing[In house]	0	RampUpTimeFileSharing[OpenIntent]	0	DevelopmentTimeDataExchangeFormat[Unformatted data]	0
BatteryUsageReportSyncing[Explicit]	0	RampUpTimeFileSharing[In house]	0	ReliabilityFileSharing[OpenIntent]	0
BatteryUsageReportSyncing[Implicit]	0	RampUpTimeReportSyncing[Explicit]	0	DeploymentTimeLocationFinding[GPS]	0
BatteryUsageChatProtocol[XMPP (Open Fire)]	0	RampUpTimeReportSyncing[Implicit]	0	ReliabilityFileSharing[In house]	0
BatteryUsageChatProtocol[In house]	0	RampUpTimeChatProtocol[XMPP (Open Fire)]	0	DeploymentTimeFileSharing[OpenIntent]	0
DeploymentTimeFileSharing[OpenIntent]	0	DeploymentTimeLocationFinding[radio triangulation]	0	DeploymentTimeReportSyncing[Explicit]	0
BatteryUsageMapAccess[On Demand (Google)]	0	RampUpTimeChatProtocol[In house]	0	ReliabilityReportSyncing[Explicit]	0
BatteryUsageMapAccess[Cache on server]	0	RampUpTimeMapAccess[On Demand (Google)]	0	ReliabilityReportSyncing[Implicit]	0
BatteryUsageMapAccess[Preloaded (ESRI)]	0	RampUpTimeMapAccess[Cache on server]	0	DeploymentTimeChatProtocol[XMPP (Open Fire)]	0
BatteryUsageHardwarePlatform[Nexus I (HTC)]	0	RampUpTimeMapAccess[Preloaded (ESRI)]	0	ReliabilityChatProtocol[In house]	0
BatteryUsageHardwarePlatform[Droid (Motorola)]	0	RampUpTimeConnectivity[Wifi]	0	DeploymentTimeMapAccess[Cache on server]	0
BatteryUsageConnectivity[Wifi]	0	RampUpTimeConnectivity[3G on Nexus I]	0	ReliabilityMapAccess[On Demand (Google)]	0
BatteryUsageConnectivity[3G on Nexus I]	0	RampUpTimeConnectivity[3G on Droid]	0	DeploymentTimeMapAccess[Preloaded (ESRI)]	0
BatteryUsageConnectivity[3G on Droid]	0	RampUpTimeConnectivity[Bluetooth]	0	DeploymentTimeConnectivity[Wifi]	0
BatteryUsageConnectivity[Bluetooth]	0	RampUpTimeDataBase[MySQL]	0		
BatteryUsageDataBase[MySQL]	0	RampUpTimeDataBase[sqLite]	0		
BatteryUsageDataBase[sqLite]	0	RampUpTimeArchitecturalPattern[Peer-to-peer]	0	ReliabilityMapAccess[Cache on server]	0
BatteryUsageArchitecturalPattern[Peer-to-peer]	0	RampUpTimeArchitecturalPattern[Client-Server]	0	ReliabilityMapAccess[Preloaded (ESRI)]	0
BatteryUsageArchitecturalPattern[Client-Server]	0	RampUpTimeArchitecturalPattern[Push-based]	0	DeploymentTimeConnectivity[3G on Nexus I]	0
BatteryUsageArchitecturalPattern[Push-based]	0	RampUpTimeDataExchangeFormat[XML]	0	ReliabilityConnectivity[Wifi]	0
BatteryUsageDataExchangeFormat[XML]	0	RampUpTimeDataExchangeFormat[Compressed XML]	0	DeploymentTimeConnectivity[3G on Droid]	0
BatteryUsageDataExchangeFormat[Compressed XML]	0	RampUpTimeDataExchangeFormat[Unformatted data]	0	ReliabilityConnectivity[3G on Nexus I]	0
BatteryUsageDataExchangeFormat[Unformatted data]	0	CostLocationFinding[GPS]	0	DeploymentTimeConnectivity[Bluetooth]	0
ResponseTimeLocationFinding[GPS]	0	CostMapAccess[Cache on server]	0	ReliabilityConnectivity[3G on Droid]	0
ResponseTimeLocationFinding[radio triangulation]	0	CostMapAccess[Preloaded (ESRI)]	0	DeploymentTimeDataBase[MySQL]	0
ResponseTimeFileSharing[OpenIntent]	0	CostHardwarePlatform[Nexus I (HTC)]	0	ReliabilityConnectivity[Bluetooth]	0
ResponseTimeFileSharing[In house]	0	CostHardwarePlatform[Droid (Motorola)]	0	DeploymentTimeDataBase[sqLite]	0
ResponseTimeReportSyncing[Explicit]	0	CostConnectivity[Wifi]	0	ReliabilityDataBase[MySQL]	0
ResponseTimeReportSyncing[Implicit]	0	CostConnectivity[3G on Nexus I]	0	DeploymentTimeArchitecturalPattern[Peer-to-peer]	0
ResponseTimeChatProtocol[XMPP (Open Fire)]	0	CostConnectivity[3G on Droid]	0	DeploymentTimeArchitecturalPattern[Client-Server]	0
ResponseTimeChatProtocol[In house]	0	CostConnectivity[Bluetooth]	0	ReliabilityDataBase[sqLite]	0
ResponseTimeMapAccess[On Demand (Google)]	0	DevelopmentTimeLocationFinding[GPS]	0	ReliabilityArchitecturalPattern[Peer-to-peer]	0
ResponseTimeMapAccess[Cache on server]	0	DevelopmentTimeLocationFinding[radio triangulation]	0	DeploymentTimeArchitecturalPattern[Push-based]	0
ResponseTimeMapAccess[Preloaded (ESRI)]	0	DevelopmentTimeFileSharing[OpenIntent]	0	ReliabilityArchitecturalPattern[Client-Server]	0
ResponseTimeHardwarePlatform[Nexus I (HTC)]	0	DevelopmentTimeFileSharing[In house]	0		
ResponseTimeHardwarePlatform[Droid (Motorola)]	0	DevelopmentTimeReportSyncing[Explicit]	0		
ResponseTimeConnectivity[Wifi]	0	DevelopmentTimeReportSyncing[Implicit]	0		
ResponseTimeConnectivity[3G on Nexus I]	0	DevelopmentTimeChatProtocol[XMPP (Open Fire)]	0		
ResponseTimeConnectivity[3G on Droid]	0	DevelopmentTimeChatProtocol[In house]	0		
ResponseTimeConnectivity[Bluetooth]	0	DevelopmentTimeMapAccess[On Demand (Google)]	0		
ResponseTimeDataBase[MySQL]	0	DevelopmentTimeMapAccess[Cache on server]	0		
ResponseTimeDataBase[sqLite]	0	DevelopmentTimeMapAccess[Preloaded (ESRI)]	0		
ResponseTimeArchitecturalPattern[Peer-to-peer]	0	DevelopmentTimeDataBase[MySQL]	0		
ResponseTimeArchitecturalPattern[Client-Server]	0	DevelopmentTimeDataBase[sqLite]	0		
ResponseTimeArchitecturalPattern[Push-based]	0	DevelopmentTimeArchitecturalPattern[Peer-to-peer]	0		
ResponseTimeDataExchangeFormat[XML]	0	DevelopmentTimeArchitecturalPattern[Client-Server]	0		
ResponseTimeDataExchangeFormat[Compressed XML]	0	DevelopmentTimeArchitecturalPattern[Push-based]	0		
ResponseTimeDataExchangeFormat[Unformatted data]	0	DevelopmentTimeDataExchangeFormat[XML]	0		

Table 7.5: Information Value Analysis for the Situation Awareness System Model.

7.2.2.4 Comparison To Previous Analysis Approaches

Esfahani et al.[84] was the first to use the Emergency Response System to evaluate the GuideArch approach. GuideArch, a fuzzy logic-based framework, explores architectural solution space in order to aid informed decisions in the presence of uncertainty. Esfahani et al. optimised the design of an Emergency Response System by considering seven system properties, such as development time, battery usage, response time, cost, ramp uptime and reliability.

Letier et al.[166] also used the Emergency Response System in evaluating the Multi-Objective Decision Analyser (MODA) –a statistical decision analysis technique and Pareto-based multi-objective optimisation to early requirements and architecture design decisions. Letier et al. optimised the design of an Emergency Response System by considering the same system properties as Esfahani et al.[84]. They defined the optimisation objectives as maximising the project utility and minimising the project failure risk.

The RADAR model presented for the SAS model in previous pages is similar to the equations and parameter values in the GuideArch and MODA models.

With respect to the decision analysis approach, RADAR's analysis approach differs from GuideArch, but similar to that of MODA. GuideArch shortlists its optimal solutions by comparing candidate architectures using fuzzy logic values that are not falsifiable or cannot be validated empirically. Like RADAR, MODA uses exhaustive search to shortlist the Pareto optimal solutions and computes expected value of perfect and imperfect information. Both MODA and RADAR shortlisted solutions (using exhaustive search) that have the same alternative options per decision. Their shortlisted solutions agree on the selected options for four decisions, such as "Location finding", "Report Syncing", Map Access" and "Database". The remaining decisions agree on at least one alternative option. In terms of the analysis runtimes, RADAR significantly outperforms MODA: RADAR took less than 2 minutes to analyse the SAS model while MODA took 5 minutes and required manual coding and significant optimisation of the simulation function in R (without optimisation, the simulation of the whole search-space would have taken 7 hours). We also observed similar values for the EVTPI: RADAR and MODA estimate

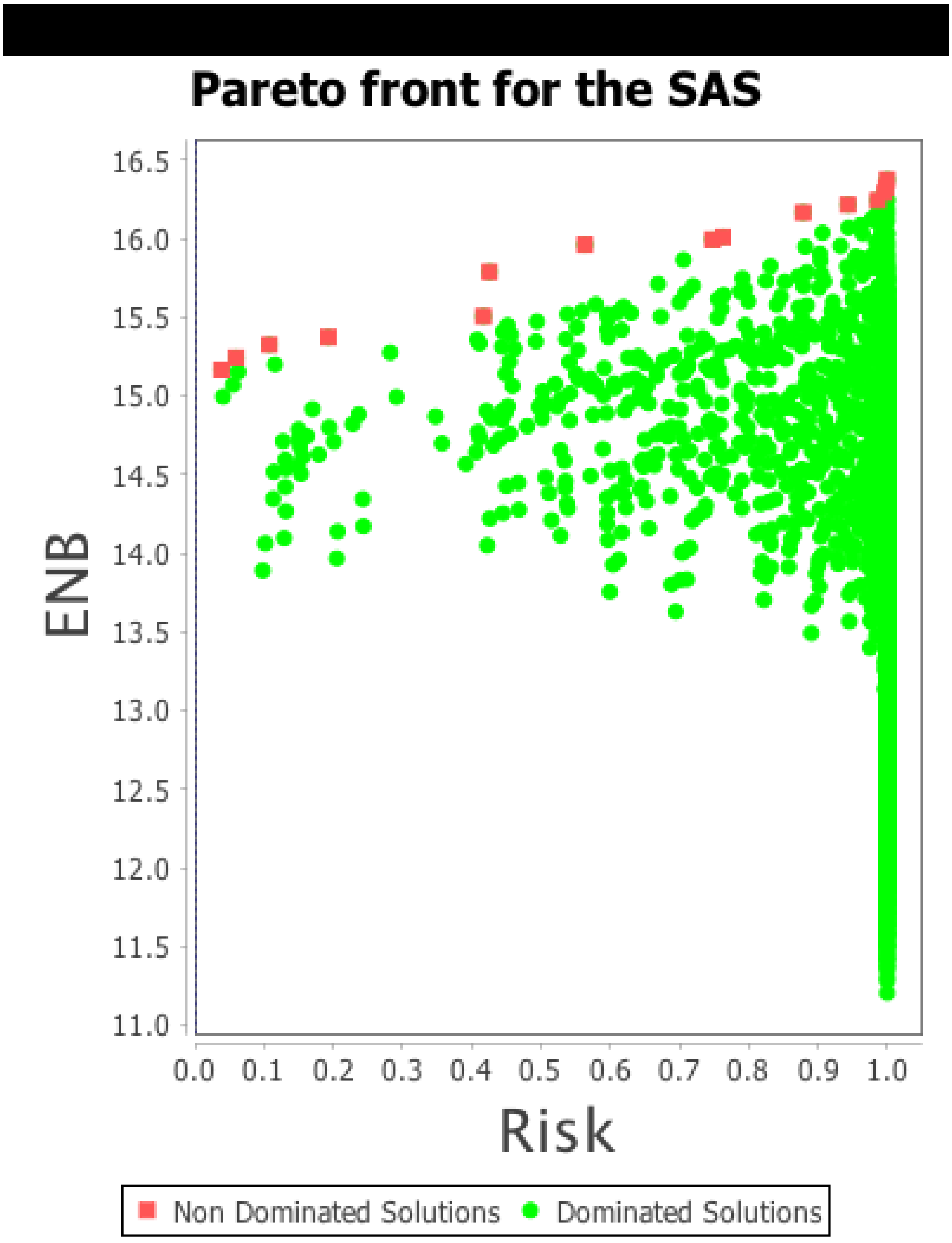


FIGURE 7.4: Pareto front of the the emergency response system model analysis.

the EVTPI as 0.04 and 0.05, respectively. These results help to validate the correctness of our automated decision analysis technique and show the superiority of RADAR's automated approach over the manual approach used in MODA.

7.2.3 Building Security Policy Decision System

7.2.3.1 Problem Statement

Building security policies allows organisations to restrict access to confidential information only to authorised personnel, i.e. employees of the organisation. Our modelling and analysis of this problem is motivated by and based on previous studies of improving security policy decisions with models [51] and modelling and simulating systems security policy [50].

The organisation is concerned about its employees habit of sharing documents through a globally shared drive. Storing files on the shared drive facilitates the employees work but increases the risk and extent of insiders leaks. The organisation is considering recommending its employees to share documents through emails or portable media devices instead of using the shared drive.

The organisation is also concerned by theft of documents stored on portable media devices (e.g. USB, CD). The model considers only theft of such devices inside the organisation premises. To prevent thieves from entering the building, its entrance is equipped with automated gates where employees have to swipe their access card to enter the building. Because of tailgating risks, the organisation is considering adding a security guard to reinforce security at the building entrance.

The building security policy model decisions are thus:

- the *document sharing policy* decision in which the organisation recommends that employees use "email" or "external media" or the use of "shared drive". In our model, we refer to "shared drive" as Neutral.
- the *building entry security* decision that can be "not-guarded", in which case the organisation does not deploy a security person at the entrance of the company and "guarded" in which a security person is deployed at the entrance.

These decisions impacts the cost incurred by the organisation when confidential information are leaked. The organisation wants to minimise the costs of disclosure and minimise the chances of disclosure with very high costs (above £1m)

7.2.3.2 RADAR Model

Modelling the Optimisation Objectives

The primary decision objectives are related to the uncertain costs associated to the disclosure of confidential documents:

```
Objective  Min ExpectedCostOfDisclosures = EV(CostOfDisclosures);
Objective  Min RiskOfCatastrophicDisclosures = Pr(CostOfDisclosure > 10^6);
```

Modelling Costs of Disclosures

Our model assumes that there are three categories of confidential documents, i.e. high, medium, and low confidentiality, with different costs to the organisation, if they are leaked. Our model assumes that the cost of document disclosure is a function of number of leaked confidential documents and the cost of the leaked documents:

```
CostOfDisclosures =
NbrHighConfidentialityLeaks × CostHighConfidentialityLeak
    + NbrMediumConfidentialityLeaks × CostMediumConfidentialityLeak
        + NbrLowConfidentialityLeaks × CostLowConfidentialityLeak;
```

The number of leaked confidential documents, for each category of confidential documents, is the product of the number of leaked documents multiplied by the ratio of confidential documents leaked:

```
NbrHighConfidentialityLeaks = NbrLeakedDoc × RatioHighConfientialityDocs;
NbrMediumConfidentialityLeaks =
                NbrLeakedDoc × RatioMediumConfientialityDocs;
NbrLowConfidentialityLeaks = NbrLeakedDoc × RatioLowConfientialityDocs;
```

Each document confidentiality category has some uncertainty about the ratio of confidential documents leaked and uncertainty in cost of disclosure:

RatioHighConfidentialityDocs = **uniform**(0.5%, 2%);

RatioMediumConfidentialityDocs =**uniform**(4%, 10%);

RatioLowConfidentialityDocs = **uniform**(30%, 60%);

CostHighConfidentialityLeak = **normalCI**($0.5 * 10^6, 5 * 10^6$);

CostMediumConfidentialityLeak = **normalCI**($10^4, 5 * 10^4$);

CostLowConfidentialityLeak = **normalCI**($10^2, 10^4$);

Modelling Documents Leaks

Documents can be leaked by insiders or stolen by outsiders who managed to gain access to the building.

NbrLeakedDoc = NbrDocsLeakedByInsiders × NbrDocsStolenByOutsiders;

Modelling Insiders' Leaks

The number of documents leaked by an insider is a function of the probability of an insider leak, *ProbabilityInsidersLeak* , the number of documents on shared drive, *Nbr-DocsOnSharedDrive* and the *SharedDriveLeakRange*, which is the portion of documents on the shared drive that are leaked by insider when a leaks occurs:

NbrDocsLeakedByInsiders = ProbabilityInsidersLeak × NbrDocsOnSharedDrive
× SharedDriveLeakRange;

ProbabilityInsidersLeak = **deterministic**(10^{-3});

SharedDriveLeakRange = **triangular**(10%, 50%, 100%);

Modelling Attackers' Intrusions

The attackers' intrusion model attempts to predict the number of documents stolen by outsiders over a year based on whether or not the building security gates are guarded or not. The number of documents stolen over a year depends on the number of intrusions during the year and the number of documents stolen during each intrusion, which depends on the number of external media in use and the number of documents stored in each media.

```
NbrDocsStolenByOutsiders =
        NbrOfficeIntrustions × NbrDocsStolenPerIntrusion;
NbrOfficeIntrusions = decision("Building entry security") {
    "not guarded" : triangular(0, 3, 6);
    "guarded" : triangular(0, 1, 2);
}
NbrDocsStolenPerIntrusion =
        NbrExternalMediaStolenPerIntrusion × NbrDocsPerExternalMedia
NbrExternalMediaStolenPerIntrusion =
        NbrExternalMediaInUse × PercentageMediaStolenPerInstrusion
PercentageMediaStolenPerInstrusion = triangular(0, 1%, 10%);
```

Modelling Documents Sharing

The number of documents on the shared drive and on external media depends on the organisations document sharing policy. Data about the number of documents on shared drive and on external media come from Table 1 in [50].

```
NbrDocsOnSharedDrive = decision("Document Sharing Policy"){
    "Neutral": deterministic(143);
    "Recommend Email": deterministic(44);
    "Recommend External Media": deterministic(91);
}
NbrDocsOnExternalMedia = decision("Document Sharing Policy"){
    "Neutral": deterministic(0);
    "Recommend Email": deterministic(0);
    "Recommend External Media": deterministic(52);
}

NbrExternalMediaInUse = NbrDocsOnExternalMedia / NbrDocsPerMedia;
NbrDocsPerMedia = triangular(0, 5, 10) ;
```

To help visualise the model structure, RADAR generates the AND/OR refinement graph and decision dependency graphs from the building security model equations. Fig. 7.5

shows the partial AND/OR refinement graphs for the building security model starting from the model variable CostOfDisclosure. Fig. 7.6 shows the decisions decision graph.

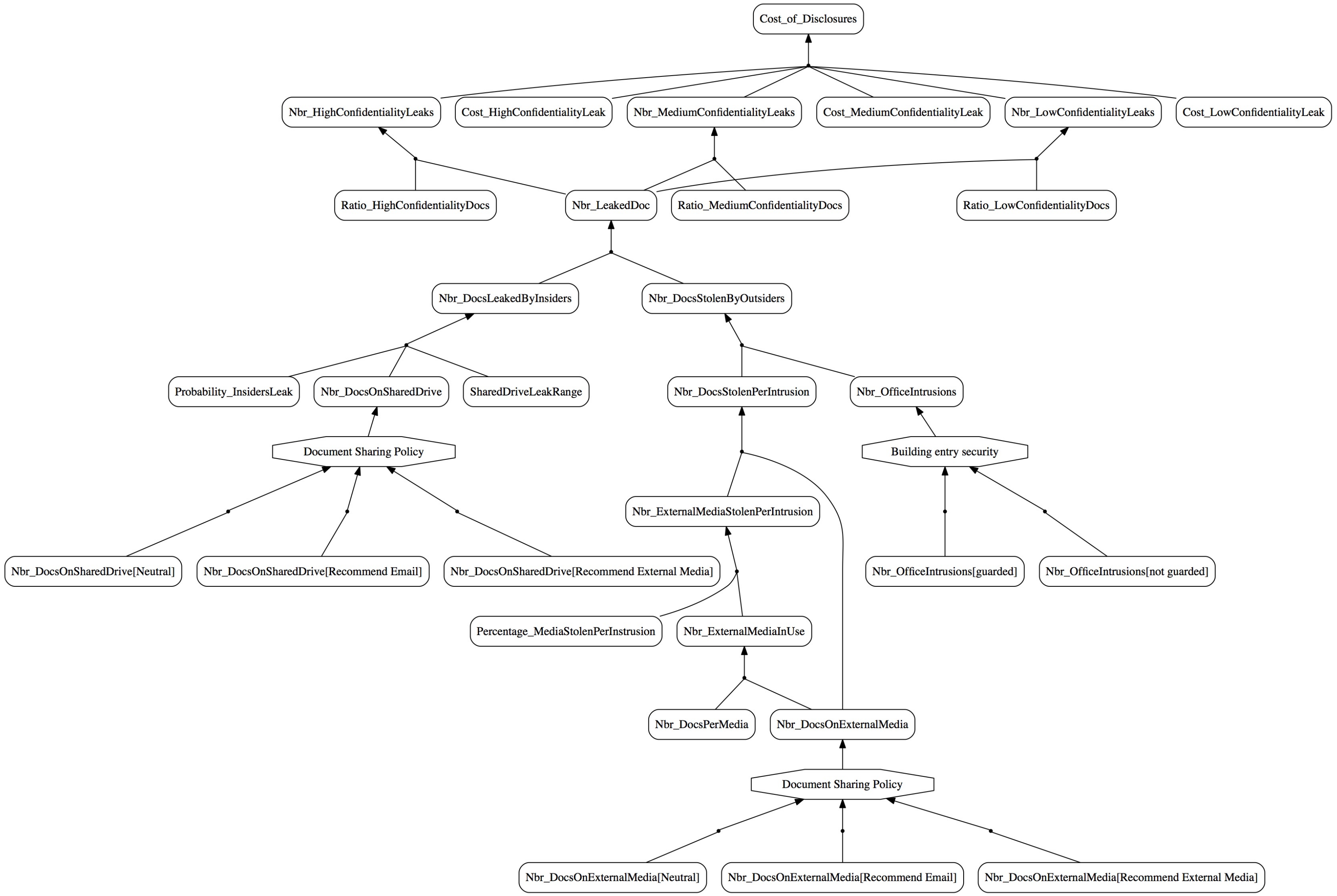

Cost_of_Disclosures
Nbr_HighConfidentialityLeaks
Cost_HighConfidentialityLeak
Nbr_MediumConfidentialityLeaks
Cost_MediumConfidentialityLeak
Nbr_LowConfidentialityLeaks
Cost_LowConfidentialityLeak
Ratio_HighConfidentialityDocs
Nbr_LeakedDoc
Ratio_MediumConfidentialityDocs
Ratio_LowConfidentialityDocs
Nbr_DocsLeakedByInsiders
Nbr_DocsStolenByOutsiders
Probability_InsidersLeak
Nbr_DocsOnSharedDrive
SharedDriveLeakRange
Nbr_DocsStolenPerIntrusion
Nbr_OfficeIntrusions
Document Sharing Policy
Building entry security
Nbr_ExternalMediaStolenPerIntrusion
Nbr_DocsOnSharedDrive[Neutral]
Nbr_DocsOnSharedDrive[Recommend Email]
Nbr_DocsOnSharedDrive[Recommend External Media]
Nbr_OfficeIntrusions[guarded]
Nbr_OfficeIntrusions[not guarded]
Percentage_MediaStolenPerInstrusion
Nbr_ExternalMediaInUse
Nbr_DocsPerMedia
Nbr_DocsOnExternalMedia
Document Sharing Policy
Nbr_DocsOnExternalMedia[Neutral]
Nbr_DocsOnExternalMedia[Recommend Email]
Nbr_DocsOnExternalMedia[Recommend External Media]

FIGURE 7.5: AND/OR refinement graph for the model variable CostOfDisclosure of the building security model.

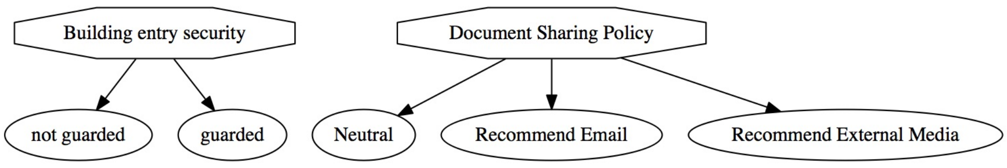


Figure 7.6: Decision dependency graph for the building security decision model.

7.2.3.3 Analysis Results

The RADAR analysis of the Building Security Policy Decision Model (BSPDM) is presented in Table 7.6, which shows the results of the optimisation and information value analysis on the model.

The first part of Table 7.6 is the optimisation analysis results, which show that the two shortlisted solutions (those that are Pareto optimal) include "recommend email" for the *document sharing policy* decision. This means that, in our model, the objective value of the option share documents by email outperforms both the "neutral" and "external media" in both objectives. But when the "recommend email" option is selected, the shortlist includes both the "guard" and "not guarded" options of the building entry security decision; each combination representing a different tradeoffs between minimising the expected cost of disclose, *ExpectedCostOfDisclosures* and minising the risk of catastrophic disclosure, *RiskOfCatastrophicDisclosures*. To visualise such tradeoffs, RADAR generates the graph in Fig. 7.7, plotting the objective values for the shortlised solutions (shown in red) and all other non shortlisted ones (shown in green).

The second part of Table 7.6 is the information value analysis results, which shows that the EVTPI for this problem is 0 and EVPPI for all the model parameters are 0. This means that in this model, there is no parameter worth investigating further before deciding between the shortlisted solutions to be selected for implementation. Reducing uncertainty about any of the parameters would bring no value to the decision.

7.2.3.4 Comparison To Previous Analysis Approaches

Caulfield et al. used the Building Security Policy System in their work on improving security policy decisions with models [49] and modelling and simulating security policy

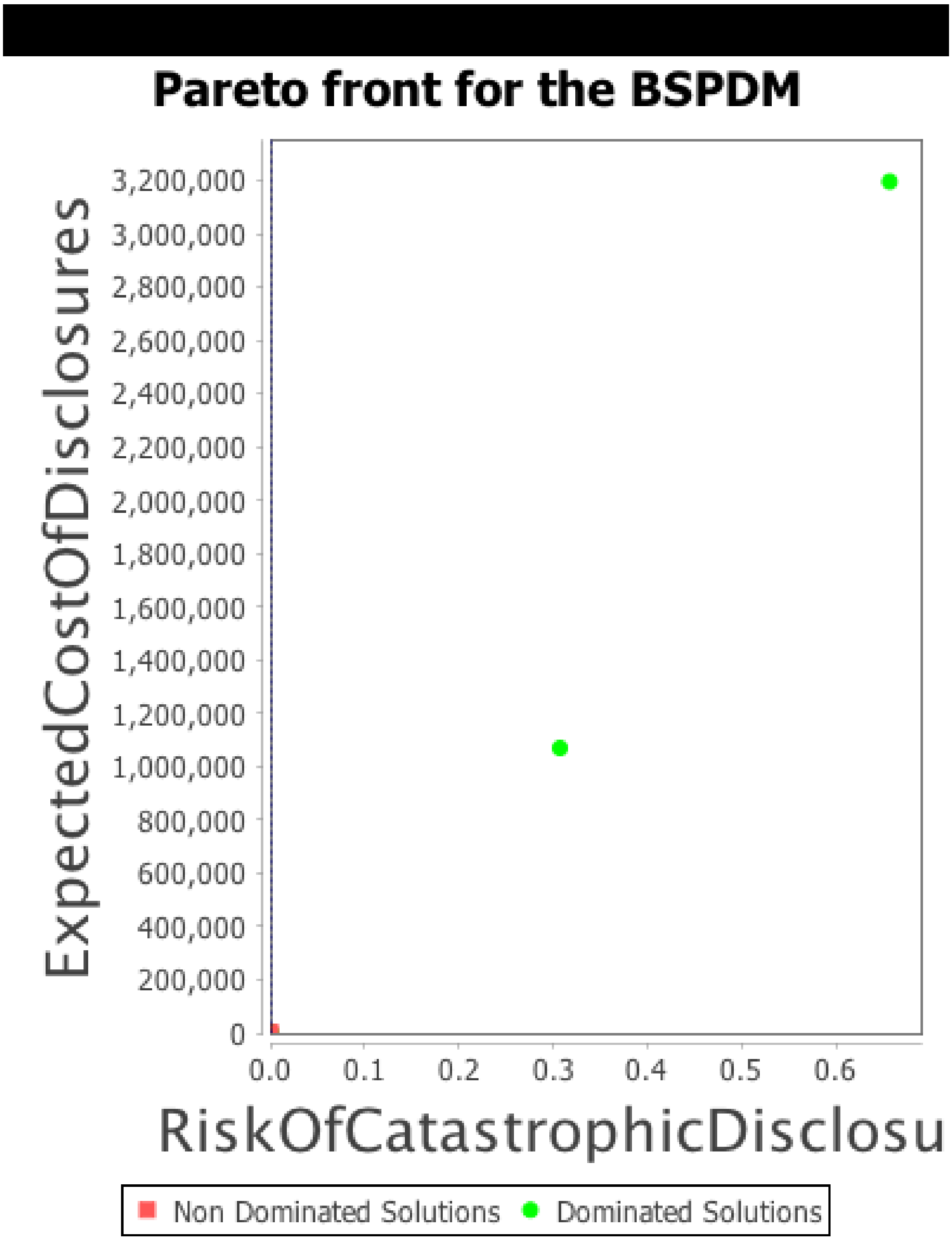


FIGURE 7.7: Pareto front of the the building security policy model analysis.

system design choices [48].

Caulfield et al. [48] developed a modelling methodology and framework for predicting the impact and effectiveness of alternative security policy choices on an organisation's

Optimisation Analysis

Objective:	**Max** ExpectedCostOfDisclosures
Objective:	**Min** RiskOfCatastrophicDisclosures
SolutionSpace	6
Minimal SolutionSet	6
Shortlisted	2
Nbr. Variables	23
Nbr. Parameters	11
Nbr. Decisions	2
Runtime(s)	0

ID	Building entry security	Document Sharing Policy	ExpectedCostOfDisclosures	RiskOfCatastrophicDisclosures	Optimal
1	not guarded	Recommend Email	904	0	Yes
2	guarded	Recommend Email	904	0	Yes
3	guarded	Recommend External Media	1052645	0.2985	No
4	not guarded	Recommend External Media	3181382	0.6464	No
5	not guarded	Neutral	2937	0	No
6	guarded	Neutral	2937	0	No

Information Value Analysis

Objective:	**Max** ENB
EVTPI:	0

Parameter	EVPPI
RatioHighConfidentialityDocs	0
RatioMediumConfidentialityDocs	0
RatioLowConfidentialityDocs	0
CostHighConfidentialityLeak	0
CostMediumConfidentialityLeak	0
CostLowConfidentialityLeak	0
SharedDriveLeakRange	0
NbrOfficeIntrusions[not guarded]	0
NbrOfficeIntrusions[guarded]	0
PercentageMediaStolenPerInstrusion	0
NbrDocsPerMedia	0

Table 7.6: Optimisation analysis and information value analysis results for the building security policy analysis model

operations. They used approaches from mathematical modelling and simulation, information security and economics to model and analyse security decision models. The models developed in [48] considers three different areas of an organisation's security: tailgating behaviour of staffs and intruders at the entrance to the building; confidential documents sharing between employees within the office, when the usual, secure sharing method is unavailable; and loss of employees devices, possibly containing confidential information. In their approach, the decision about which alternative system design to select is evaluated through simulation (10^4 executions) using a utility function defined below as:

$$Utility = \sum_{a \in A} w_a * f_a * (v_a - \vec{v}_a) \tag{7.1}$$

where A is a set of attributes stakeholders care about, e.g, the number of documents added to the global share and the number of documents found by intruders inside the office; w_a represents weights assigned to attributes denoting importance to the user; v_a and $\vec{v}_a$ are the actual and target values of attribute a; and f_a captures how stakeholders' care about the difference in between the actual and target values for attribute a.

The RADAR model presented for the building security policy system model differs from that presented by Caulfield et al. [48], but uses similar model parameter values. Their model uses pre-defined, fixed model equations to assign abstract values to stakeholder attributes in the decision model. Our models focus on elaborating domain specific decision problems within the organisation security context. However, the solutions suggested by RADAR's are similar to that obtained in Caulfield et al. [48]: both approaches suggest the organisation recommend to their employees to use the "email" policy irrespective of whether the organisation deploys a guard or not.

7.2.4 Multi-Objective Next Release Problem

7.2.4.1 Problem Statement

The requirement subset selection problem addresses the question of "what features to build in the next release of a software system". This type of problem is commonly referred to as the Next Release Problem (NRP). Generally, the NRP consists in selecting among N requirements, a subset of requirements to be implemented in the next release of a product [30, 283, 285]. The problem has a solution space of 2^N solutions, where N is the total number of candidate requirements.

Bagnal et al. formulated a mathematical model for the NRP [30]. Zhang et al. [283] reformulated this problem to a Multi-Objective optimisation problem with two objectives —maximising value and minimising cost. In Zhang's NRP model, a set of requirements, $R = \{r_1, r_2, ..., r_n\}$, with respective costs, C = { $\text{cost}_1, cost_2, ..., cost_n\}$ are requested to be added to an (existing) software product in order to satisfy a set of stakeholders (customers), $S = \{s_1, s_2, ..., s_m\}$. These stakeholders have individual degree of importance to the company denoted by a Weight, W $= \{w_1, w_2, ..., w_m\}$, where $w_i \in [0, 1]$ and $\sum_{i=1}^{m} w_i = 1$. In this model, the author assumed that requirements are independent and that a customer places different level of importance to each requirement. Each stakeholder $s \in S$ assigns a value denoted by $value(r_j, s_i)$ to a requirement $r_j (1 \leq j \leq n)$, where $value(r_j, s_i) > 0$ if stakeholder i desires implementation of the requirement j and 0 otherwise. The value derived by a company for a given set of requirements can be represented using the matrix below:

$$Value = \begin{pmatrix} v_{1,1} & v_{1,2} & \cdots & v_{1,i} & \cdots & v_{1,n} \\ v_{2,1} & v_{2,2} & \cdots & v_{2,i} & \cdots & v_{2,n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ v_{j,1} & v_{j,2} & \cdots & v_{j,i} & \cdots & v_{j,n} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ v_{m,1} & v_{m,2} & v_{m,3} & \cdots & v_{m,n} & \end{pmatrix} \tag{7.2}$$

The overall value or importance for a given requirement $r_j (1 \leq j \leq n)$ is regarded as the *Score* and is defined below [283]:

$$Score_j = \sum_{i=1}^{m} w_i.value(r_j, s_i) \tag{7.3}$$

The cost vector for the set of requirements $r_j (1 \leq j \leq n)$ is given as

$$Cost = \{cost_1, cost_2, ..., cost_n\} \tag{7.4}$$

In Zhang's model, the optimistaion objectives are:

$$Maximise \sum_{j=1}^{n} score_j.x_j \tag{7.5}$$

$$Minimise \sum_{j=1}^{n} cost_j.x_j \tag{7.6}$$

where x_j is the decision vector $\vec{x} = \{x_1, x_2, ..., x_n\} \in \{0,1\}$ that determines if a requirement is selected or not.

A key factor that also impacts requirements subset selection activity is the inter-dependency relationship between requirements. Such relationships might be that some requirements are coupled, i.e., they have to be implemented simultaneously. In other cases, a requirement requires another requirement to be fulfilled before it can be implemented. To address this concern, NRP variants extend the problem by including dependency constraints between requirements [285]. These constraint relationships, for example, include AND, OR and precedence constraint relationships.

We illustrate the application of RADAR on the NRP of a commercial Release Planning Tool with 25 requirements [142] as shown Table 7.1. The appendix section presents another application of release planning, i.e., we apply release planning method to plan future releases of a Word Processor that has 50 requirements.

7.2.4.2 RADAR Model

In the commercial Release Planning Tool [142], the stakeholders are interested in maximising the value derived from implementing the requirements and minimising implementation costs which includes the design cost, development cost, the cost accrued from external tool development, and the cost due to software testing. The requirements have dependency (requires and coupling) relationships between them. The datasets used in developing our model is provided in [142] and contains information about the requirements cost estimates, requirements dependencies, available budget and expected revenue and value.

Modelling Optimisation Objectives

The optimisation objectives for the release planner are given below:

```
Objective  Max ExpectedNetBenefit = EV(NB);
Objective  Min ProjectRisk = Pr(ProjectFailure);
Objective  Max ExpectedFrequencyOfUse = EV(FrequencyOfUse);
Objective  Min ExpectedDissatisfaction = EV(Dissatisfaction);
Objective  Min ExpectedRequirementVolatility = EV(Volatility);
```

In the above expression, the first objective is a maximisation of the expected net benefit (ExpectedNetBenefit) from selecting a subset of requirements, and the second objective is a minimisation of the project risk (ProjectRisk). The third objective maximises the Frequency of Use i.e. we are interested in requirements that have been implemented before or used frequently. The fourth objective is a minimisation of stakeholders dissatisfaction. The fifth objective states that the we want to minimise requirements changes after the basic set have been agreed.

The net benefit of implementing any subset of requirements is defined below:

```
NB = Value - Cost;
```

The Value derived from the subset of requirements implemented is estimated considering uncertainty in the perceived values of each requirements by the stakeholders.

We model the Value as below:

```
Value = decision-subset(+) ("Next Release"){
    "Hierarchical dependencies" : normalCI(200,400);
    "Grouping of features" : normalCI(100,200);
    "Pre-assignments" : normalCI(300,600);
    "Feature dependencies" : normalCI(600,700);
    "Feasibility analysis" : normalCI(200,500);
    "Flexible number of releases" : normalCI(200,400);
    "Flexible number and type of criteria" : normalCI(200,600);
    "Fexible number and type of resources" : normalCI(100,900);
    "Type 1 Stakekholder consensus driven planning" : normalCI(100,200);
    "Type 2 Financially driven planning" : normalCI(800,900);
    "Ranking of features based on different criteria" : normalCI(200,400);
    "Similarity analysis" : normalCI(200,900);
    "Dual charts using ranking and disagreement analysis": normalCI(200,400);
    "Comparison of priorities between stakeholders": normalCI(100,300);
    "Import manual plan" : normalCI(400,500);
    "Import of project data" : triangular(400,500,900);
    "Re-import of updated project data" : triangular(200, 400, 800);
    "Export of plans and project data" : normalCI(200,900);
    "Export of generated analysis charts" : triangular(200, 400, 600);
    "Trade-off analysis" : normalCI(300,900);
    "Estimated stakeholder satisfaction analysis" : normalCI(200,900);
    "Consensus analysis between alternative plans" : normalCI(100,500);
    "Structure of alternative plans" : normalCI(200,300);
    "Quality evaluation of alternative plans" : normalCI(100,400);
    "Resource evaluation of alternative plans" : normalCI(100,900);
}
```

We define the Project Failure below:

```
ProjectFailure = 1 − ( (1 − RiskExceedingBackEndDevCost) ×
                       (1 − RiskExceedingFrontEndDevCost) ×
                       (1 − RiskExceedingTestingCost) ×
                       (1 − RiskExceedingProjectManagementCost) ×
                       (1 − RiskExceedingRequirementAnalysisCost);
```

We model the risk of exceeding back-end development budget as:

```
RiskExceedingBackEndDevCost = BackEndDevCost > BackedDevBudget;
BackedDevBudget = deterministic(2000);
```

The risk of exceeding front-end development budget is modelled as:

```
RiskExceedingFrontEndDevCost = FrontEndDevCost > FrontEndDevBudget;
FrontEndDevBudget = deterministic(1104);
```

We model the risk of exceeding testing budget as:

```
RiskExceedingTestingCost = TestingCost > TestingBudget;
TestingBudget = deterministic(2160);
```

The risk of exceeding project management budget is modelled as:

```
RiskExceedingProjectManagementCost = ProjectManagementCost > ProjectManagementBud-
get;
ProjectManagementBudget = deterministic(1060);
```

We model the risk of exceeding the Quality Assurance budget as:

```
RiskExceedingQACost = QACost > QABudget;

QABudget = deterministic(1680);
```

The risk of exceeding requirements elicitation budget is modelled as:

```
RiskExceedingRequirementAnalysisCost = RequirementAnalysisCost > RequirementAnalysis-
Budget;

RequirementAnalysisBudget = deterministic(600);
```

The frequency of use for each requirement is modelled below:

```
FrequencyOfUse = decision-subset(+) ("Next Release"){
    "Hierarchical dependencies" : normalCI(1,9);
    "Grouping of features" : normalCI(1,7);
    "Pre-assignments" : normalCI(1,9);
    "Feature dependencies" : normalCI(1,6);
    "Feasibility analysis" : normalCI(4,7);
    "Flexible number of releases" : normalCI(1,9);
    "Flexible number and type of criteria" : normalCI(1,9);
    "Fexible number and type of resources" : normalCI(1,9);
    "Type 1 Stakekholder consensus driven planning" : normalCI(1,9);
    "Type 2 Financially driven planning" : normalCI(1,9);
    "Ranking of features based on different criteria" : normalCI(1,9);
    "Similarity analysis" : normalCI(1,3);
    "Dual charts using ranking and disagreement analysis" : normalCI(1,6);
    "Comparison of priorities between stakeholders" : triangular(1,1,9);
    "Import manual plan" : normalCI(1,9);
    "Import of project data" : normalCI(1,7);
    "Re-import of updated project data" : normalCI(1,6);
    "Export of plans and project data" : triangular(1,1,4);
    "Export of generated analysis charts" : normalCI(1,7);
    "Trade-off analysis" : normalCI(2,8);
    "Estimated stakeholder satisfaction analysis" : normalCI(1,6);
    "Consensus analysis between alternative plans" : normalCI(1,9);
    "Structure of alternative plans" : normalCI(1,9);
    "Quality evaluation of alternative plans" : normalCI(3,9);
    "Resource evaluation of alternative plans" : normalCI(1,6);
}
```

We model stakeholders' dissatisfaction as below:

```
Dissatisfaction = decision-subset(+) ("Next Release"){
    "Hierarchical dependencies" : normalCI(1,9);
    "Grouping of features" : normalCI(1,8);
    "Pre-assignments" : normalCI(1,9);
    "Feature dependencies" : normalCI(1,5);
    "Feasibility analysis" : normalCI(1, 4);
    "Flexible number of releases" : normalCI(1,9);
    "Flexible number and type of criteria" : normalCI(1,9);
    "Fexible number and type of resources" : normalCI(1,9);
    "Type 1 Stakekholder consensus driven planning" : normalCI(1,2);
    "Type 2 Financially driven planning" : normalCI(1,9);
    "Ranking of features based on different criteria" : normalCI(1,5);
    "Similarity analysis" : normalCI(1,7);
    "Dual charts using ranking and disagreement analysis" : normalCI(1,5);
    "Comparison of priorities between stakeholders" : normalCI(1,6);
    "Import manual plan" : normalCI(1,7);
    "Import of project data" : normalCI(1,9);
    "Re-import of updated project data" : normalCI(1,9);
    "Export of plans and project data" : triangular(1,1,6);
    "Export of generated analysis charts" :normalCI(1,9);
    "Trade-off analysis" : normalCI(1,9);
    "Estimated stakeholder satisfaction analysis" : normalCI(1,4);
    "Consensus analysis between alternative plans" : normalCI(1,6);
    "Structure of alternative plans" : normalCI(1,7);
    "Quality evaluation of alternative plans" : normalCI(1,9);
    "Resource evaluation of alternative plans" : triangular(1,7,9);
}
```

The requirements volatility is modelled below:

```
Volatility = decision-subset(+) ("Next Release"){
    "Hierarchical dependencies" : normalCI(1,4);
    "Grouping of features" : normalCI(3,8);
    "Pre-assignments" : normalCI(7,9);
    "Feature dependencies" : normalCI(5,7);
    "Feasibility analysis" : normalCI(5,6);
    "Flexible number of releases" : normalCI(6,8);
    "Flexible number and type of criteria" : normalCI(3,7);
    "Fexible number and type of resources" : normalCI(7,8);
    "Type 1 Stakekholder consensus driven planning" : normalCI(8,9);
    "Type 2 Financially driven planning" : normalCI(6,8);
    "Ranking of features based on different criteria" : normalCI(7,9);
    "Similarity analysis" : normalCI(4,6);
    "Dual charts using ranking and disagreement analysis" : normalCI(4,7);
    "Comparison of priorities between stakeholders" : normalCI(4,7);
    "Import manual plan" : normalCI(3,8);
    "Import of project data" : triangular(2,6,7);
    "Re-import of updated project data" : normalCI(1,9);
    "Export of plans and project data" : triangular(1,1,6);
    "Export of generated analysis charts" : normalCI(6,7);
    "Trade-off analysis" : normalCI(7, 8);
    "Estimated stakeholder satisfaction analysis" : normalCI(4,5);
    "Consensus analysis between alternative plans" : normalCI(7,8);
    "Structure of alternative plans" : normalCI(6,8);
    "Quality evaluation of alternative plans" : normalCI(8,9);
    "Resource evaluation of alternative plans" : normalCI(7,8);
}
```

The cost of requirements is given as:

```
Cost = BackEndDevCost
        + FrontEndDevCost
        + TestingCost
        + ProjectManagementCost
        + QACost
        + RequirementAnalysisCost;
```

We model the Back-end development cost as :

```
BackEndDevCost = decision-subset(+) ("Next Release"){
    "Hierarchical dependencies" : triangular(100, 150, 200);
    "Grouping of features" : triangular(350, 400, 450);
    "Pre-assignments" : triangular(50, 75, 100);
    "Feature dependencies" : triangular(400, 450, 500);
    "Feasibility analysis" : triangular(350, 400, 450);
    "Flexible number of releases" : triangular(350, 400, 450);
    "Flexible number and type of criteria" : triangular(550, 575, 600);
    "Fexible number and type of resources" : triangular(400, 450, 500);
    "Type 1 Stakekholder consensus driven planning" : triangular(80, 100, 120);
    "Type 2 Financially driven planning" : triangular(150, 200, 250);
    "Ranking of features based on different criteria" : triangular(350, 400, 450);
    "Similarity analysis" : triangular(100, 150, 200);
    "Dual charts using ranking and disagreement analysis" : triangular(50, 75, 100);
    "Comparison of priorities between stakeholders" : triangular(40, 50, 60);
    "Import manual plan" : triangular(50, 60, 80);
    "Import of project data" : triangular(50, 75, 100);
    "Re-import of updated project data" : deterministic(0);
    "Export of plans and project data" :triangular(200, 250, 300);
    "Export of generated analysis charts" : triangular(150, 200, 250);
    "Trade-off analysis" : triangular(40, 50, 60);
    "Estimated stakeholder satisfaction analysis" : triangular(80, 100, 120);
    "Consensus analysis between alternative plans" : triangular(80, 100, 120);
    "Structure of alternative plans" : deterministic(0);
    "Quality evaluation of alternative plans" : triangular(150, 200, 250);
    "Resource evaluation of alternative plans" : triangular(80, 100, 120);
}
```

The front end development cost is given as:

```
FrontEndDevCost = (+) ("Next Release"){
    "Hierarchical dependencies" : triangular(150, 200, 250);
    "Grouping of features" : triangular(250, 300, 350);
    "Pre-assignments" : triangular(100, 120, 140);
    "Feature dependencies" : triangular(300, 350, 400);
    "Feasibility analysis" : triangular(160, 180, 200);
    "Flexible number of releases" : triangular(100, 120, 140);
    "Flexible number and type of criteria" : triangular(400, 420, 450);
    "Fexible number and type of resources" : triangular(10, 30, 50);
    "Type 1 Stakekholder consensus driven planning" : triangular(10, 30, 50);
    "Type 2 Financially driven planning" : triangular(30, 40, 50);
    "Ranking of features based on different criteria" : triangular(40, 50, 60);
    "Similarity analysis" : triangular(30, 40, 50);
    "Dual charts using ranking and disagreement analysis" : triangular(160, 180, 200);
    "Comparison of priorities between stakeholders" : triangular(10, 30, 50);
    "Import manual plan" : triangular(5, 10, 15);
    "Import of project data" : triangular(50, 75, 100);
    "Re-import of updated project data" : deterministic(0);
    "Export of plans and project data" :triangular(80, 100, 120);
    "Export of generated analysis charts" : triangular(80, 100, 120) ;
    "Trade-off analysis" : triangular(40, 60, 80);
    "Estimated stakeholder satisfaction analysis" : triangular(80, 100, 120);
    "Consensus analysis between alternative plans" : triangular(30, 40, 50);
    "Structure of alternative plans" : triangular(50, 60, 70);
    "Quality evaluation of alternative plans" : triangular(50, 70, 90);
    "Resource evaluation of alternative plans" : triangular(80, 90, 100);
}
```

We model the testing cost as:

```
TestingCost = decision-subset(+) ("Next Release"){
    "Hierarchical dependencies" : triangular(100, 120, 150);
    "Grouping of features" : triangular(120, 150, 200);
    "Pre-assignments" : triangular(8, 10, 12);
    "Feature dependencies" : triangular(350, 375, 400);
    "Feasibility analysis" : triangular(250, 300, 350);
    "Flexible number of releases" : triangular(80, 100, 120);
    "Flexible number and type of criteria" : triangular(380, 400, 450);
    "Fexible number and type of resources" : triangular(80, 100, 120);
    "Type 1 Stakekholder consensus driven planning" : triangular(380, 400, 450);
    "Type 2 Financially driven planning" : triangular(380, 400, 450);
    "Ranking of features based on different criteria" : triangular(80, 100, 120);
    "Similarity analysis" : triangular(380, 400, 450);
    "Dual charts using ranking and disagreement analysis" : triangular(200, 225, 250);
    "Comparison of priorities between stakeholders" : triangular(200, 250, 300);
    "Import manual plan" : triangular(100, 120, 140);
    "Import of project data" : triangular(250, 300, 350);
    "Re-import of updated project data" : triangular(80, 100, 250);
    "Export of plans and project data" :triangular(350, 400, 450);
    "Export of generated analysis charts" : triangular(200, 250, 300) ;
    "Trade-off analysis" : triangular(250, 300, 350);
    "Estimated stakeholder satisfaction analysis" : triangular(100, 150, 200);
    "Consensus analysis between alternative plans" : triangular(200, 250, 300);
    "Structure of alternative plans" : triangular(80, 100, 120);
    "Quality evaluation of alternative plans" : triangular(100, 150, 200);
    "Resource evaluation of alternative plans" : triangular(250, 300, 350);
}
```

The project management cost is given as:

```
ProjectManagementCost = decision-subset(+) (”Next Release”){
    ”Hierarchical dependencies” : deterministic(0);
    ”Grouping of features” : triangular(120, 150, 200);
    ”Pre-assignments” : deterministic(0);
    ”Feature dependencies” : triangular(100, 125, 150);
    ”Feasibility analysis” : triangular(40, 50, 60);
    ”Flexible number of releases” : deterministic(0);
    ”Flexible number and type of criteria” : triangular(150, 200, 250);
    ”Fexible number and type of resources” : triangular(40, 50, 70);
    ”Type 1 Stakekholder consensus driven planning” : triangular(80, 100, 150);
    ”Type 2 Financially driven planning” : triangular(100, 150, 200);
    ”Ranking of features based on different criteria” : triangular(80, 100, 120);
    ”Similarity analysis” : triangular(100, 125, 150);
    ”Dual charts using ranking and disagreement analysis” : triangular(200, 225, 250);
    ”Comparison of priorities between stakeholders” : triangular(100, 140, 150);
    ”Import manual plan” : triangular(100, 120, 140);
    ”Import of project data” : triangular(110, 120, 150);
    ”Re-import of updated project data” : triangular(120, 150, 200);
    ”Export of plans and project data” :triangular(300, 400, 500);
    ”Export of generated analysis charts” : triangular(200, 250, 300) ;
    ”Trade-off analysis” : triangular(200, 250, 300);
    ”Estimated stakeholder satisfaction analysis” : triangular(80, 100, 120);
    ”Consensus analysis between alternative plans” : triangular( 90, 100, 110);
    ”Structure of alternative plans” : triangular(200, 250, 300);
    ”Quality evaluation of alternative plans” : deterministic(0);
    ”Resource evaluation of alternative plans” : triangular(150, 200, 250);
}
```

We model the Quality Assurance Cost as:

```
QACost = decision-subset(+) (”Next Release”){
    ”Hierarchical dependencies” : triangular(150, 200, 250);
    ”Grouping of features” : triangular(150, 200, 250);
    ”Pre-assignments” : triangular(50, 75, 100);
    ”Feature dependencies” : triangular(450, 500, 550);
    ”Feasibility analysis” : triangular(350, 400, 450);
    ”Flexible number of releases” : triangular(150, 200, 250);
    ”Flexible number and type of criteria” : triangular(200, 250, 300);
    ”Fexible number and type of resources” : triangular(350, 400, 450);
    ”Type 1 Stakekholder consensus driven planning” : triangular(30, 40, 50);
    ”Type 2 Financially driven planning” : triangular(30, 50, 70);
    ”Ranking of features based on different criteria” : triangular(30, 40, 50);
    ”Similarity analysis” : triangular(300, 400, 450);
    ”Dual charts using ranking and disagreement analysis” : triangular(250, 300, 350);
    ”Comparison of priorities between stakeholders” : triangular(100, 200, 250);
    ”Import manual plan” : triangular(100, 190, 200);
    ”Import of project data” : triangular(400, 450, 500);
    ”Re-import of updated project data” : triangular(80, 100,120);
    ”Export of plans and project data” :triangular(300, 400, 500);
    ”Export of generated analysis charts” : triangular(200, 250, 300) ;
    ”Trade-off analysis” : triangular(200, 250, 300);
    ”Estimated stakeholder satisfaction analysis” : triangular(200, 300, 320);
    ”Consensus analysis between alternative plans” : triangular( 190, 200, 210);
    ”Structure of alternative plans” : triangular(100, 150, 200);
    ”Quality evaluation of alternative plans” : triangular(80, 100, 150);
    ”Resource evaluation of alternative plans” : triangular(150, 200, 250);
}
```

The requirements analysis cost is given as:

```
RequirementAnalysisCost = decision-subset(+) ("Next Release"){
    "Hierarchical dependencies" : triangular(40, 60, 80);
    "Grouping of features" : triangular(120, 150, 200);
    "Pre-assignments" : triangular(10, 20, 50);
    "Feature dependencies" : triangular(150, 200, 250);
    "Feasibility analysis" : triangular(100, 150, 200);
    "Flexible number of releases" : triangular(5, 10, 15);
    "Flexible number and type of criteria" : triangular(150, 200, 250);
    "Fexible number and type of resources" : triangular(1,5, 10);
    "Type 1 Stakekholder consensus driven planning" : triangular(50, 100, 150);
    "Type 2 Financially driven planning" : triangular(30, 50, 70);
    "Ranking of features based on different criteria" : triangular(10, 20, 30);
    "Similarity analysis" : triangular(100, 150, 200);
    "Dual charts combining using and disagreement analysis" : triangular(50, 60, 70);
    "Comparison of priorities between stakeholders" : triangular(50, 60, 70);
    "Import manual plan" : triangular(30, 40, 50);
    "Import of project data" : triangular(30, 50, 70);
    "Re-import of updated project data" : triangular(40, 50, 60);
    "Export of plans and project data" :triangular(20, 50, 60);
    "Export of generated analysis charts" : triangular(80, 100, 120) ;
    "Trade-off analysis" : triangular(50, 100, 150);
    "Estimated stakeholder satisfaction analysis" : triangular(10, 25, 30);
    "Consensus analysis between alternative plans" : deterministic(0);
    "Structure of alternative plans" : triangular(40, 50, 60);
    "Quality evaluation of alternative plans" : triangular(40, 50, 60);
    "Resource evaluation of alternative plans" : triangular(40, 50, 60);
}
```

We model the constraints relationships between requirements below:

Constraint "Next Release" : "Pre-assignments" **couples**"Next Release" : "Feature dependencies";

Constraint "Next Release" : "Pre-assignments" **couples**"Next Release" : "Feasibility analysis;

Constraint "Next Release" : "Feature dependencies" **couples**"Next Release" : "Feasibility analysis;

Constraint "Next Release" : "Flexible number and type of criteria" **couples**"Next Release": "Trade-off analysis;

Constraint "Next Release" : "Dual charts combining ranking and disagreement analysis" **requires**"Next Release" : "Ranking of features based on different criteria;

Constraint "Next Release" : "Import manual plan" **requires**"Next Release" : "Import of project data;

Constraint "Next Release" : "Re-import of updated project data" **requires**"Next Release" : "Import of project data;

Constraint "Next Release" : "Export of plans and project data" **requires**"Next Release" : "Import of project data;

Constraint "Next Release" : "Export of generated analysis charts" **requires**"Next Release" : "Import of project data;

Constraint "Next Release" : "Export of generated analysis charts" **requires**"Next Release" : "Trade-off analysis;

Constraint "Next Release" : "Export of generated analysis charts" **requires**"Next Release" : "Estimated stakeholder satisfaction analysis;

Constraint "Next Release" : "Export of generated analysis charts" **requires**"Next Release" : "Consensus analysis between alternative plans;

Constraint "Next Release" : "Export of generated analysis charts" **requires**"Next Release" : "Structure of alternative plans;

Constraint "Next Release" : "Export of generated analysis charts" **requires**"Next Release" : "Quality evaluation of alternative plans;

Constraint "Next Release" : "Export of generated analysis charts" **requires**"Next Release" : "Resource evaluation of alternative plans;

7.2.4.3 Analysis Result

Optimisation Analysis

The RADAR analysis of the RADAR models developed for the Commercial Release Planning tools is presented Table 7.7. The problem was analysed using a multi-objective evolutionary algorithm, i.e., NSGAII, since exhaustive strategy was infeasible. NSGAII was run using the 1+λ optimisation approach [130] and algorithmic parameters similar to the settings used in [285]: population size of 100, crossover probability of 0.9, mutation probability of 0.1 and maximum number of fitness evaluation of 50000.

The results in Table 7.7 show that four solutions were shortlisted out of a total of 2^{25} solutions. None of the shortlisted solutions suggests including all features in the next release, but they suggest 5 common features, such as: Hierarchical dependencies, Ranking of features based on different criteria, Dual charts combining ranking and disagreement analysis, Import of project data, Quality evaluation of alternative plans. Once these 5 solutions are selected, the shortlist includes all possible combinations of other features; each combination representing a trade-off between maximising Expected-NetBenefit, minimising ProjectRisk, maximising ExpectedFrequencyOfUse, minimising ExpectedDissatisfaction and minimising ExpectedRequirementVolatility.

Information Value Analysis

The EVTPI is £6.14m and EVPPI for all model parameters is approximately equal to zero. This means that none of the model parameters is worth further data collection or analysis.

Optimisation Analysis

Objective:	**Max** ExpectedNetBenefit
Objective:	**Min** ProjectRisk
Objective:	**Max** ExpectedFrequencyOfUse
Objective:	**Min** ExpectedDissatisfaction
Objective:	**Min** ExpectedRequirementVolatility
Solution Space:	2^{25}
Optimisation Approach:	1+λ
Algorithm Name:	NSGAII
Population Size:	100
Crossover Probability:	0.9
Mutation Probability:	0.1
Nbr. Fitness Evaluations:	50000
Shortlisted:	4
Nbr. Decisions:	1
Nbr. Variables:	25
Runtime(s) :	38

	Solution 1	Solution 2	Solution 3	Solution 4
	Hierarchical dependencies	Hierarchical dependencies	Hierarchical dependencies	Hierarchical dependencies
	Flexible number of releases	Flexible number of releases	Fexible number and type of resources	Flexible number and type of criteria
	Type 1 Stakekholder consensus driven planning	Type 1 Stakekholder consensus driven planning	Type 1 Stakekholder consensus driven planning	Trade-off analysis
	Type 2 Financially driven planning	Type 2 Financially driven planning	Type 2 Financially driven planning	Similarity analysis
	Ranking of features based on different criteria	Ranking of features based on different criteria	Dual charts combining ranking and disagreement analysis	Dual charts combining ranking and disagreement analysis
Features in the Next Release	Dual charts combining ranking and disagreement analysis	Dual charts combining ranking and disagreement analysis	Export of plans and project data	Consensus analysis between alternative plans
	Comparison of priorities between groups of stakeholders	Comparison of priorities between groups of stakeholders	Ranking of features based on different criteria	Ranking of features based on different criteria
	Import manual plan	Import manual plan		Import manual plan
	Import of project data	Import of project data	Import of project data	Import of project data
	Structure of alternative plans	Structure of alternative plans	Structure of alternative plans	Quality evaluation of alternative plans
	Export of plans and project data	Pre-assignments	Pre-assignments	
	Estimated stakeholder satisfaction analysis	Feature dependencies	Feature dependencies	
	Re-import of updated project data	Grouping of features	Grouping of features	
	Quality evaluation of alternative plans	Quality evaluation of alternative plans	Quality evaluation of alternative plans	
	Resource evaluation of alternative plans	Feasibility analysis	Feasibility analysis	
ExpectedNetBenefit	5652	8607	8123	6231
ProjectRisk	1	1	1	1
ExpectedFrequencyOfUse	45	63	70	64
ExpectedDissatisfaction	42	54	59	60
ExpectedRequirementVolatility	60	87	95	90

TABLE 7.7: Optimisation Analysis and Information Value analysis results of requirements subset selection problem for the Release Planning tool.

7.2.4.4 Comparison To Previous Analysis Approaches

Several analysis of the NRP have been proposed in the literature [30, 283, 285]. Many of these analyses have been described in Chapter 2.3.2.

The Release planning system has been previously used in the literature to study the next release problem. Kareem et al [142] used the system as a case study in evaluating a proposed theme-based release planning method that supports delivering software releases that contain features that are related in the value they deliver to stakeholders. Pitangueira et al. [202] also used the same system to evaluate their risk-aware multi-objective next release problem (MONRP) approach that reformulates the MONRP with an extra objective that caters for stakeholders dissatisfaction risk.

Like many other NRP models, the models developed by Kareem et al [142] and Pitangueira et al. [202] for the release planning system used generic decision objectives and pre-established model equations (e.g. typically weighted sums) to specify "cost" and "value" scores to alternative solutions. The model developed by Kareem et al. ignored uncertainty in model parameters and does not analyse uncertainty. However, Pitangueira et al. modelled uncertainty as the variance in the stakeholders estimates of requirements attributes (costs and values).

With respect to the optimisation analysis, like RADAR, Kareem et al. used the multi-objective optimisation algorithm (NSGAII) to shortlist Pareto optimal solutions. Pitangueira et al. used both approximate and exact multi-objective optimisation algorithm: the authors used NSGAII to generate the initial set of Pareto optimal solutions and then used SMT solver (Z3 [68] and Yice [77]) to shortlist exact Pareto optimal by transforming the problem to an SMT problem and encoding candidate requirements as Boolean variables and solving the model using an SMT solver. Using NSGAII with crossover probability of 0.9, mutation probability of 0.1 and number of generation as 500, RADAR optimisation analysis and the optimisation approach used in Kareem et al. shortlisted 4 solutions out of 2^{25}. The reason for the same number of solutions could be that they both used NSGAII implementation in JMetal. While the analysis approach used in Pitangueira et al. also used NSGGII, the authors did not report the number of solutions shortlisted by NSGAII, but Yices and Z3 shortlisted 146 and 143 exact solutions, respectively.

In terms of information value analysis, RADAR includes such analysis to estimate the financial value of reducing uncertainty in the model. Such analysis is not available in the approaches proposed by Kareem et al. and Pitangueira et al.

7.2.5 Public Bike Sharing System

7.2.5.1 Problem Statement

The public bike sharing system is deployed in many metropolitan cities, such as London, to increase travellers' transit options, reduce energy utilisation, improve the quality of life by reducing air and noise pollution, and minimise traffic congestion within the city.

Chapter 3.2 described in details the different components of the bike sharing system. Some of these decisions are the bikes security strategy to use in securing the bikes; bike manufacturer brand to deploy; the type of bike docking station to use; decision about alternative ways by which users can register to use the system; decision about user access to the system; decision about rewarding users to return bikes to inconvenient locations; decision about the method to get status updates from the system.

7.2.5.2 RADAR Model

Modelling the Optimisation Objectives

Two key concerns of the bike sharing system are to maximise the net benefit of introducing the system to the city and to minimise the project risk.

```
Objective  Max ExpectedNetBenefit = EV(NB);
Objective  Min Risk = Pr(NB < 0);
```

The net benefit (NB) is defined as the difference between Benefit and Cost.

```
    NB = Benefit − Cost;
```

```
Cost = CostOfBikes
           + CostOfSecuringBicycles
           + CostOfDockStations
           + CostOfSystemAccessMgt
           + CostSystemRegistrationMgt
           + CostOfOtherComponents;
```

The Cost is composed of the costs of implementing the system, which include the cost of securing the bike, cost of bikes, cost of a dock station, and the cost of other sub-systems such as the registration component and the system access management. In Section 3.4.2, we have presented a partial decision model developed for the bike sharing system with focus on elaborating the different costs components of the system and the constraint relationships between the options of decisions. This section elaborates on the first objective optimisation objectives: System NetBenefit.

We model the system Benefit as:

```
Benefit = BenefitOfSecuringBicycles
              + BenefitOfDockStations
              + BenefitOfSystemAccessMgt
              + BenefitOfSystemRegistrationMgt
              + BenefitOfNonMandatorySystemComponents;
```

```
BenefitOfSecuringBicycles = decision-subset(+)("Securing Bicycles"){
        "Localisation feature" : BenefitOfBikeLocalisation;
        "Anti-theft feature" : triangular(2,5,10);
}
BenefitOfBikeLocalisation = decision-subset(+)("Tracking Mechanism"){
        "GPS feature" : triangular(5,10, 15);
        "RFID feature" : triangular(20, 22, 25);
}
BenefitOfDockStations = decision("Dock Station"){
        "Permanently Fixed" : triangular(15,17, 19);
        "Temporarily Fixed " : triangular(20,22,25);
        "Flexible " : triangular(24,27,30);
}
BenefitOfSystemAccessMgt = decision-subset(+)("System Access Mgt"){
        "Smart card" : triangular(20,25, 30);
        "Smart Phone" : triangular(15, 17,20);
        "Key Card" : triangular(29, 30,35);
}
BenefitOfSystemRegistrationMgt = decision-subset(+)("System Registration Mgt"){
        "Kisok Reg" : BenefitOfKioskReg;
        "Dock Station Reg" : triangular(18, 20,22);
        "Web Reg" : triangular(27, 30,33);
}
BenefitOfKioskReg = decision-subset(+)("Kisok Registration"){
        "Touch Screen" : triangular(2, 5, 7);
        "Key card reader" : triangular(8, 10,12);
        "Credit Card" : triangular(10, 12,15);
        "Card Dispenser" : triangular(12, 15, 18);
}
BenefitOfNonMandatorySystemComp = decision-subset(+)("NonMandatorySystemComp"){
        "System Status Info" : BenefitOfHavingSysStatusInfo;
        "Bike Maintenance" : triangular(12,15, 20);
        "Bike Redistribution" : BenefitOfBikeRedistribution;
}
```

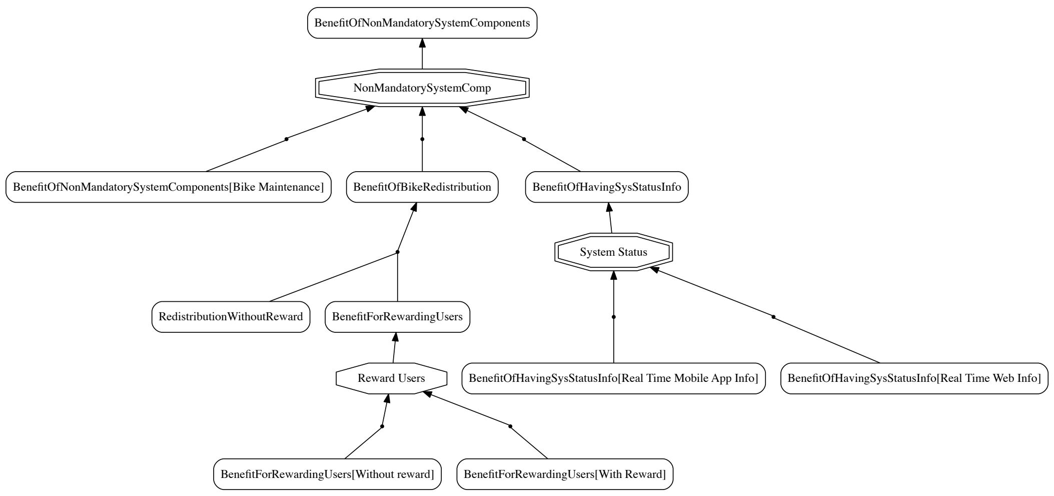

BenefitOfNonMandatorySystemComponents
NonMandatorySystemComp
BenefitOfNonMandatorySystemComponents[Bike Maintenance]
BenefitOfBikeRedistribution
BenefitOfHavingSysStatusInfo
System Status
RedistributionWithoutReward
BenefitForRewardingUsers
Reward Users
BenefitOfHavingSysStatusInfo[Real Time Mobile App Info]
BenefitOfHavingSysStatusInfo[Real Time Web Info]
BenefitForRewardingUsers[Without reward]
BenefitForRewardingUsers[With Reward]

FIGURE 7.8: Partial AND/OR refinement graph for the model variable BenefitOfNon-ManadatorySystemComponents of the bike sharing model.

```
BenefitOfHavingSysStatusInfo = decision-subset(+)(”System Status”){
        ”Real Time Web Info” : triangular(5, 10, 10);
        ”Real Time Mobile App Info” : triangular(15, 20, 30);
}
BenefitOfBikeRedistribution = RedistributionWithoutReward + BenefitForRewardingUsers;
RedistributionWithoutReward = triangular(10,15, 20);
BenefitForRewardingUsers = decision(”Reward Users”){
        ”Without reward” : deterministic(0);
        ”With Reward” : triangular(25, 28, 30);
}
```

To help visualise the model structure, RADAR generates the AND/OR refinement graph and decision dependency graphs from the bike sharing model equations. Fig.7.8 shows the partial AND/OR refinement graphs for the bike sharing model starting from the model variable BenefitOfNonManadatorySystemComponents. Fig.3.4 in Chapter 3 shows a partial model decision dependency graph for the bike sharing model.

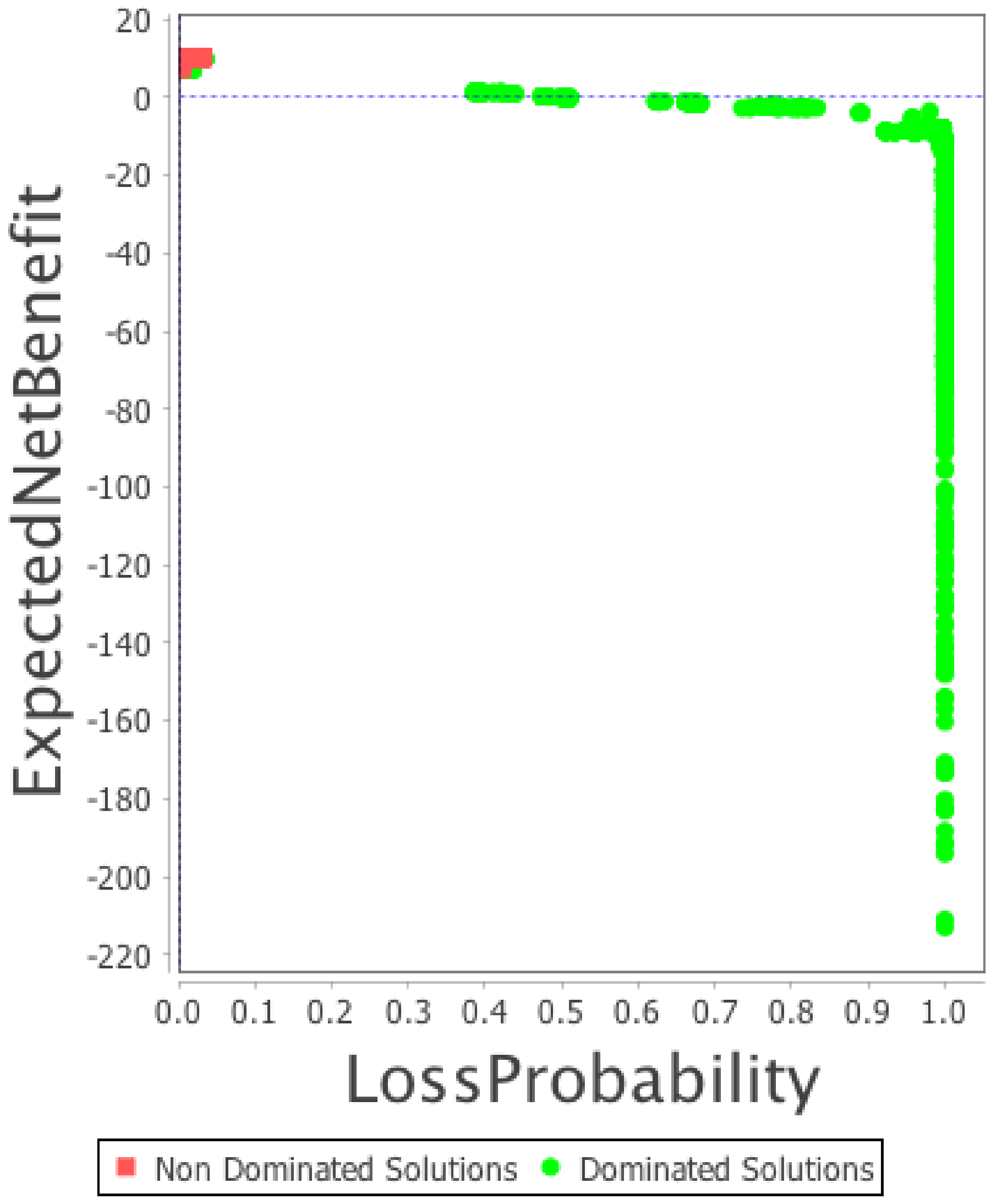


FIGURE 7.9: Pareto front for the bike sharing system problem.

7.2.5.3 Analysis Results

The RADAR analysis of the bike sharing model is presented in Fig. 7.8, which shows the results of the optimisation and information value analysis on the model.

The first part of Fig.7.8 is the optimisation analysis result. RADAR shortlisted 35 solutions out of 15×2^{20} possible alternatives using a multi-objective evolutionary algorithm, i.e., NSGAII, since exhaustive strategy was infeasible. NSGAII was run using the $1+\lambda$

optimisation approach [130] described in Chapter 5.1.4 and used algorithmic parameters similar to the settings used in [285] : population size of 100, crossover probability of 0.9, mutation probability of 0.1 and maximum number of fitness evaluation of 50000. All shortlisted solutions include the "A-Bike" option. With this option selected, the shortlisted solutions include different possible combinations of the bike security decision, tracking mechanism decision, system registration decision and other non mandatory system component (e.g., Bike maintenance, system status information and bike redistribution). Each combination represents a trade-off between maximising the expected net benefit and minimising risk. RADAR generates the graph in Fig.7.9, plotting the objective values for the shortlisted solutions (shown squares at the top of the graph) and all other non shortlisted ones (shown as circles).

The second part of Fig.7.9 is the information value analysis results, which show that the EVTPI for this problem is £0.81m. and EVPPI for the unit cost (UnitCost) of A-Bike brand to be £0.03m, and the EVPPI of the number of bikes to deploy (NbrBicyclesToDeploy) to be £0.70m. This means that reducing uncertainty about the number of bikes to deploy has a higher value than reducing uncertainty about the cost of the A-Bike brand

Optimisation Analysis

Objective:	**Max** ExpectedNetBenefit
Objective:	**Min** Risk
Solution Space:	20×10^{20}
Optimisation Approach:	$1+\lambda$
Algorithm Name:	NSGAII
Population Size:	100
Crossover Probability:	0.9
Mutation Probability:	0.1
Nbr. Fitness Evaluations:	50000
Shortlisted:	36
Nbr. Decisions:	10
Nbr. Variables:	36
Runtime(s) :	11

ID	Securing Bicycles	Tracking Mechanism	Dock Station	System Registration Mgt	Kisok Reg	NonMandatorySystemComp	System Status	Reward Users	Manufacturer Brand	ExpectedNetBenefit	Risk
1	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	Without reward	A-Bike	9.97	0.01
2	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Temporarily Fixed	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	Without reward	A-Bike	10.3	0.03
3	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	Without reward	A-Bike	7.47	0
4	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	Without reward	A-Bike	7.79	0
5	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Credit Card	System Status Info	—	Without reward	A-Bike	7.47	0
6	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	7.79	0
7	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Touch Screen;Credit Card	System Status Info	—	Without reward	A-Bike	9.97	0.01
8	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	System Status Info	—	Without reward	A-Bike	10.3	0.03
9	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Key card reader;Credit Card	—	—	Without reward	A-Bike	7.47	0
10	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Key card reader;Card Dispenser	System Status Info	—	Without reward	A-Bike	7.47	0
11	Localisation feature	RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	System Status Info	—	Without reward	A-Bike	5.13	0
12	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	With Reward	A-Bike	7.79	0
13	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Credit Card	System Status Info	—	Without reward	A-Bike	9.97	0.01
14	Localisation feature	RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	5.13	0
15	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Key card reader;Credit Card	—	—	Without reward	A-Bike	9.97	0.01
16	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	With Reward	A-Bike	9.97	0.01
17	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	System Status Info	—	Without reward	A-Bike	7.79	0
18	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	7.47	0
19	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	System Status Info	—	Without reward	A-Bike	7.79	0
20	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Key card reader;Card Dispenser	—	—	Without reward	A-Bike	9.97	0.01
21	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Key card reader;Credit Card	—	Real Time Web Info;Real Time Mobile App Info	With Reward	A-Bike	7.47	0
22	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Credit Card	—	—	Without reward	A-Bike	7.79	0
23	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	—	—	Without reward	A-Bike	7.79	0
24	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Touch Screen;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	With Reward	A-Bike	7.79	0
25	Localisation feature	RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	—	—	Without reward	A-Bike	5.13	0
26	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	With Reward	A-Bike	7.47	0
27	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Key card reader	System Status Info	—	Without reward	A-Bike	7.47	0
28	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Temporarily Fixed	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	With Reward	A-Bike	10.3	0.03
29	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Temporarily Fixed	Kisok Reg	Credit Card	System Status Info	—	With Reward	A-Bike	10.3	0.03
30	Localisation feature	RFID feature	Temporarily Fixed	Kisok Reg	Touch Screen;Card Dispenser	System Status Info	—	Without reward	A-Bike	5.13	0
31	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Credit Card	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	9.97	0.01
32	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Flexible	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	9.97	0.01
33	Localisation feature;Anti-theft feature	RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	—	—	With Reward	A-Bike	7.79	0
34	Localisation feature;Anti-theft feature	GPS feature;RFID feature	Temporarily Fixed	Kisok Reg	Key card reader;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	Without reward	A-Bike	10.3	0.03
35	Localisation feature;Anti-theft feature	RFID feature	Flexible	Kisok Reg	Touch Screen;Card Dispenser	—	Real Time Web Info;Real Time Mobile App Info	With Reward	A-Bike	7.47	0

TABLE 7.8: Optimisation Analysis results for the public bike sharing model

7.2.5.4 Comparison To Previous Analysis Approaches

Previous modelling and analyses of the public bike sharing system (BSS) have focused generally on developing feature models that capture variability and commonality of the different configurations of the system.

Ter et al. [249] identified the main components, features, commonalities and variability of the system by text mining a series of documents from the literature [177], that detail the public bike sharing system. Based on the features identified, the authors developed the bike sharing feature models using different tools: SPLOT [175] (which allows editing, debugging, analysing, configuring, sharing, and downloading feature models in simple XML feature model formats); FeatureIDE [145] –a similar tool, but allows generation of feature models in the graphical Feature Oriented Domain Analysis (FODA) syntax as well as Java or C++ codes; and Clafer [18] –a modelling language that supports feature and domain modelling, configuration and verification. ClaferMOO [192] extends Clafer with a feature to enable exact multi-objective optimisation.

The bike sharing model developed in SPLOT, FeatureIDE and Clafer/ClaferMOO describe the variability and commonality of features of the bike sharing system. The Clafer/ClaferMOO model allows specification of optimisation objectives (maximising the customer satisfaction, minimise cost, maximise security and maximise capacity) in the model and generates Alloy model that contains attributed feature model and solved using Alloy solvers [139, 178]. Since Clafer/ClaferMOO requires model transformations. This may lead to model synchronisation complexity and a risk of model inconsistencies.

The bike sharing models developed in previous work used weighted sums of feature attribute values (cost, defect count, performance etc). They do not allow the elaboration of domain-specific decision models that RADAR supports through AND/OR refinements of model variables. Also, existing feature modelling tools and frameworks do not allow one to explicitly capture uncertainty in domain quantities and cannot handle Boolean expressions, such as logical OR and logical AND, that RADAR handles. Finally, they generally lack techniques for analysing uncertainty and informing decision makers about the financial value of reducing uncertainty in a decision model.

As a comparison to the RADAR optimisation results, ClaferMOO shortlisted 249 solutions using exact multi-objective optimisation whereas RADAR shortlisted 36 solutions through evolutionary multi-objective optimisation algorithm (NSGAII). Unlike Clafer-MOO, RADAR estimates EVTPI and EVPPI about model parameters to determine which aspects of the model requires further analysis.

7.3 Conclusions From Our Experiments

7.3.1 Applicability

By successfully applying our modelling language and decision analysis technique to a wide range of requirements and architecture decisions, we have shown that:

- **Claim 1:** The RADAR modelling language is expressive enough to model real-world requirements and architecture decision problems;
- **Claim 2:** The RADAR optimisation technique can be applied to real-world requirements and architecture decision problems.

With respect to Claim 1, RADAR gives requirements engineers and software architects the ability to elaborate domain specific requirements and architecture decision problems. For example, in the bike sharing model presented in Section 7.2.5, we elaborated the total cost of bikes to depend on other domain quantities such as the number of additional bikes to deploy; the number of bikes currently deployed, and the unit cost of a bike. Such fine-grained elaboration of model equations are necessary to capture stakeholders' real objectives and ultimately ensuring decision-makers make the right decisions. One limitation, however, is that the current implementation of RADAR does not cater for mathematical functions such as AVERAGE, MAX, MIN and SQRT. As a consequence, RADAR is not currently applicable to the requirements and architecture decision problems that require these mathematical functions. For example in the LAS model presented in Appendix A, RADAR was not able to model the Euclidean distance between an ambulance and the incident location using an Euclidean function between two points. This is a limitation to addresses in future work to increase RADAR's expressiveness.

7.3.2 Usefulness

We have illustrated RADAR's modelling capability and its automated decision analysis technique on problems described in Table 7.1. This section highlights RADAR's usefulness in supporting requirements and architecture decisions under uncertainty. Our discussion of usefulness is, at this stage, speculative.

RADAR provides decision-makers with useful information about the trade-offs between alternatives. It gives information about what objective values can be attained with different design alternatives. For example, in the fraud detection example of Section 7.2.1, Fig. 7.2 helps decision-makers to understand the trade-off between maximising the fraud detection benefit and minimising the investigation load. The figure shows four shortlisted alternatives (shown squares at the top of the graph) and all other non shortlisted ones (shown as circles). Similarly, in the emergency response system example described in Section 7.2.2, RADAR generates Fig 7.4 to show the trade-off between maximising expected net benefit and minimising the project risk. In the bike sharing example presented in Section 7.2.5, Fig 7.9 helps to understand the trade-off between maximising the expected net benefit and minimising risk.

RADAR also identifies the different decision-options combinations that represent the trade-off between stakeholders' objectives. For example, in the fraud detection optimisation result presented in Table 7.2, RADAR shortlists four solutions each with different decision-option combinations: $s_1 = \{($**blocking policy**, block first), (**processing type**, $\{$continuous$\}$), (**fraud detection method**, $\{$rule-based$\}$); $s_2 = \{($**blocking policy**, block first), (**processing type**, $\{$continuous$\}$), (**fraud detection method**, $\{$classifier$\}$), (**alert threshold**, $\{$medium$\}$); $s_3 = \{($**blocking policy**, block first), (**processing type**, $\{$continuous$\}$), (**fraud detection method**, $\{$classifier$\}$), (**alert threshold,** $\{$high$\}$); $s_4 = \{($**blocking policy**, block first), (**processing type**, $\{$continuous$\}$), (**fraud detection method**, $\{$classifier$\}$), (**alert threshold**, $\{$low$\}$).

RADAR helps to identify the decisions that are better than the others in a given decision model. For example, the results presented in Table 7.2 for the fraud detection system shows that all shortlisted solutions include the "block first" policy and "continuous" processing type. This suggests to decision-makers that a decision with "block first" policy and "continuous" processing type options is better than one without these options. Similarly, for the emergency response system presented in Table 7.4, RADAR suggests to decision-makers that a decision with "radio triangulation", "OpenIntent", "Implicit", "XMPP (Open Fire)", "Preloaded (ESRI)" is better than a decision without these options.

RADAR provides useful information about what parameter uncertainty may deserve additional data collection and analysis before making their decision and what parameter uncertainty does not matter to their decision. For example, RADAR's information value analysis results of the fraud detection example in Table 7.2 shows that the EVTPI for the fraud detection problem is 220 and EVPPI for AverageFraudValue is 122. All other parameters have EVPPI below 2. This means that in this model, the average fraud value is the only parameter worth further investigation before making the final decision. Reducing uncertainty about other parameters would bring no value to the decision. Consider also Table 7.6 which shows the EVTPI and EVPPI for all model parameters of the building security policy model to be zero. In this case, RADAR suggests to the stakeholders that their parameter estimates are accurate and that there is no need for additional data collection or analysis before making their decision.

Finally, RADAR provides useful graphical representations to visualise the decision models through AND/OR refinements graphs commonly used in goal-oriented requirements engineering. It also presents the decision dependency graphs which shows model decisions, their corresponding options and the relationships between decisions and options. These graphs help to communicate and validate traceability links between strategic stakeholder goals and technical software characteristics [264]. They can also be used to review the model structure with other non technical stakeholders. For example, in the fraud detection model, RADAR generates the AND/OR graph shown in Fig 7.1. The graph starts from a model variable Benefit and AND-refined into BaseLineFinancialLoss and FinancialLosss until reaching leaf variables that are parameter estimations. Figure 7.5 shows a similar refinement graph for the building security policy model presented in Section 7.2.3. The CostOfDisclosure is AND-refined into NbrHighConfidentialLeaks, CostOfHighConfidentialLeaks, NbrMediumConfidentialLeaks, CostOfMediumConfidentialLeaks, NbrLowConfidentialLeaks and CostOfLowConfidentialLeaks. Each sub-variable are refined until reaching the leaf variables.

7.3.3 Threats To Validity

7.3.3.1 The model validity threats

In our approach, the correctness of the analysis results is relative to the validity of the decision model. If the model's equations are not valid, the predicted objective values for the different solutions might be wrong. In the twelve examples, although some equations and parameters estimation are based on observed data, we have mostly validated our models by checking that our equations 'make sense' rather than testing them empirically. We cannot therefore guarantee their validity.

One should observe that when making decisions about systems that have yet to be built, it will in general not be possible to validate all equations empirically because some of the equations will refer to phenomena that cannot be observed yet. It will only be possible to validate these equations empirically once the system is in use. This is an inherent difficulty of requirements and architecture decision problems.

With respect to the problem of model validation, our approach needs to be compared with the state-of-the-art in requirements and architecture decision making that, by relying on fixed, predefined, and unfalsifiable equations, ignore the issue of model validity. By contrast, RADAR models can be criticised, reviewed, and modified to improve their validity.

7.3.3.2 The cost of modelling threats

Another possible problem of our method is that the cost of elaborating the decision models might outweigh its benefits. Our objective in designing RADAR was to reduce the difficulty and cost of modelling compared to existing approaches that require the model to be developed in a general purpose programming language. We have, however, not yet tested how easily people will be able to use our language and tool.

With respect to cost-effectiveness, a potential benefit of our approach is that it enables an iterative modelling and analysis approach where information value analysis might be used to decide what parts of an initially simple model (such as the refactoring model in Chapter 3.4) should be refined to improve decisions. This will reduce modelling cost by helping modellers develop fine-grained models only where needed and leave other parts of the problem modelled at a coarse level of granularity. We intend to develop and evaluate such iterative approach in future work.

7.4 Summary

We have applied RADAR's modelling language and decision analysis technique on twelve real-world requirements and architecture decision problems from different application domains. We have also compared the RADAR's analysis results with other requirements and architecture decision analysis technique to emphasise the benefits of using RADAR over the state-of-the-art decision analysis techniques. These benefits include: (i) the ability to use simple mathematical equations to elaborate domain specific decision models that capture stakeholders' concerns. (ii) The automated decision analysis technique that RADAR provides with the ability to determine which model parameters need further investigation before making the final decision. (iii) The graphical representation of decision models (AND/OR graphs and decision dependency graphs) to give decision-makers insights into the relationships between high-level objectives and low-level technical details of a model. The following chapter studies the performance of RADAR's optimisation algorithms.

Information Value Analysis

Objective:	**Max** ExpectedNetBenefit
EVTPI:	0.81

Parameter	EVPPI
BenefitOfSecuringBicycles[Anti-theft feature]	0.7
BenefitOfBikeLocalisation[GPS feature]	0.7
BenefitOfBikeLocalisation[RFID feature]	0.7
BenefitOfDockStations[Permanently Fixed]	0.7
BenefitOfDockStations[Temporarily Fixed]	0.7
BenefitOfDockStations[Flexible]	0.7
BenefitOfSystemAccessMgt[Smart card]	0.7
BenefitOfSystemAccessMgt[Smart Phone]	0.7
BenefitOfSystemAccessMgt[Key Card]	0.7
BenefitOfSystemRegistrationMgt[Dock Station Reg]	0.7
BenefitOfSystemRegistrationMgt[Web Reg]	0.7
BenefitOfKioskReg[Touch Screen]	0.7
BenefitOfKioskReg[Key card reader]	0.7
BenefitOfKioskReg[Credit Card]	0.7
BenefitOfKioskReg[Card Dispenser]	0.7
BenefitOfNonMandatorySystemComp[Bike Maintenance]	0
BenefitOfHavingSysStatusInfo[Real Time Web Info]	0
BenefitOfHavingSysStatusInfo[Real Time Mobile App Info]	0
RedistributionWithoutReward	0
BenefitForRewardingUsers[With Reward]	0
NbrBikesToDeploy	0.7
UnitCost[A-Bike]	0.03
UnitCost[Cortina Cycles]	0.03
UnitCost[Derby Cycle]	0.03
UnitCost[Bianchi]	0.03
UnitCost[Catrike]	0.03
CostOfSecuringBicycles[Anti-theft feature]	0.03
CostOfBikeLocalisation[GPS feature]	0.03
CostOfBikeLocalisation[RFID feature]	0.03
CostOfDockStations[Permanently Fixed]	0.7
CostOfDockStations[Temporarily Fixed]	0.7
CostOfDockStations[Flexible]	0.7
CostOfSystemAccessMgt[Smart card]	0.7
CostOfSystemAccessMgt[Smart Phone]	0.7
CostOfSystemAccessMgt[Key Card]	0.7
CostSystemRegistrationMgt[Dock Station Reg]	0.7
CostSystemRegistrationMgt[Web Reg]	0.7
CostOfKioskReg[Touch Screen]	0.7
CostOfKioskReg[Key card reader]	0.7
CostOfKioskReg[Credit Card]	0.7
CostOfKioskReg[Card Dispenser]	0.7
CostOfNonMandatorySystemComponents[Bike Maintenance]	0.7
CostOfHavingSysStatusInfo[Real Time Web Info]	0.7
CostOfHavingSysStatusInfo[Real Time Mobile App Info]	0.7
RedistributionCostWithoutReward	0.03

Table 7.9: Information Value Analysis results for the Bike Sharing model

Chapter 8

RADAR Performance Evaluation

This chapter presents an empirical evaluation that aims to examine the scalability of RADAR's exhaustive strategy and the performance of RADAR's evolutionary search-based algorithms.

8.1 Research Questions

Following standard experimental methodology [80] described in section 8.3, we evaluate RADAR's optimisation algorithms by answering the following research questions:

RQ1 (Scalability): What is the scalability of RADAR's exhaustive strategy? Adding inclusive OR decisions to a decision model significantly increases the size of the solution space, and therefore may make the use of exhaustive strategy infeasible. This research question provides insight on the scalability of RADAR's exhaustive strategy during simulation and optimisation of a problem design space. We measure RADAR's running time and memory consumption on large RADAR synthetic models. We perform experiments to answer the following sub-research questions:

- **RQ1.1:** What is the scalability of RADAR's exhaustive search strategy with respect to the number of simulations?
- **RQ1.2:** What is the scalability of RADAR's exhaustive search strategy with respect to the size of the design space?

- **RQ1.3:** What is the scalability of RADAR's exhaustive search strategy with respect to the number of objectives?
- **RQ1.4:** What is the time spent and memory consumed by each analysis step?

RQ2 (Performance Analysis): What is the performance of RADAR's alternative search-based evolutionary algorithms? Since many elitist evolutionary multi-objective algorithms (EMOAs), such as NSGAII, SPEAII, IBEA, differ in how they evolve solutions between generations and in how they estimate dominance value between two solutions, we perform experiments to answer the following research questions:

1. **RQ2.1:** What is the execution time of RADAR's optimisation analysis using different evolutionary multi-objective algorithms?
2. **RQ2.2:** What is the quality of solutions shortlisted by RADAR using different evolutionary multi-objective algorithms?

8.2 RADAR Models Analysed

We have analysed the RADAR tool on both real world requirements and architecture decision models and synthetic RADAR models. Using synthetic models helps to examine the scalability of RADAR's exhaustive strategy.

8.2.1 Real Models

Our experiments used real-world requirements and architecture decision problems presented in Chapter 7 and summarised in Table 7.1: decisions about features to implement for the next release of a commercial release planning system and a word processor system [142]; design decisions of a system to coordinate the deployment of emergency response teams [85, 166]; architecture decisions for the NASA satellite processing system designed to collect and process satellite images [151, 179]; decisions about selecting an optimal set of features in a product family the public Bike Sharing System [177, 250]; a PHP-based framework for web content management [228]; an e-commerce System 67 [176];

Core Model Constructs	**Value Ranges**	**Step Size**
Objectives	[2-5]	1
Decisions	[10-20]	1
Options Per Decision	[3-10]	1
Min of Variables	[100-1000]	100

Table 8.1: Ranges of values used to generate RADAR synthetic models.

Amazon Web Service (AWS) elastic compute cloud [98]; Berkeley Relational Database Management System [238]. These problems have design space size between 10^3 and 2^{40}.

8.2.2 Synthetic Models

We have implemented a synthetic model generator that generates random syntactically valid RADAR models with any given number of objectives, decisions, number of options per decisions and minimum number of model variables. The model generator can produce RADAR models with or without decision dependencies. It constructs a synthetic RADAR model following RADAR's syntax: a model's objective is declared as either a maximisation or a minimisation problem, whose definition could be an expectation or a probability defined over a random variable. For each objective, the random variable that defines it is refined into three child variables which are related by randomly chosen arithmetic operators (e.g. "+", "–", "/" and "*"). The three child variables are typically of the form *ANDRefinement*, *ParameterEstimation* and *OrRefinement*, respectively. Following the same format, each child variable is further refined and related to three variables until both the specified number of decisions and minimum number of model variables have been attained. For synthetic models without decision dependencies, each expression corresponding to the individual option of an *OrRefinement* is always a parameter estimation. However, for models with decision dependencies, each expression corresponding to the individual option of an *OrRefinement* could be an *ANDRefinement*, *ParameterEstimation* or *OrRefinement*. The number of decision dependencies specified determines the number of times an *OrRefinement* variable is linked to the option of another.

The model generator and all models generated for the experiments below are available from the tool's website (`https://ucl-badass.github.io/radar/`).

8.3 Experimental Methodology

We answer **RQ1** by applying RADAR on randomly generated RADAR models using different combinations of RADAR's core model constructs, i.e., the number of objectives, number of decisions, options per decision, AND/OR variables and parameter estimations. We measure RADAR's running time and memory consumption. Table 8.1 shows the range of values for each model constructs used to generate the synthetic models. The number of objectives is between 2 and 5; the number of decisions ranges from 10 to 20, inclusive; the number of options per decision is between 3 and 10; and the minimum number of model variables is between 100 and 1000. All model construct values are increased by a unit step size, except the minimum number of model variables which is increased in steps of 100.

For **RQ2**, we experiment on the real-world requirements and architecture problems presented in Table 8.3 using four Evolutionary Multi-objective Optimisation Algorithms (EMOAs), such as NSGAII [69], SPEAII [289], MoCell [184] and IBEA [288], the two optimisation approaches RADAR employs in handling constraints, i.e., (λ+1) and 1+λ, and compare results against random search. We chose the four EMOAs because they have already been implemented in JMetal5 [185] and have been used extensively in many multi-objective software engineering research problems [88, 126, 130, 157, 229, 230, 235, 285].

For each studied stochastic multi-objective optimisation algorithm, we follow recommended practices [27, 269] and assess the quality of solutions shortlisted by measuring widely used metrics, such as hypervolume and coverage [100, 130, 204, 235], to provide information about the convergence and diversity of solutions in the Pareto front approximation. To investigate the number of valid solutions (i.e., solutions that satisfy the constraints declared in a RADAR model) shortlisted in the Pareto front approximation, we measured another metric called validity ratio [130]. However, since the optimal Pareto front cannot always be known, we follow the standard practice of using a *reference front* that combines the best solutions produced by all EMOAs studied.

- **Hypervolume (HV).** The hypervolume of a set of solutions can intuitively be understood as a measure of how far the boundary created by a set of solutions is from some reference point corresponding to the objective function values for the worst possible solution (for example, in a maximisation problem, a solution with zero values for all objective functions). Higher values of HV indicate better Pareto front approximations.

- **Coverage (Cov).** This is the ratio between the size of Pareto front approximation in the exact Pareto optimal solutions to the total number of exact Pareto optimal solutions. It measures how close the approximate Pareto front is to the exact Pareto front in the solution space. Higher values of coverage depict better Pareto front approximations.

- **Validity Ratio (VR).** This is the ratio of the number of solutions in the Pareto front approximation, that satisfy the model constraint relationship, on average for the independent runs that have at least one valid solution. We compute this metric only for problems that have constraints. Higher values of VR implies efficacy of an EMOA in finding valid solutions in the Pareto front approximations.

To compare these metrics across the five algorithms, we conduct a post-hoc analysis with two-way comparison that includes *statistical differences* and *effect sizes*. For statistical differences, we use the non-parametric Man-Whitney U-test, at 5% significance level, to report the p-values i.e. the probabaility that two EMOAs give different values. For effect sizes, we apply the Vargha-Delaney A-measure [251] to give the probability that a particular EMOA outperforms another.

8.4 Experimental Settings

All our experiments were run on a computer with a four-core 2.6 GHz processor and 7GB RAM. For the multi-objective optimisation algorithm parameter settings such as population size, crossover and mutation probabilities, we used single point crossover and bit flip mutation. For the default parameter settings, we used similar parameter levels in [285], which is a crossover probability of 0.8, mutation probability to be the inverse of the total number of options in a decision model, and a population size of 100. All algorithms terminate after 50,000 fitness function evaluations. We compute fitness values through 10^4 simulations of a particular RADAR solution to obtain simulation values for the objective functions. Because stochastic multi-objective evolutionary algorithms include randomness, following guidelines in [27, 269], we have run each experiment 30 independent times on each RADAR model.

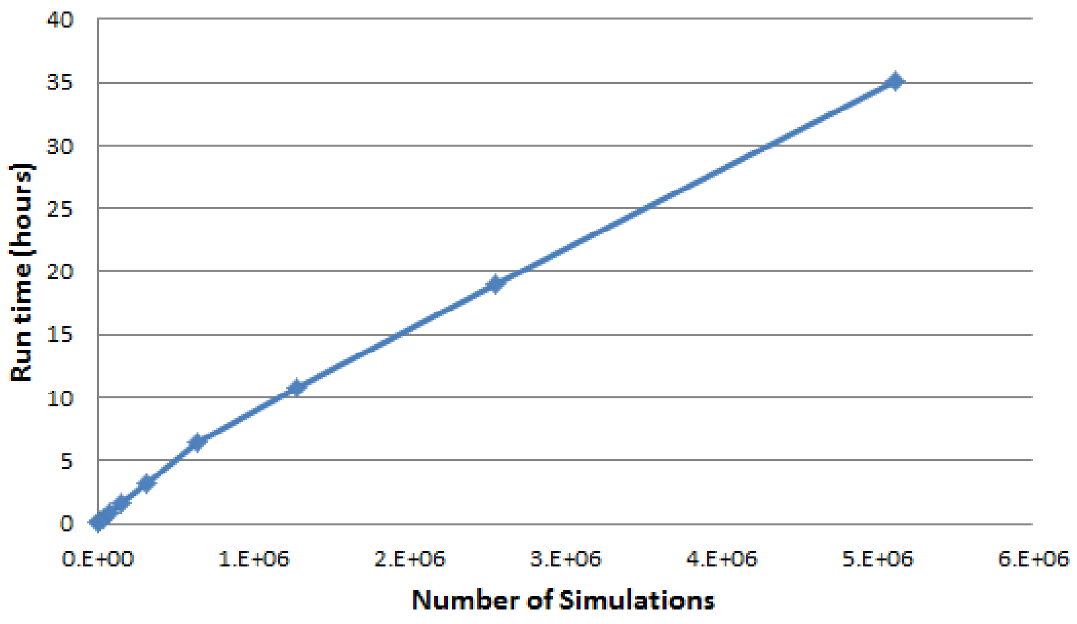
8.5 Results and Analysis

8.5.1 Scalability of RADAR Exhaustive Strategy

RQ1.1: What is RADAR's Scalability with respect to the number of simulations? To evaluate how RADAR run-time and memory usage increases as the number of simulation, N, increases, we have generated a synthetic model whose characteristics are similar to that of the emergency response system, i.e. it contains 2 objectives, 10 decisions, 3 options per decisions, and no decision dependencies. We have then measured the running times and memory consumption of analysing this model when doubling N 10 times from 10^4 to 512×10^4. The results are shown in Figure 8.1 and indicate that the running time and memory usage increase linearly with N.

RQ1.2: What is RADAR's scalability with respect to design space size? To evaluate how RADAR run-time and memory usage increases when the design space size increases, we have generated synthetic models with decision dependencies by incrementally and separately increasing the number of decisions and options per decisions until the resulting models could no longer be analysed in less than an hour. The synthetic models generated have 2 objectives and at least 100 model variables. Figure 8.2 shows the result of this experiment. RADAR was able to evaluate in less than one hour models with a design space of up to 153,751 solutions. The model with the largest design space included 11 decisions with 7 options and was analysed in 35 hours. The figure also shows that on our synthetic models the run-time and memory usage increase roughly linearly with the size of the design space.

RQ1.3: What is RADAR's scalability with respect to the number of objectives? To evaluate how RADAR run-time and memory usage increases when the number of objectives increases, we have generated synthetic models with 10 decisions, 3 options per decisions, and incrementally increased the number of objectives from 2 to 5 until the resulting models could no longer be analysed in less than an hour. The synthetic models generated do not have decision dependencies. Figure 8.3 shows that on our synthetic



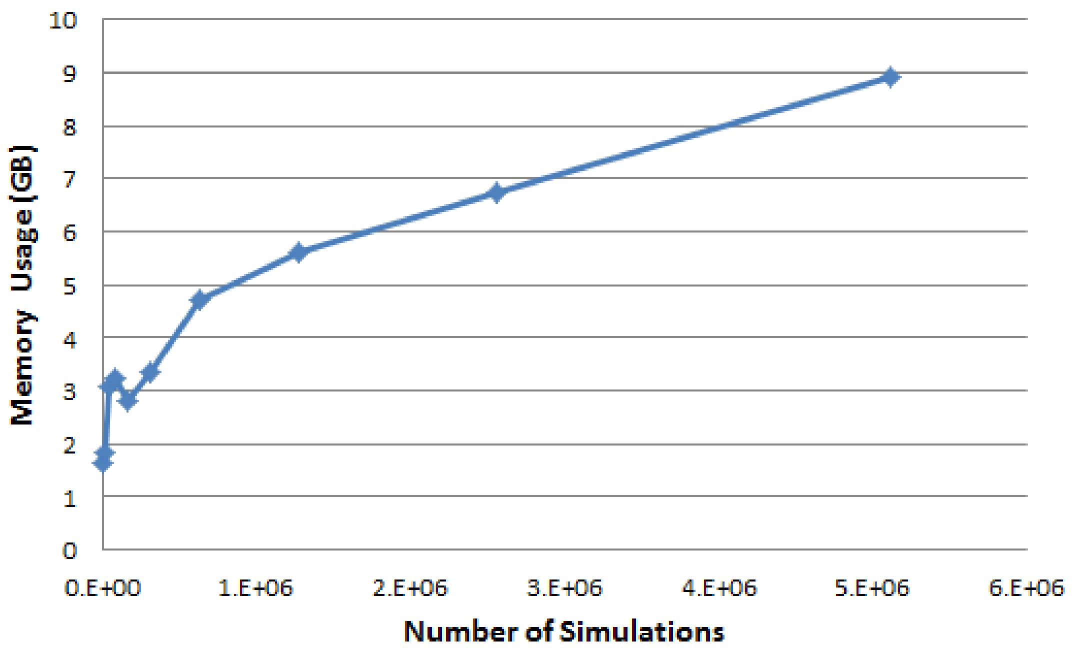
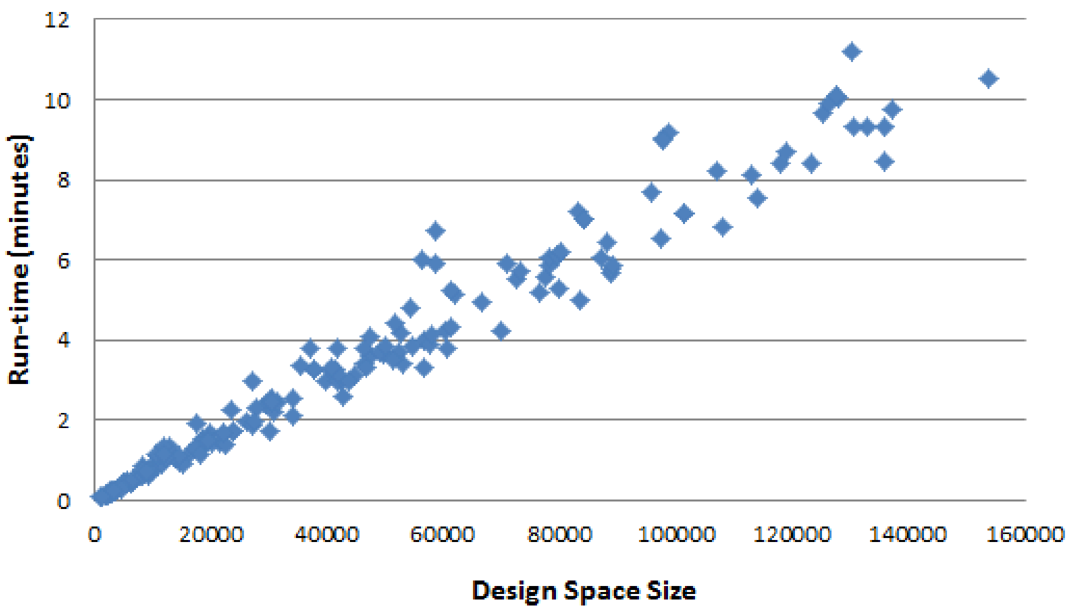


FIGURE 8.1: Total run-time (left) and memory usage (right) measured for doubling the number of simulations, N, from 10^4 to 512×10^4.



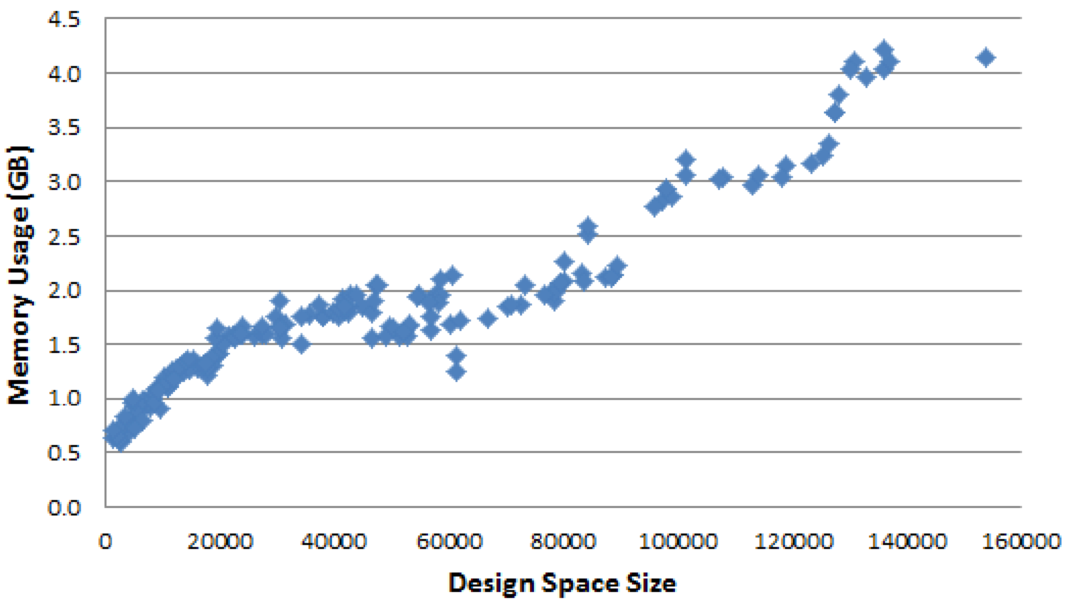
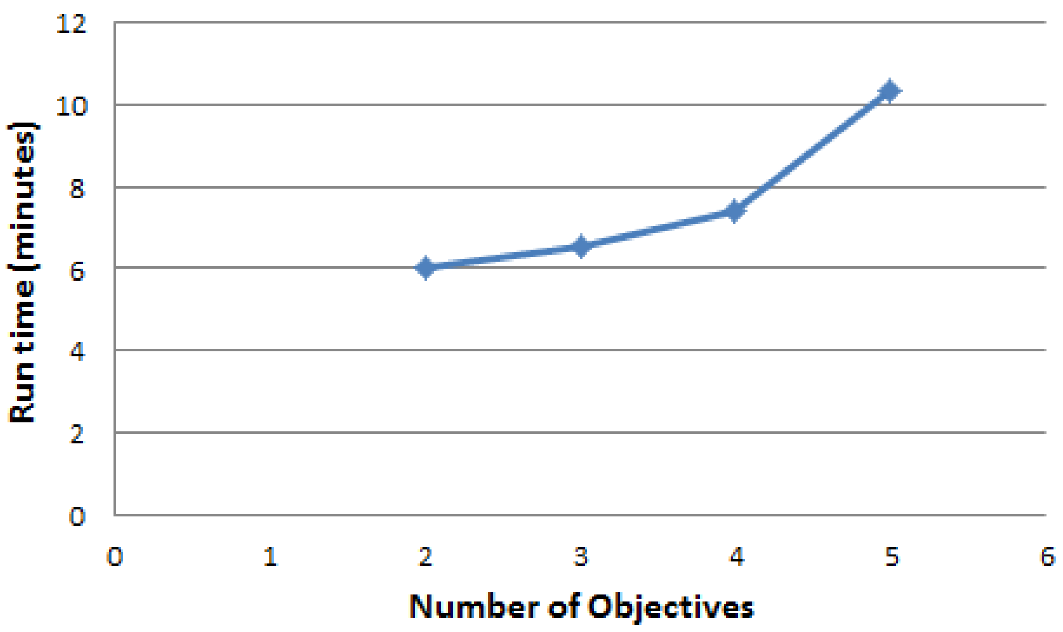


FIGURE 8.2: Total run-time (left) and memory usage (right) measured for 180 RADAR models with different design space size.



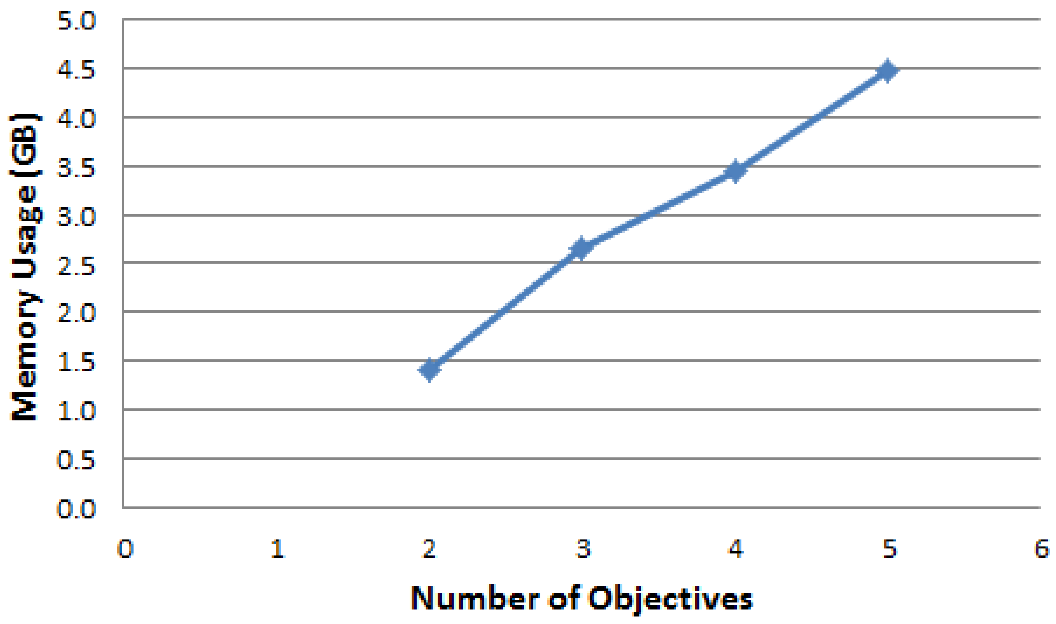


FIGURE 8.3: Total run-time (left) and memory usage (right) measured for 2,3,4 and 5 objectives.

models the run-time and memory usage increase roughly linearly with the number of objectives.

RQ1.4: What is the time spent and memory consumed by each analysis step? For each synthetic model generated in the experiment to answer RQ1.4, we measured the fraction of time spent and memory used in each of the four analysis step: generating

Algorithm Step	Average % Total Time	Average % Memory Usage
Generating the design space	0	0
Simulating all solutions in the design space	100	96
Shortlisting the Pareto-optimal solutions	0	1
Computing expected information value over the shortlisted solutions	0	3

Table 8.2: Average fraction of time for each analysis step over all synthetic models [43].

Model	**System Description**	**Objectives**	**#Decisions (#XOR/#OR)**	**#Options Ranges**	**#Variables**	**#Constraints**	**Solution space**
MSM	Requirements subset selection for the future release of a Microsoft Word Processor [142]	Max Net Benefit Min Risk	0/1	50	251	39	2^{50}
RPM	Requirements subset selection for the future release of a commercial decision support tool for release planning [142]	Max Project Benefit Max Frequency Of Use Min Risk Min Stakeholder Dissatisfaction Min Requirements Volatilty	0/1	25	267	15	2^{25}
SAS	Software architecture evaluation of an emergency response system to coordinate teams in emergency situations [85, 166]	Max Net Benefit Min Risk	10/0	2-4	254	0	6912
ECS	Software architecture evaluation of a satellite Image Processor for collecting and managing satellite data [151, 179]	Max Project Utility Min Cost	10/0	2	86	0	2^{10}
BDM	Optimal feature selection of the Berkeley Relational Database Management System [238]	Max Net Benefit Min Resource Utilisation	1/2	2-6	27	0	2^{9}
BSS	Optimal feature selection of a public Bike Sharing System [177, 250]	Max Net Benefit Min Loss Probability	2/8	2-5	74	4	15×2^{20}
AWM	Amazon Web Service Elastic Compute Cloud optimal configuration [98]	Max Feature Richness Max Instance ECU Max EC2 Cores Max Instance RAM Max SSD Backed Min CostHour Min CostMonth	12/2	2-5	68	0	405×2^{10}
WPM	Optimal feature selection of a Web Portal System [176]	Max Feature Richness Max Feature Reuse Min Defect Count Min Cost	6/7	2-5	135	21	3×2^{26}
DPM	Optimal feature selection in Drupal— a PHP-based framework for web content management [228]	Max Net Benefit Min Risk	4/6	2-24	27	0	2^{40}

Table 8.3: RQ2: Real-world requirements and architecture decision problems

the design space, simulating the design space, shortlisting the Pareto-optimal solutions, and computing expected information value. Table 8.2 shows the average fraction of time for each analysis step over all synthetic models. The table shows that the simulation of all solutions takes the largest portion of time (100%) and memory consumption (96%).

In summary, our results show that: the run time and memory consumption of the radar analysis steps are linearly proportional to the number of simulations (N), the design space size and the number of objectives; the simulation of the design space takes the highest average proportion of the run time (100%) and memory usage (96%); the design space of models without decision dependencies increases exponentially with the number of decisions and options. As a rule-of-thumb, radar's exhaustive search strategy would struggle solving problems with more than 10 independent decisions with around 3 options per decisions.

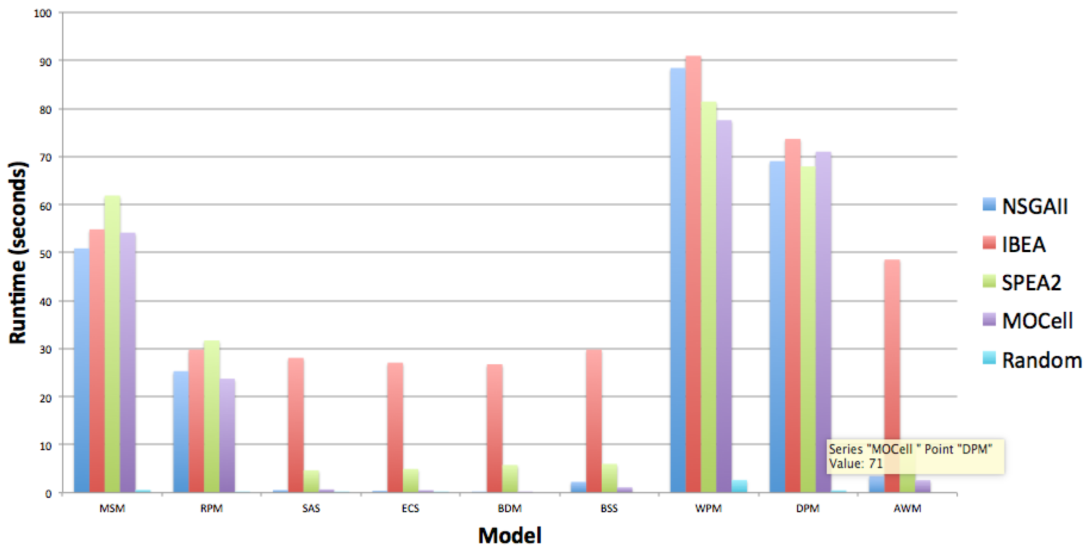


FIGURE 8.4: $(\lambda + 1)$ optimisation approach.

8.5.2 Performance Analysis of RADAR Search-Based Approaches

Figures 8.4 and 8.5 show the runtimes between the $(\lambda+1)$ and $1+\lambda$ approaches and for all model-algorithm pairs. Table 8.4 shows the mean value of hypervolume (HV), coverage (COV) and validity ratio (VR) quality indicators for each problem-algorithm pair. In Table 8.4, the best and second best values of the quality indicators for each model-algorithm pair are shaded dark grey and light grey, respectively. We report validity ratio for problems with constraints only.

RQ2.1. Time taken by RADAR's optimisation analysis using different EMOAs? The mean run-time results in Figure 8.4 and Figure 8.5 show that all algorithms finished under 2 minutes. Between the $(\lambda + 1)$ and $1 + \lambda$ approaches and for all model-algorithm pairs, IBEA is the slowest. This is because IBEA carries out extra computation when computing dominance value between two candidate solutions. IBEA considers optimisation objectives (which captures user's preferences) in assigning weights to candidate solutions based on the quality indicator (hypervolume). When comparing the other algorithms, the run-time varies, with the exception that random search was the fastest. Therefore, we conclude that RADAR with EMOAs scales well in analysing models that are infeasible to solve using the exhaustive strategy of RADAR.

RQ2.2. Quality of solutions shortlisted by RADAR using different EMOAs?

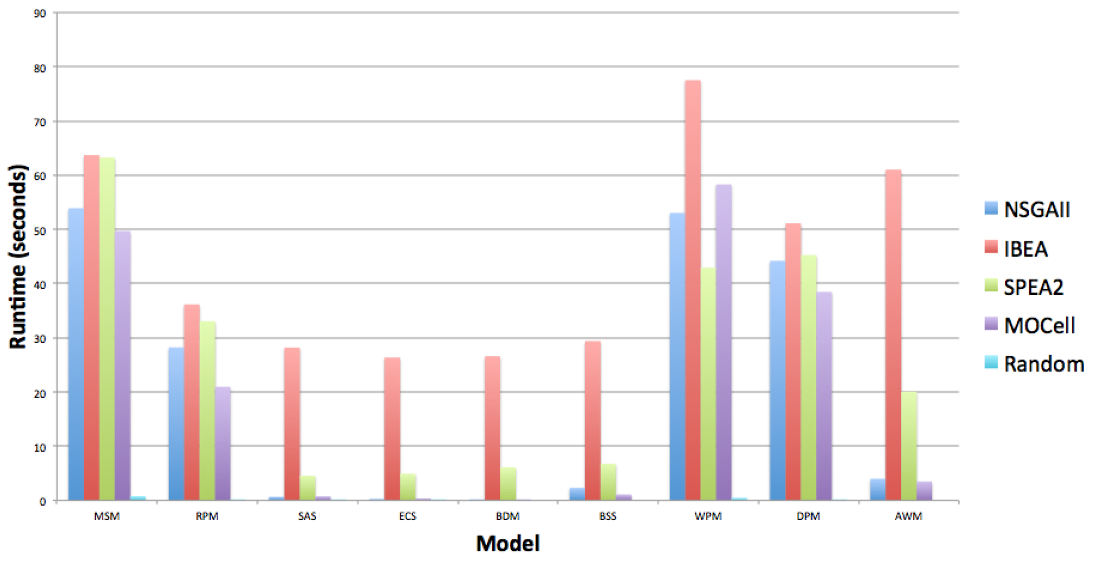


FIGURE 8.5: $1 + \lambda$ optimisation approach.

The results presented in Table 8.4 show that between the $(\lambda{+}1)$ and $1{+}\lambda$ approaches and for all model-algorithm pairs, SPEA2 and MOCell outperformed NSGAII and IBEA in mean HV and Cov except in model WPM where IBEA produced superior results than the other EMOAs. Between SPEA2 and MOCell, there is no clear winner. Between NSGAII and IBEA, the former mostly outperformed the latter in mean Cov and mean HV. For BDM model, all EMOAs produced HV of 0. Further investigation revealed that the number of solutions with unique objective values in the Pareto front approximations is very small (at most three) for each EMOA. The low HV means that the Pareto front approximation are not able to cover a significant part of the objective space.

The post-hoc analysis of two way comparison between alternative RADAR algorithms is presented in Table 8.5. This analysis shows statistical evidence to support these findings. Specifically, we carried out a total of 360 unique tests (9 problems, 5 algorithms, 2 metrics, 2 optimisation approaches) in which 324 tests (90%) showed statistical significance (p-value < 0.05) in the difference between alternative RADAR EMOAs, with 72% of the tests having large effect size (A-measure ≥ 0.8).

From the results in Table 8.4 and Table 8.5, we conclude that between the $(\lambda + 1)$ and $1 + \lambda$ approaches and for all model-algorithm pairs, SPEA2 and MOCell produced the best Pareto front approximation in terms of proximity to the reference (true) Pareto front and the spread of solutions in the Pareto front.

		SPEA2		**MOCell**		**NSGAII**		**IBEA**		**Random**	
	Model	$(\lambda+1)$	$1+\lambda$	$(\lambda+1)$	$1+\lambda$	$(\lambda+1)$	$1+\lambda$	$(\lambda+1)$	$1+\lambda$	$(\lambda+1)$	$1+\lambda$
Cov	MSM	0.1240	0.0378	0.1490	0.0111	0.0089	0.0111	0.0063	0.0156	0.0000	0.0000
	RPM	0.0078	0.0250	0.0124	0.0181	0.0076	0.0008	0.0076	0.0056	0.0738	0.0047
	SAS	1.0000	1.0000	1.0000	1.0000	0.8610	0.8520	0.0111	0.0143	0.2750	0.2830
	ECS	1.0000	1.0000	1.0000	1.0000	0.8210	0.7900	0.0864	0.0778	0.7440	0.7200
	AWM	1.0000	1.0000	1.0000	1.0000	0.7440	0.7560	0.0646	0.0729	0.4420	0.5060
	BDM	0.9330	1.0000	1.0000	0.9670	0.7670	0.7000	0.1330	0.1670	0.7670	0.8830
	BSS	1.0000	1.0000	0.9670	0.9440	0.7670	0.9670	0.5000	0.5560	0.0000	0.0000
	DPM	0.0105	0.0210	0.0114	0.0534	0.0064	0.0017	0.0113	0.0132	0.0041	0.0000
	WPM	0.0098	0.0138	0.0255	0.0270	0.0303	0.0072	0.0586	0.0661	0.0073	0.0008
HV	MSM	0.4820	0.0120	0.4920	0.0305	0.3620	0.0249	0.3490	0.0186	0.2860	0.0010
	RPM	0.1400	0.1740	0.1590	0.1670	0.1430	0.0331	0.1380	0.0803	0.1710	0.0968
	SAS	0.4400	0.5900	0.4400	0.5900	0.4260	0.5840	0.0972	0.1520	0.3070	0.4640
	ECS	0.7200	0.7200	0.7200	0.7200	0.7160	0.7140	0.6380	0.6430	0.7080	0.7120
	AWM	0.0163	0.0163	0.0163	0.0163	0.0154	0.0153	0.0089	0.0010	0.0154	0.0147
	BDM	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
	BSS	0.3870	0.2450	0.3740	0.2370	0.3100	0.2450	0.2870	0.2110	0.0000	0.0000
	DPM	0.3700	0.3610	0.3490	0.3430	0.3240	0.2030	0.3800	0.3050	0.3050	0.1530
	WPM	0.2120	0.2220	0.2010	0.2130	0.2090	0.1550	0.2370	0.2310	0.1860	0.1720
VR(%)	MSM	0.17	0	2	0	0.20	0	0.17	0	0	0
	RPM	6.33	100	5.94	100	5.87	100	63.60	100	6.72	100
	BSS	100	100	98.30	100	98.30	100	100	100	98.30	100
	DPM	19.10	100	16.80	100	15.50	100	82.20	100	6.65	100
	WPM	41.30	100	33.10	100	40.70	100	100	100	26.20	100

Table 8.4: Mean coverage, mean hypervolume and validation ratio between $(\lambda + 1)$ and $1 + \lambda$ for all model-algorithm pairs (30 runs). The best and second best values of the metrics for each model-algorithm pair are shaded dark grey and light grey, respectively. MSM (Microsoft Word Processor model), RPM (Release Planning tool model), SAS (Situation Awareness System model), ECS (NASA Satellite processing system model), AWM (Amazon Web Service model), BDM (Building Security model), BSS (Bike sharing model), DPM (Drupal PHP System model), WPM (E-commerce Web portal system). We report validity ratio for problems with constraints only.

With respect to the number of valid solutions shortlisted in the Pareto front approximation, the mean VR presented in Table 8.4 shows that all algorithms returned at least one valid solutions in each model for both $(\lambda + 1)$ and $1 + \lambda$ optimisation approaches. The only exception is observed in MSM model where the mean VR for each EMOAs is 0 for the $1+\lambda$ approach. This means using the $1+\lambda$ approach, none of the EMOAs was able to generate optimal solution that satisfy model constraints with 50000 evaluations of the problem solution space (2^{50}). We also observed in general that between $(\lambda + 1)$ and $1 + \lambda$ optimisation approaches, all the EMOAs produced very high mean VR ($\approx$ 100%) with the $1+\lambda$ approach than $(\lambda+1)$ approach. Of these EMOAs, IBEA has the best mean VR with the $(\lambda + 1)$. This is because IBEA incorporates decision-maker's objectives in the Pareto dominance criteria.

Therefore, we conclude that independent of the EMOA used during RADAR analysis, the $1+\lambda$ approach is more effective than the $(\lambda+1)$ approach in shortlisting valid solutions in the Pareto front approximation. This is because the $1 + \lambda$ optimisation approach prioritises solutions with fewer constraint violations when estimating dominance value between two candidate solutions. And when more valid solutions are generated, they are likely maintained across different generations thereby serving as a seed to generate further valid solutions. This finding is consistent with those obtained in [130] for $1 + \lambda$ approach.

		($\lambda + 1$) optimisation approach					**$1 + \lambda$ optimisation approach**				
	Indicator	**SPEA2**	**NSGAII**	**MOCell**	**IBEA**	**Random**	**SPEA2**	**NSGAII**	**MOCell**	**IBEA**	**Random**
SPEA2	**Cov**		MSM, SAS, ECS, BSS, DPM, AWM		MSM, SAS, ECS, BDM, BSS, AWM	MSM, SAS, ECS, BDM, BSS, WPM, DPM, AWM		MSM, RPM, SAS, ECS, BDM, BSS, WPM, DPM, AWM	MSM, RPM, BSS	MSM, RPM, SAS, ECS, BDM, BSS, DPM, AWM	MSM, RPM, SAS, ECS, BDM, BSS, WPM, DPM, AWM
	HV		MSM, SAS, ECS, BSS, WPM, DPM, AWM	WPM, DPM	MSM, SAS, ECS, BSS, AWM	MSM, SAS, ECS, BSS, WPM, DPM, AWM		RPM, SAS, ECS, BSS, WPM, DPM, AWM	BSS, WPM	RPM, SAS, ECS, BSS, DPM, AWM	MSM, RPM, SAS, ECS, BSS, WPM, DPM, AWM
NSGAII	**Cov**	WPM		WPM	SAS, ECS, BDM, BSS, WPM, DPM, AWM	MSM, SAS, BSS, WPM, DPM, AWM				SAS, ECS, BDM, BSS, AWM	MSM, SAS, BSS, WPM, DPM, AWM
	HV			WPM	MSM, SAS, ECS, WPM, AWM	MSM, SAS, ECS, BSS, WPM, DPM, AWM				SAS, ECS, BDM, BSS, AWM	MSM, SAS, BSS, WPM, DPM, AWM
MOCell	**Cov**	RPM, WPM	MSM, RPM, SAS, ECS, BDM, BSS, DPM, AWM		MSM, RPM, SAS, ECS, BDM, BSS, WPM, AWM	MSM, SAS, ECS, BSS, WPM, DPM, AWM	WPM, DPM	RPM, SAS, ECS, BDM, BSS, WPM, DPM, AWM		RPM, SAS, ECS, BDM, BSS, DPM, AWM	MSM, RPM, SAS, ECS, BSS, WPM, DPM, AWM
	HV	RPM	MSM, RPM, SAS, ECS, BSS, DPM, AWM		MSM, RPM, SAS, ECS, BSS, AWM	MSM, SAS, ECS, BSS, WPM, DPM, AWM		RPM, SAS, ECS, WPM, DPM, AWM		RPM, SAS, ECS, BSS, DPM, AWM	MSM, RPM, SAS, ECS, BSS, WPM, DPM, AWM
IBEA	**Cov**	WPM, DPM	DPM			MSM, BSS, WPM, DPM	WPM	RPM, WPM, DPM	WPM		MSM, BSS, WPM, DPM
	HV	WPM, DPM		WPM, DPM		MSM, WPM, DPM	WPM	RPM, WPM, DPM	WPM		MSM, BSS, WPM, DPM
Random	**Cov**	RPM	RPM	RPM	RPM, SAS, ECS, BDM, AWM			RPM		SAS, ECS, BDM, AWM	
	HV	RPM	RPM	RPM	RPM, SAS, ECS, AWM			RPM		SAS, ECS, AWM	

TABLE 8.5: Post-hoc analysis results of pairwise comparison of model-algorithm pair. Each cell in the table are models for which algorithms in the rows are significantly ($p < 0.05$) better than those in the columns.

8.6 Threats to Validity

8.6.1 Internal validity threats

This concerns the biases in the reported empirical results due to the random nature of Evolutionary Multi-objective Optimisation Algorithms (EMOAs). We handled this threat by following guidelines in [27]. We run all studied EMOAs (i.e. NSGAII, SPEAII, MoCell and IBEA) on our subject models 30 times. We also checked for differences in statistical significance and effect sizes in the achieved results using Man-Whitney U-test and Vargha-Delaney A-measure [251], respectively. In our experiments, we have not set random seeds that would have ensured that running each EMOAs multiple times on a model generate exactly the same Pareto-optimal solutions. Instead, running each EMOAs 30 times on each model enable us to study the variability of Pareto-optimal solutions due to the algorithms' randomness. For practical use of the tool, modellers can set a specific random seed to eliminate such variability if they wish.

Another possible threat to internal validity is in the choice of algorithmic parameter settings, which could have affected the results obtained. Our evaluation uses well known default algorithmic parameters for EMOAs previously used in many multi-objective optimisation problems in software engineering [193, 235, 285]. One could also criticise the validity of the metrics we have used to compare our EMOAs. In particular, we have focused on hypervolume, coverage, and validity ratio used in previous similar studies [128, 130, 193], but have ignored other measures of solutions diversity, such as Spread and generational distance [290].

8.6.2 External validity threat

This concerns the limitations of generalising from our results about the applicability and usefulness of RADAR in modelling all requirements and architecture decision problems. We reduced this threat by performing our evaluations on 9 subject models which are based on real world systems used in the literature [85, 98, 142, 151, 166, 176, 177, 179, 228, 238, 250]. These problems are characterised with different complexities, such as different number of objectives, different number of decisions, different number of options per decision, different number of constraints between options of decisions, different number of model variables, and different sizes of the solution space. Nonetheless, our results may not generalise to all requirements and architectural decision problems as more validations still need to be carried out in the future.

8.7 Summary

This chapter presented an evaluation of RADAR's performance analysis. We have conducted an empirical study to assess the performance of RADAR's exhaustive strategy and its alternative search-based multi-objective evolutionary algorithms.

We conducted an empirical scalability analysis of the RADAR's exhaustive strategy on synthetic RADAR models, and measured RADAR's execution time and memory consumed with respect to increasing number of simulations; number of objectives and the design space. The results of the study (see section 8.5.1) show the design space size limits RADARs exhaustive simulation, thereby limiting the class of problems RADAR's exhaustive strategy can analyse.

We also conducted an empirical study to assess the execution time and quality of solutions generated by RADAR's evolutionary multi-objective algorithms, such as NSGAII [69], SPEAII [289], MoCell [184] and IBEA [288] (see section 8.5.2). Our results show that: (i) these algorithms scale well with real-world requirements and architecture decision problems of different complexities (see figures 8.4 and 8.5); (ii) independent of the evolutionary multi-obejective algorithm used with RADAR, the $1 + \lambda$ approach is better than the $(\lambda + 1)$ approach; and (iii) between the $(\lambda + 1)$ and $1 + \lambda$ constraint handling approaches and between the algorithms, SPEA2 and MOCell produced the best Pareto front approximation in terms of convergence and diversity across different decision models (see tables 8.4 and 8.5).

Chapter 9

Conclusion and Future Work

In software engineering, requirements engineering and software architecture are key phases of engineering software intensive systems. They involve making critical decisions that have an impact on the software project cost, schedule and the ability of the system to deliver business values and satisfy stakeholders goals.

Determining the optimal requirements and architectural decisions becomes complex when decision-makers have to deal with multiple conflicting objectives including, for example, minimising cost, maximising value, and minimising risks; uncertainty about the impact of decision choices on objectives and a huge space of alternative solutions which are difficult to explore manually.

Several requirements and architectural decision support methods have been proposed. Examples include: qualitative goal-oriented decision models (e.g. the NFR [180], i* [277] frameworks); methods that use abstract non-verifiable scores and pre-established, fixed, non-falsifiable model equations (e.g. the EVOLVE release planning method [216], the Cost-Benefit Architecture Method (CBAM) [149, 179] and other search-based methods for requirements selection and optimisation [201]); and the approaches that use quantitative problem specific decision models (e.g. the quantitative extensions to NFR/i* [3, 199], KAOS [126, 165] and the Bayesian decision analysis in software engineering [166]). The main limitations of these approaches has been the difficulty to elaborate domain-specific decision models and/or the lack of integrated tool support for automated decision analysis under uncertainty.

In this thesis, we have presented a new modelling language and automated decision analysis method, implemented in a tool called RADAR, to mitigate the limitations of existing approaches. The modelling language facilitates the elaboration and analysis of domain-specific requirements engineering and software architecture decision problems under uncertainty. It enables decision-makers to model design time decisions at different level of complexities: (i) decisions characterised by single option selection similar to mutually exclusive option selection (XOR-nodes) of feature diagrams used in software product lines. (ii) decisions characterised by three different constraints (excludes, requires and coupling) relationships between options of decisions and multiple options selection similar to non-mutually exclusive options selections (OR-nodes) of feature diagrams. The proposed automated decision analysis technique involves analysing uncertainty (epistemic) through Monte-Carlo simulations, shortlisting Pareto-optimal solutions through multi-objective optimisation, and computing expected value of information that can be used to decide whether to seek more information or perform a more detailed analysis before making a decision.

The thesis also presented an evaluation of the applicability, usefulness and performance of RADAR. Our evaluation results show that RADAR's modelling language and analysis technique is applicable on a range of real-world requirements and architecture decision problems (see Table 7.1 for the requirements and architecture decision problems with design space size between 6 and 2^{50}), and that in few seconds, RADAR can analyse decision problems characterised by large design space using highly performant optimisation method through the use of evolutionary search-based algorithms instead of exhaustive search. RADAR is useful in providing feedback to decision-makers about which decisions give the best trade-offs between conflicting stakeholder goals and which aspects of a decision model need further analysis or additional information before making a decision. RADAR also improves decision analysis in Software Product Line as it allows domain-specific equations which existing SPL tools lack.

9.1 Future Work

We highlight potential future research directions following the research presented in this thesis as follows:

9.1.1 Language and Analysis Extensions

The empirical evaluations presented in chapters 7 and 8 show that RADAR's modelling language is expressive enough to model real-world requirements and architecture decisions under uncertainty, and the automated decision analysis technique scales well in analysing these problems. An area of possible improvement is to augment the tool with mathematical functions such as MAX, MIN, AVERAGE and SQRT to further increase the tool's applicability. Extending the language with the capability of modelling state-based operations of software systems is a feasible future work. To achieve this, we propose using techniques from goal-oriented requirements engineering that use linear temporal logic operators to define the possible states of a system both in the past and future bounded by time [65, 162].

With respect to RADAR's decision analysis, an area of improvement is to augment RADAR's current constraint handling approach. Future work would be to explore other approaches, such as defining evolution functions (i.e., repair operators) to avoid generating invalid solutions [285], and combining constraint solving approaches with search-based optimisation techniques [128].

9.1.2 Iterative Decision Analysis Approach using Information Value Analysis

The thesis has shown how RADAR's analysis technique evaluates the expected value of perfect information (EVTPI) in order to determine the financial value of reducing uncertainty in a decision model. Computing EVTPI helps to identify parts of a decision model that needs additional data collection or further analysis through requirements elicitation, prototyping and modelling. If the estimated financial value of seeking more information significantly exceeds the effort required, then a new modelling and analysis cycle is triggered. In this thesis, we have not demonstrated such iterative method on a real-world problem. Future work is required to develop and evaluate an iterative decision analysis method that uses information value analysis to support informed requirements and architecture decisions.

9.1.3 Runtime Adaptation of Executing System

RADAR has been developed to support requirements and architecture decision at design-time. Such decisions, however, often have to be revised at run-time as more options become available or as uncertainty about some of the decision model parameters is reduced through observation of the running system. RADAR may provide the basis for such run-time adaptation, for example by embedding RADAR models in the running system through requirements reflection [35, 231]. Techniques for reassessing stakeholders' preferences at run-time may also be used to update RADAR models parameters that encode such preferences [34, 124].

9.1.4 Handing Other Forms of Uncertainty

Of the different forms of uncertainty (epistemic, alaetory and linguistic), this thesis have mainly considered epistemic uncertainty, i.e., lack of total knowledge about how a proposed system will operate or incomplete knowledge about the actual consequences of alternative decision choices on stakeholders' goals [166]. We have solved this form of uncertainty only at design time by computing the expected gain in some objective values (e.g. net benefit) of interest given total or partial information about some estimated model parameters. In the future, we propose to extend our solutions beyond design time to include system runtime where uncertainty may either reduce significantly or otherwise. In addition, we intend to tackle other forms of uncertainty, i.e., alaotory uncertainty which arise from random physical phenomenon within the context of a system or its executing environment.

9.1.5 Automated Model Validation and Calibration

In Chapter 7, we have exemplified RADAR on a set of real world examples. While some of the model equations developed for these examples are based on observed data, the correctness of RADAR's automated analysis presented relies on having a valid decision models. Throughout the thesis, we have only checked that these decision models developed for each example system are plausible and are sufficient to represent real-world system properties that stakeholders would be interested in based on experience. RADAR's approach, however, exposes a gap in requirements and architecture decision making research: there is currently lack of automated techniques for validating requirements and architecture decision models against observed data, and for automatically calibrating and inferring decision models from observed data. Therefore, to answer the question of validity of RADAR models, future work is required to implement techniques for validating and calibrating RADAR models against run time data of software systems. Techniques from data mining [114] and Bayesian inference could be used and adapted [274, 275].

9.1.6 Tool Evaluation in Organisation Context

An important future work would be to further evaluate the tool's language and decision analysis approach on large industrial case studies within an organisational setting. This will help in answering the questions about the simplicity, conciseness and clarity of the language constructs; whether the language constructs are sufficient to express the problems it addresses; whether the elaboration of a domain-specific requirements and architecture decision models are not excessively human resource-expensive; whether it is reasonably easy for modellers to learn the language; and finally validate the benefits of the RADAR's approach against the state of the art.

Appendix A

Modelling and Analysing The London Ambulance System

Problem Statement

The London Ambulance System (LAS) deploys ambulances to an incident location as quickly as possible [90, 126]. It is also used to provide pre-arranged services such as transporting and finding hospital beds for patients. In 1992, the UK Government imposed performance standards for accident and emergency calls, which states that "an ambulance must arrive at the scene within 14 minutes of the reported incidents". This standard necessitated the need for the first automated version of the LAS [126, 162].

Suppose, for example, a designer wants to perform a cost-benefit analysis of deciding whether to automate all or parts of the LAS. With the current system, the goal of 14 minutes response time is seldom achieved [90]. However, automating all or parts of the system will reduce the time it takes the crew to intervene at an incident. But, the cost and benefit of this automation are highly uncertain.

The design decisions of the London Ambulance System include [41, 126]:

- The *incident form encoding method* with the option to use a paper-based method to encode details about the calls reporting an incident or use a computer-based encoding, or automated encoding using call location mechanism, or a mix of computer-based encoding with automated call location.

- The *ambulance location method* with the option to use a radio and paper-based method to locate an idle ambulance, or an automated vehicle location method.
- The *ambulance allocation method* for dispatching the nearest ambulances to an incident location. Alternative options includes the use of a paper-based allocation method, an interactive allocation method, or a fully automated allocation method.
- The *mobilization communication method* that enables the ambulance crew to communicate directly with the ambulance station where incidents are reported. Alternative options includes the use of a radio, or mobile data terminals (MDT) system.
- The *number of deployed ambulances* within the city. Assume there are currently 100 ambulances, but this could be increased up to 200 or more ambulances.

RADAR Model

Modelling the Optimisation Objectives

Two key concerns of London Ambulance System are to minimise the response time for an ambulance to intervene an incident and to minimise costs, which include the cost of encoding an incident, cost of mobilising ambulances to incident locations, cost of locating and allocating ambulances.

```
Objective  Max ExpectedNetBenefit = EV(Benefit);
Objective  Min Risk = Pr(ResponseTime > 14 * 60);
```

The first stakeholders' concern of minimising ambulance response time is equivalent to maximising the net benefits of automating the ambulance management system.

We model the second stakeholders' concern as minimising the risk defined here as the probability that the ambulance response time exceeds 14 minutes (14×60 in seconds) of the reported incident. The 14 minutes is a United Kingdom Government standard [126].

Modelling Benefit

```
Benefit = Revenue - Cost;

Revenue = normalCI(100000, 200000);
```

The Benefit is defined as the difference between the Revenue and Cost. The Revenue follows a normal distribution with 90% confidence interval between £100000 and £200000.

Modelling Response Time

The ambulance RepsonseTime is the sum of the time taken to mobilise the ambulance, the travel time of the ambulance and possibly the delay an ambulance faced during travel.

```
ResponseTime = AmbulanceMobilisationTime
               + MobilisedAmbulanceTravelTime
               + MobilisedAmbulanceDelay;
```

The AmbulanceMobilisationTime is the sum of the incident call taking time, ambulance allocation time and the mobilisation communication time:

```
AmbulanceMobilisationTime = IncidentCallTakingTime
                            + AmbulanceAllocationTime
                            + MobilisationCommunicationTime;
```

```
IncidentCallTakingTime = decision("Incident Form Encoding Method"){
    "Current Paper-Based":exponential(60);
    "Computer Based":exponential(40);
    "Automated Call Location":exponential(45);
    "Computer-Based and Automated Call Location":exponential(30);
}
```

The IncidentCallTakingTime depends on the "incident form encoding method" used. If the current paper-based method is used, then the call taking time follows an exponential distribution with a mean of 60 seconds; If the computer-based method is used, then the call taking time has an exponential distribution with mean time of 40 seconds; if the automated call location method is selected, then the call taking time has exponential distribution with a mean value of 45 seconds; and if the computer-based and automated call location method is used, then the call taking time follows an exponential distribution with a mean time of 30 seconds.

```
AmbulanceAllocationTime = decision("Ambulance Allocation Method"){
    "Current Paper-Based":exponential(60);
    "Interactive Allocation":exponential(20);
    "Fully Automated Allocation":exponential(5);
}
```

The AmbulanceAllocationTime depends on the "ambulance allocation method". If the current paper-based method is used, then the ambulance allocation time follows an exponential distribution with a mean of 60 seconds; If the interactive allocation method is used, then the allocation time has an exponential distribution with mean time of 20 seconds; if the allocation is fully automated, then the allocation time has exponential distribution with a mean value of 5 seconds.

```
MobilisationCommunicationTime = decision("Mobilisation Communication Method"){
    "Current Radio-Based":triangular(45, 60, 90);
    "MDT-System A":triangular(20, 30, 40);
    "MDT-System B":triangular(10, 15, 20);
}
```

The MobilisationCommunicationTime depends on the "mobilisation communication method". If the current radio-based method is used, then the mobilisation communication time follows a triangular distribution with a lower and upper bounds of 45 and 90 seconds respectively, and a most likely time of 60 seconds; If a mobile data terminal system of a specific model type A is used, then the allocation time has a triangular distribution with a lower and upper bounds of 20 and 40 seconds respectively, and a most likely time of 30 seconds; if a different mobile data terminal system of another model type B is used, then the mobilisation communication time has a triangular distribution with a lower and upper bounds of 10 and 20 seconds respectively, and a most likely time of 15 seconds.

```
MobilisedAmbulanceTravelTime = MobilisedAmbulanceDistance / MobilisedAmbulanceAver-
ageSpeed;
```

The MobilisationAmbulanceTravelTime is the ratio of the distance the mobilised ambulance has to travel to the average mobilised ambulance speed.

```
MobilisedAmbulanceDistance = MobilisedAmbulanceDistanceFromLocation
                             + MobilisedAmbulanceLocationErrorMargin;
```

The mobilised ambulance distance is the sum of the mobilised ambulance distance from the incident location and an error margin when locating the ambulance.

Since ambulances can be at different location, we assume the mobilised ambulance to be within some distance from the incident location. Hence we assume the mobilised ambulance distance from location follows a normal distribution with confidence interval between 10000km and 100000km.

```
MobilisedAmbulanceDistanceFromLocation = normalCI(10000,100000);
```

```
MobilisedAmbulanceLocationErrorMargin = decision("Ambulance Localisation Method"){
    "Current Radio and Paper-Based":normal(60, 120);
    "Automated Vehicle Localisation A":normal(45, 60);
    "Automated Vehicle Localisation B":normal(20, 60);
}
MobilisedAmbulanceAverageSpreed = triangular(30, 50, 70);
MobilisedAmbulanceDelay = normalCI(0, 120);
```

The mobilisation ambulance location error margin depends on the ambulance localisation method; if the current radio and paper-based method is used, then the error margin follows a normal distribution with a mean of 60km and standard deviation of 120km; if the automated vehicle localisation method of model type A is used, then error margin has a normal distribution with a mean and standard deviation of 45km and 60km respectively; if a different automated vehicle localisation of model type B is used, then the error margin has a normal distribution with a mean and standard deviation of 20km and 60km respectively.

Modelling Costs

The cost is the sum of the ambulance mobilisation cost and the cost of ambulances:

```
Cost = AmbulanceMobilisationCost + CostOfAmbulance;

CostOfAmbulance =

        (NbrAmbulances - CurrentNbrOfAmbulance) * UnitCost + AnnualMaintenanceCost;

CurrentNbrOfAmbulance = deterministic(100);

UnitCost = triangular(5000, 7000, 10000);
```

The number of ambulances depends on the whether additional ambulance are deployed. The number of ambulances currently deployed is 100.

```
NbrAmbulances = decision("Additional Ambulance"){
    "Current_100_Amb" : deterministic(100);
    "120_Amb" : deterministic(120);
    "140_Amb" : deterministic(140);
    "160_Amb" : deterministic(160);
    "180_Amb" : deterministic(180);
    "200_Amb" : deterministic(200);
}
AnnualMaintenanceCost = decision("Additional Ambulance"){
    "Current_100_Amb" :triangular (5000, 7000, 10000);
    "120_Amb" : triangular(1000, 1500, 2000);
    "140_Amb" : triangular(1500, 2000, 2500);
    "160_Amb" : triangular(2000, 2500, 3000);
    "180_Amb" : triangular(2500, 3000, 3500);
    "200_Amb" : triangular(3000, 3500, 4000);
}
```

The ambulance mobilisation cost is the sum of the cost of incident form encoding and the cost of locating an idle ambulance:

```
AmbulanceMobilisationCost = CostOfIncidentFormEncoding + CostOfLocatingAmbulance;
CostOfIncidentFormEncoding = decision ("Incident Form Encoding Method"){
    "Current Paper-Based":triangular(150,200,250);
    "Computer Based":triangular(100, 150, 200);
    "Automated Call Location":triangular(50,100,120);
    "Computer-Based and Automated Call Location":triangular(10, 50, 100);
}
CostOfLocatingAmbulance = decision("Ambulance Localisation Method"){
    "Current Radio and Paper-Based": triangular(400, 500, 600);
    "Automated Vehicle Location A":triangular(300, 400, 500 );
    "Automated Vehicle Location B": triangular(200, 300, 400);
}
```

RADAR generates the AND/OR refinement graph and decision dependency graphs from the London ambulance model equations to aid visualisation of model structure. Fig. A.1a shows the partial AND/OR refinement graphs for the London ambulance model starting from the model variable AmbulanceMobilisationTime. Fig. A.1b shows the decision graph for the model.

Analysis Results

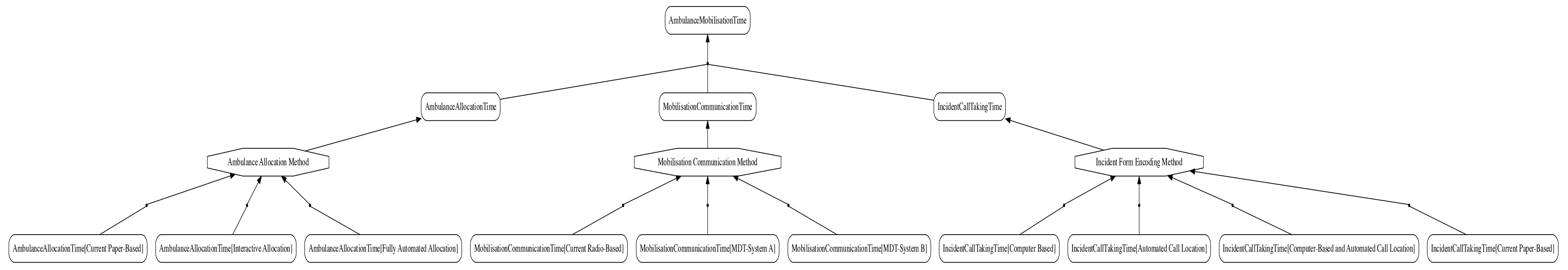
The RADAR analysis of the LAS model is presented in Fig. A.5 which shows the results of the optimisation analysis on the model. The results show that all shortlisted solutions include the paper-based option, radio-based option and the option to maintain the number of ambulances at 100. This implies that, in our model, the paper-based option outperforms the interactive and fully automated options of the ambulance allocation method. Similarly, the radio-based option outperforms the MDT-system of the ambulance mobilisation method. But once these two options are selected, the shortlist includes all possible combinations of incident encoding methods and ambulance allocation methods; each combination representing a tradeoff between maximising ExpectedNetBenefit and minimising Risk.

To visualise such tradeoffs, RADAR generates the graph in Fig. A.2 plotting the objective values for the shortlisted solutions (shown squares at the top of the graph) and all other non shortlisted ones (shown as circles).

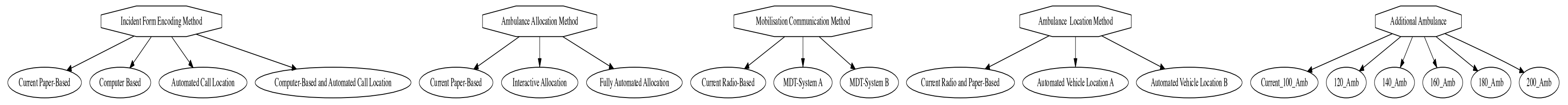
For information value analysis, RADAR estimates the expected value of perfect information. The estimated value of EVPI (for all model parameters) is approximately equal to zero. This means that none of the model parameters is worth further data collection or analysis.

Comparison To Previous Analysis

Previous modelling and analysis of the London Ambulance System (LAS) involves using the quantitative extension of the KAOS framework [65, 266] to elaborate stakeholder goals of achieving fast ambulance intervention within 14 minutes of every urgent call reporting an incident.


AmbulanceMobilisationTime
AmbulanceAllocationTime
MobilisationCommunicationTime
IncidentCallTakingTime
Ambulance Allocation Method
Mobilisation Communication Method
Incident Form Encoding Method
AmbulanceAllocationTime[Current Paper-Based]
AmbulanceAllocationTime[Interactive Allocation]
AmbulanceAllocationTime[Fully Automated Allocation]
MobilisationCommunicationTime[Current Radio-Based]
MobilisationCommunicationTime[MDT-System A]
MobilisationCommunicationTime[MDT-System B]
IncidentCallTakingTime[Computer Based]
IncidentCallTakingTime[Automated Call Location]
IncidentCallTakingTime[Computer-Based and Automated Call Location]
IncidentCallTakingTime[Current Paper-Based]

(A) Partial AND/OR refinement graph for the model variable AmbulanceMobilisationTime of the London Ambulance decision model.


Incident Form Encoding Method
Current Paper-Based
Computer Based
Automated Call Location
Computer-Based and Automated Call Location
Ambulance Allocation Method
Current Paper-Based
Interactive Allocation
Fully Automated Allocation
Mobilisation Communication Method
Current Radio-Based
MDT-System A
MDT-System B
Ambulance Location Method
Current Radio and Paper-Based
Automated Vehicle Location A
Automated Vehicle Location B
Additional Ambulance
Current_100_Amb
120_Amb
140_Amb
160_Amb
180_Amb
200_Amb

(B) Decision dependency graph for the London Ambulance decision model.

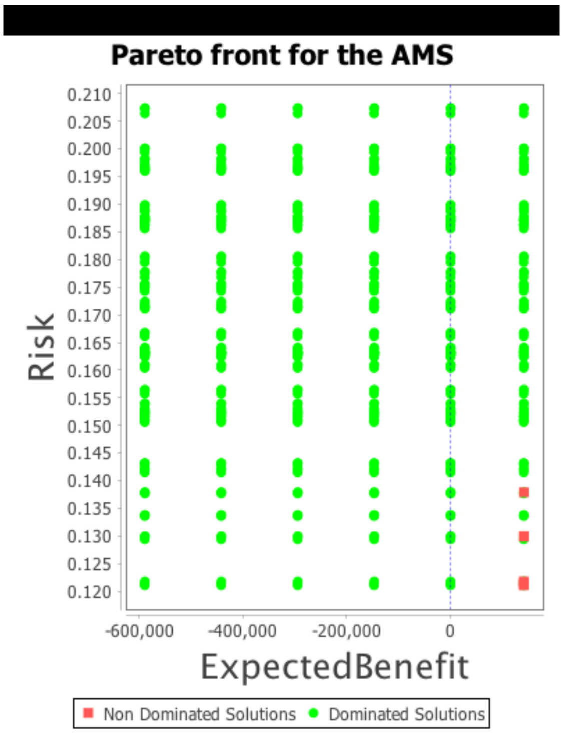


Figure A.2: Pareto front of the London ambulance model analysis.

Letier et al. [165] developed a quantitative technique that specifies partial degrees of goal satisfaction, and quantifies the impact of alternatives on the extent of goal satisfaction in terms of refinement equations, which are defined over random variables involved in the system‘s functional goals. They computed their objective functions for higher-level goals using estimated probability distribution functions from the leaf or low level quality variables.

Heaven et al. [126] extended the quantitative goal refinement model presented by Letier et al. [165] and developed a simulation and optimisation framework that evaluates the impact of alternative system designs on high level goals and uses multi-objective genetic algorithm (NSGA2) to shortlist the best alternative designs. They found the Pareto optimal design options among the alternatives options, that optimises the achievement of 8 and 14 minutes response time of the London Ambulance System at a low cost. Heaven et al. [126] developed a partial goal model for the LAS system with the decision points of alternative design as shown in Figure A.3. Their top level goal Achieve[Ambulance Intervention] is defined as shown below:

Goal Achieve[Ambulance Intervention]

Definition

For every urgent call reporting an incident, there should be an ambulance at the incident scene within 14 minutes after receiving the first call.

Formal Definition (∀ i:Incident, c: UrgentCall)

Reporting(c,i) $\Rightarrow \Diamond_{\leq 14minutes}$(∃a: Ambulance) Intervention(a,i).

Objective Functions

14MinResponseRate = MAX [P(ResponseTime ≤ 14 mins)]

Cost = MIN [AmbulanceCost]

Quality Variable

ResponseTime: Incident -> Time

def: the duration in seconds between the start of the first call reporting the incident and the arrival of the first ambulance at the incident scene.

In the above definition, the objective function of the top goal Achieve[Ambulance Intervention] is defined with respect to its quality variable (incident response time) which is recursively related to the quality variables of the sub-goals through refinement equations [126, 165]. For example, in Figure A.4, the quality variable ResponseTime of goal Achieve[Ambulance Intervention] is related to the quality variables MobilisationTime, MobilisationDistance, and AmbulanceDelay of goals Achieve[Ambulance Mobilisation] and Achieve[Mobilised Ambulance Intervention] by the equation:

$$ResponseTime = MobilisationTime + MobilisationDistance + AmbulanceDelay \tag{A.1}$$

As a comparison to our approach, the LAS goal-oriented decision model (LASGM) presented by Heaven et al. is similar to the RADAR model developed for LAS. However, RADAR was not able to model the Euclidean distance between an ambulance and the incident location using an Euclidean function between two points. In our model, we have modelled such distances using probability distributions.

Unlike the RADAR analysis technique that has tool support, the simulation and optimisation framework proposed by Heaven et al. lacks tool support for decision analysis and

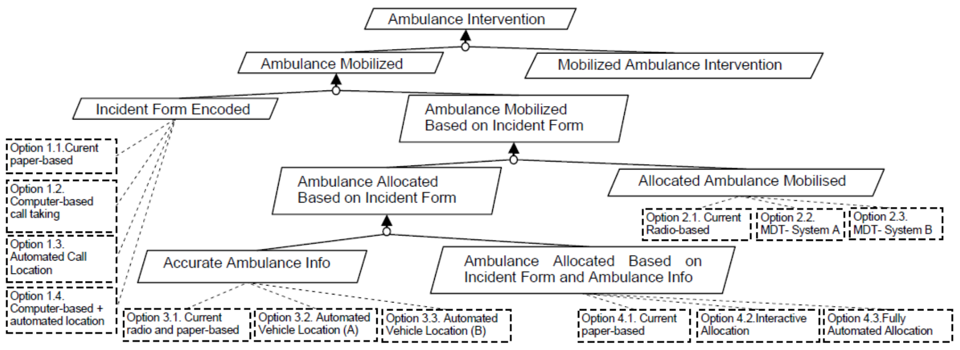

Ambulance Intervention
Ambulance Mobilized
Mobilized Ambulance Intervention
Incident Form Encoded
Ambulance Mobilized Based on Incident Form
Option 1.1.Curent paper-based
Option 1.2. Computer-based call taking
Option 1.3. Automated Call Location
Option 1.4. Computer-based + automated location
Ambulance Allocated Based on Incident Form
Allocated Ambulance Mobilised
Option 2.1. Current Radio-based
Option 2.2. MDT- System A
Option 2.3. MDT- System B
Accurate Ambulance Info
Ambulance Allocated Based on Incident Form and Ambulance Info
Option 3.1. Current radio and paper-based
Option 3.2. Automated Vehicle Location (A)
Option 3.3. Automated Vehicle Location (B)
Option 4.1. Current paper-based
Option 4.2.Interactive Allocation
Option 4.3.Fully Automated Allocation

Figure A.3: Partial goal model for the LAS system showing decision points for alternative system designs.

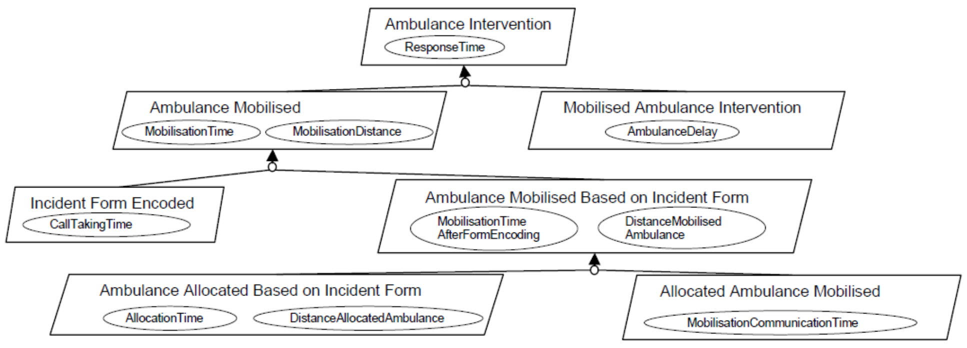

Ambulance Intervention
ResponseTime
Ambulance Mobilised
MobilisationTime
MobilisationDistance
Mobilised Ambulance Intervention
AmbulanceDelay
Incident Form Encoded
CallTakingTime
Ambulance Mobilised Based on Incident Form
MobilisationTime AfterFormEncoding
DistanceMobilised Ambulance
Ambulance Allocated Based on Incident Form
AllocationTime
DistanceAllocatedAmbulance
Allocated Ambulance Mobilised
MobilisationCommunicationTime

Figure A.4: Quality variables for the LAS goal model.

requires manual encoding of the simulation models in a general programming language, such as R and MATLAB. In their optimisation analysis of the LAS model, they used multi-objective evolutionary algorithms (e.g. NSGAII) to shortlist Pareto optimal solutions while our analysis technique used the exact multi-objective optimisation technique to guarantee finding the true optimal solutions. In addition, our approach analysed uncertainty in decision models using information value analysis which is not present in their approach.

Objective:	**Max** ExpectedNetBenefit
Objective:	**Min** Risk
Design Space:	648
Runtime(s):	0

Ambulance Allocation	Mobilisation Communication	Incident Form Encoding	Ambulance Location	Nbr. Ambulances	ExpectedNetBenefit($)	Risk(%)
Current: Paper-Based	Current: Radio-Based	Auto-Call Location	Automated Vehicle B	Current: 100	142612	0.13
Current: Paper-Based	Current: Radio-Based	Computer and Auto-Call Location	Automated Vehicle B	Current: 100	142649	0.13
Current: Paper-Based	Current: Radio-Based	Current: Paper-Based	Automated Vehicle B	Current: 100	142302	0.12
Current: Paper-Based	Current: Radio-Based	Current: Paper-Based	Automated Vehicle A	Current: 100	142402	0.12
Current: Paper-Based	Current: Radio-Based	Current: Paper-Based	Radio and Paper-Based	Current: 100	142501	0.12

FIGURE A.5: Optimisation analysis results for the AMS model. The number of design decisions is 5 and the design space is 648 (4×2×3×2 ×5).

Appendix B

Modelling and Analysing the NASA ECS Satellite Processing System

Problem Statement

The NASA's Earth Observing System Data Information (EOSDIS) Core System (ECS) is a large scale distributed data information system used in managing and distributing huge volumes of climate related data in different forms around the world, 24 hours each day. This system collects and manages more than 1000 gigabytes of data from several satellites using various sensors [150, 179].

As reported by Kazman et al. [151], the ECS went through a maintenance phase and a planning process of boosting its capabilities. However, the ECS project manager had limited annual budget to maintain and enhance the functionality of the system. In a bid to achieve this goal, prior analysis, using the ATAM methodology, was performed to identify a set of architectural strategies, which represents the decisions in the model (as shown in Table B.1), to be made in enhancing the system. The manager is faced with selecting among the set of decisions that give maximum utility in the project and also minimise the project cost.

Architectural Strategies	**Alternative Options**
Order Reassignment	Current: not possible to reassign order Allow Order Reassignment
Forced Order Completion	Current: not possible to force order completion Allow Forced Order Completion
Order Persistence Strategy	Current: store when processed Store as soon as received
Order Segmentation	Current: orders are segmented Orders are segmented
Hung Order Recovery	Current: no order retry Allow Order Retry
Failed Order Notification	Current: no notification User notified of failed order
Order Tracking	Current: order level Granule-level order tracking
Available User Information	Current: no link to user info Link to user information
Order Chunking	Current: no order chunking Order Chunking
Order Bundling	No Order Bundling Order Bundling

Table B.1: Overview of NASA ECS Architectural Strategies (decisions) and the corresponding options.

RADAR Model

Modelling the Optimisation Objectives

The primary decision objectives of the ECS model include two conflicting objectives: maximise the expected utility and minimise the cost associated to each alternative architectural strategies:

```
Objective Max ExpectedUtility = EV(Utility);

Objective Min Cost;
```

Modelling the Utility

Utility = HungRequestsRatioWeight × HungRequestsRatioUtility +
    LostRequestsRatioWeight × LostRequestsRatioUtility +
    FailedOrderRatioWeight × FailedOrderRatioUtility +
    HungOrdersRatioWeight × HungOrdersRatioUtility +
    LostOrdersRatioWeight × LostOrdersRatioUtility +
    HelpNeededByUsersWeight × HelpNeededByUsersUtility +
    FailureInfoGivenToUsersWeight × FailureInfoGivenToUsersUtility +
    LimitOnOrdersWeight × LimitOnOrdersUtility +
    NotificationsFrequencyWeight × NotificationsFrequencyUtility +
    PerformanceWeight × PerformanceUtility

HungRequestsRatioWeight = **deterministic**(10);

LostRequestsRatioWeight = **deterministic**(15);

FailedOrderRatioWeight = **deterministic**(15);

HungOrdersRatioWeight = **deterministic**(10);

LostOrdersRatioWeight = **deterministic**(15);

HelpNeededByUsersWeight = **deterministic**(10);

FailureInfoGivenToUsersWeight = **deterministic**(5);

LimitOnOrdersWeight = **deterministic**(5);

NotificationsFrequencyWeight = **deterministic**(10);

PerformanceWeight = **deterministic**(5);

Hung Request Ratio Utility

HungRequestRatioUtility = (HungRequestRatio - HungRequestRatioWorst)
    /(HungRequestRatioBest - HungRequestRatioWorst);

HungRequestRatioBest = **deterministic**(0);

HungRequestRatioWorst = **deterministic**(10%);

Lost Requests Ratio Utility

```
LostRequestsRatioUtility = (LostRequestsRatio - LostRequestsRatioWorst)
                           /(LostRequestsRatioBest - LostRequestsRatioWorst);
LostRequestsRatioBest = deterministic(0);
LostRequestsRatioWorst = deterministic(5%);
```

Failed Order Ratio Utility

```
FailedOrderRatioUtility = (FailedOrderRatio - FailedOrderRatioWorst)
                           /(FailedOrderRatioBest - FailedOrderRatioWorst)
FailedOrderRatioBest = deterministic(0);
FailedOrderRatioWorst = deterministic(10%);
```

Hung Orders Ratio Utility

```
HungOrdersRatioUtility = (HungOrdersRatio - HungOrdersRatioWorst)
                           /(HungOrdersRatioBest - HungOrdersRatioWorst);
HungOrdersRatioBest = deterministic(0);
HungOrdersRatioWorst = deterministic(10%);
```

Lost Orders Ratio Utility

```
LostOrdersRatioUtility = (LostOrdersRatio - LostOrdersRatioWorst)
                           /(LostOrdersRatioBest - LostOrdersRatioWorst);
LostOrdersRatioBest = deterministic(0);
LostOrdersRatioWorst = deterministic(10%);
```

Help Needed By Users Utility

```
HelpNeededByUsersUtility = (HelpNeededByUsers - HelpNeededByUsersWorst)
                    /(HelpNeededByUsersBest - HelpNeededByUsersWorst);
HelpNeededByUsersBest = deterministic(0);
HelpNeededByUsersWorst = deterministic(50%);
```

Failure Info Given To Users Utility

```
FailureInfoGivenToUsersUtility = (FailureInfoGivenToUsers - FailureInfoGivenToUser-
sWorst)/(FailureInfoGivenToUsersBest - FailureInfoGivenToUsersWorst);
FailureInfoGivenToUsersBest = deterministic(100%);
FailureInfoGivenToUsersWorst = deterministic(10%);
```

Limit On Orders Utility

```
LimitOnOrdersUtility = (LimitOnOrders - LimitOnOrdersWorst)
                    /(LimitOnOrdersBest - LimitOnOrdersWorst);
LimitOnOrdersBest = deterministic(0%);
LimitOnOrdersWorst = deterministic(50%);
```

Notifications Frequency Utility

```
NotificationsFrequencyUtility = (NotificationsFrequency - NotificationsFrequencyWorst)
/(NotificationsFrequencyBest - NotificationsFrequencyWorst);
NotificationsFrequencyBest = deterministic(1);
NotificationsFrequencyWorst = deterministic(1/1000);
```

Performance Utility

```
PerformanceUtility = (Performance - PerformanceWorst)
                     /(PerformanceBest - PerformanceWorst);
PerformanceBest = **deterministic**(90%);
PerformanceWorst = **deterministic**(50%);
```

Modelling Quality Attributes

Hung Request Ratio

```
HungRequestRatio = HungRequestRatioCurrent
                   × (1 - ReassignedHungRequestRatio)
                   × (1 - ForcedHungRequestRatio)
HungRequestRatioCurrent = deterministic(5%);
ReassignedHungRequestRatio = decision(”Order Reassignment”){
      ”Current: not possible to reassign order” : deterministic(0);
      ”Allow Order Reassignment”: deterministic(60%);
}
ForcedHungRequestRatio = decision(”Forced Order Completion”){
      ”Current: not possible to force order completion” : deterministic(0);
      ”Allow Forced Order Completion”:deterministic(40%);
}
```

Lost Request Ratio

```
LostRequestRatio = triangular(0, 0.5, 1);
```

Failed Order Ratio

```
FailedOrderRatio = decision("Order Persistence Strategy"){
        "Current: store when processed" : deterministic(5%);
        "Store as soon as received" : deterministic(2%);
}
```

Hung Orders Ratio

```
HungOrdersRatio = HungOrdersRatioCurrent ×
                  (1 - SkippedHungOrderRatio) ×
                  (1 - RetriedHungOrderRatio)

HungOrdersRatioCurrent = deterministic(10%);
SkippedHungOrderRatio = decision("Order Segmentation"){
        "Current: no order segmentation" : deterministic(0);
        "Orders are segmented" : deterministic(60%);
}
RetriedHungOrderRatio = decision("Hung Order Recovery"){
        "Current: no order retry" : deterministic(0);
        "Allow Order Retry" : deterministic(40%);
}
```

Lost Orders Ratio

```
LostOrdersRatio = decision("Order Persistence Strategy"){
        "Current: store when processed" : deterministic(1%);
        "Store as soon as received" : deterministic(0%);
}
```

Help Needed by Users

```
HelpNeededByUsers = LostOrdersRatio × HelpNeededPerLostOrder × HelpNeededForTrack-
ingGranularity
HelpNeededPerLostOrder = **decision**(”Failed Order Notification”){
    ”Current: no notification”:**deterministic**(25%);
    ”User notified of failed order”:**deterministic**(20%);
}
HelpNeededForTrackingGranularity = **decision**(Order Tracking){
    ”Current: order level” : **deterministic**(1);
    ”Granule-level order tracking” : **deterministic**(40%);
}
```

Failure Information

```
FailureInfoGivenToUsers = 1- RatioUsersNotGettingInfo
RatioUsersNotGettingInfo = RatioUsersNotGettingInfoCurrent ×
                           (1 - NotificationOrderEffect) ×
                           (1 - OrderTrackingGranularityEffect) ×
                           (1 - UserInformationEffect)

RatioUsersNotGettingInfoCurrent = **deterministic**(50%);
NotificationOrderEffect = **decision**(”Failed Order Notification”){
    ”Current: no notification”: **deterministic**(0);
    ”User notified of failed order”: **deterministic**(80%);
}
OrderTrackingGranularityEffect = **decision**(Order Tracking){
    ”Current: order level” : **deterministic**(0);
    ”Granule-level order tracking” : **deterministic**(90%);
}
UserInformationEffect = **decision**(”Available User Information”){
    ”Current: no link to user info” : **deterministic**(0);
    ”Link to user information” : **deterministic**(20%);
}
```

Limit on Order

```
LimitOnOrders = decision(”Order Chunking”){
        ”Current: no oder chunking”: deterministic(30%);
        ”Order Chunking” : deterministic(15%);
}
```

Notification Frequency

```
NotificationFrequency = decision(”Order Bundling”){
        ”No Order Bundling” : deterministic(1);
        ”Order Bundling” : deterministic(0.01);
}
```

Performance

```
Performance = decision(”Order Bundling”){
        ”No Order Bundling” : deterministic(60%);
        ”Order Bundling” : deterministic(55%);
}
```

Modelling Cost

```
Cost = CostOrderReassignment +
                        CostForcedOrderCompletion +
                        CostOrderPersistenceStrategy +
                        CostOrderSegmentation +
                        CostHungOrderRecovery +
                        CostFailedOrderNotification +
                        CostOrderTracking +
                        CostAvailableUserInformation +
CostOrderReassignment = decision(”Order Reassignment”){
        ”Current: not possible to reassign order” : deterministic(0);
        ”Allow Order Reassignment”: triangular(360, 400, 440);
}
CostForcedOrderCompletion = decision(”Forced Order Completion”){
        ”Current: not possible to force order completion” : deterministic(0);
        ”Allow Forced Order Completion”: triangular(180, 200, 220);
}
CostOrderPersistenceStrategy = decision(”Order Persistence Strategy”){
        ”Current: store when processed” : deterministic(0);
        ”Store as soon as received” : triangular(1200, 1200, 7150);
}
CostOrderSegmentation = decision(”Order Segmentation”){
        ”Current: no order segmentation” : deterministic(0);
        ”Orders are segmented” : triangular(180, 200, 220);
}
CostHungOrderRecovery = decision(”Hung Order Recovery”){
        ”Current: no order retry” : deterministic(0);
        ”Allow Order Retry” : triangular(180, 200, 220);
}
CostFailedOrderNotification = decision(”Failed Order Notification”){
        ”Current: no notification”: deterministic(0);
        ”User notified of failed order”: triangular(270, 300, 330);
}
CostOrderTracking = decision(Order Tracking){
        ”Current: order level” : deterministic(0);
        ”Granule-level order tracking” : triangular(900, 1000, 1650);
}
CostAvailableUserInformation = decision(”Available User Information”){
        ”Current: no link to user info” : deterministic(0);
        ”Link to user information” : triangular(90, 100, 440);
}
CostOrderChunking = decision(”Order Chunking”){
        ”Current: no oder chunking” : deterministic(0);
        ”Order Chunking” : triangular(360, 400, 440);
}
CostOrderBundling = decision(”Order Bundling”){
        ”No Order Bundling” : deterministic(0);
        ”Order Bundling” :triangular(360, 400, 440);
}
```

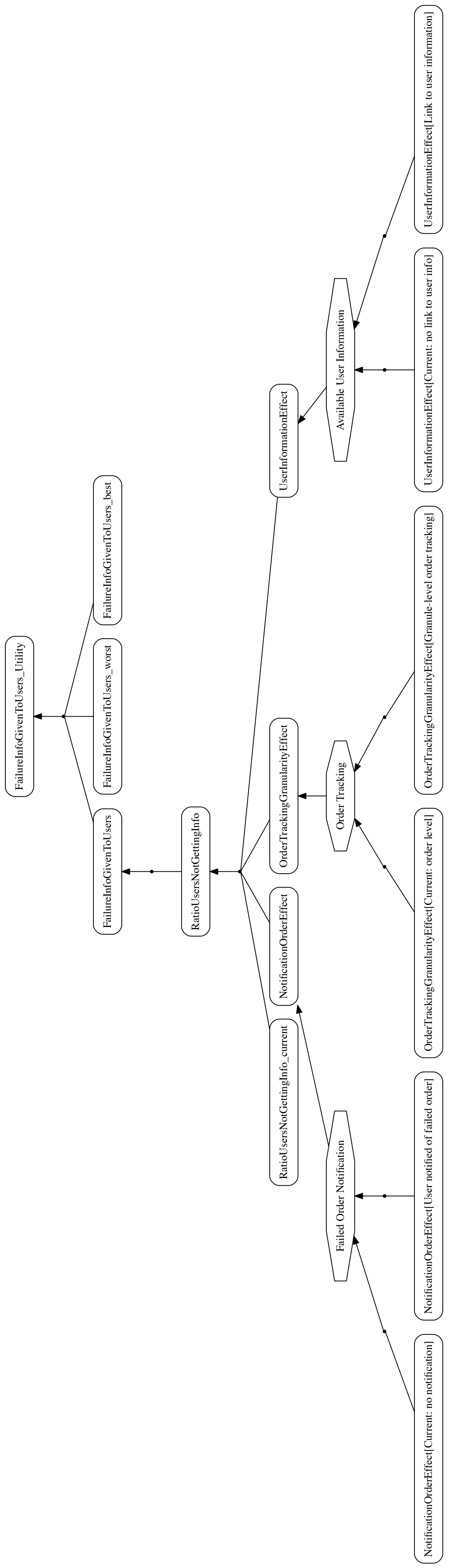
To help visualise the model structure, RADAR generates the AND/OR refinement graph and decision dependency graphs from the NASA ECS model equations. Fig. B.1a shows the partial AND/OR refinement graphs for the building security model starting from the model variable FailureInfoGivenToUsersUtility. Fig. B.1b shows a partial decision graph for the model.

Analysis Results

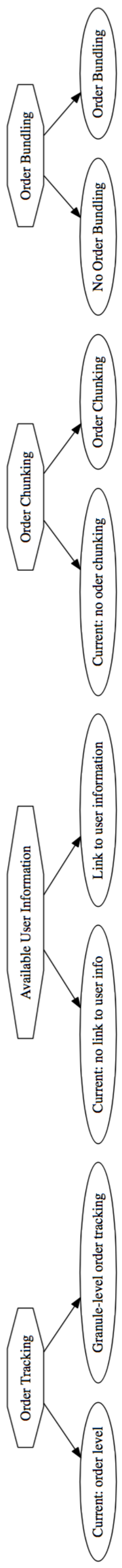
The RADAR analysis of the NASA ECS Decision Model (ECSM) is presented in Fig. B.3, which shows the results of the optimisation and information value analysis on the model.

The first part of Fig. B.3 is the optimisation analysis results, which shows that all shortlisted solutions include the options “Order Chunking” , “Orders are segmented” and “No order Bundling”. This means that, in our model, these three options, respectively, outperform the options “Current: No Order Chunking”, “Current: No Order Segmentation” and “No Order Bundling” on both objectives. But once these three options are selected, the shortlist includes different combinations of *Order Reassignment*, *Forced Order Completion*, *Order Persistence Strategy*, *Hung Order recovery*, *Failed Order Notification*, *Order Tracking*, *Available User Information*; each combination representing a different tradeoffs between maximising ExpectedUtility and minimising the Cost. To visualise such tradeoffs, RADAR generates the graph in Fig. B.2, plotting the objective values for the shortlised solutions (i.e. shown squares at the top of the graph) and all other non shortlisted ones (shown as circles).

The information value analysis results is presented in Table B.2, which shows that the EVTPI for this problem is 0.04 and EVPPI for all the model parameters is 0. This means that in this model, there is no parameter worth investigating further before deciding between the shortlisted solutions to be selected for implementation. Reducing uncertainty about any of the parameters would bring no value to the decision.


FailureInfoGivenToUsers_Utility
FailureInfoGivenToUsers
FailureInfoGivenToUsers_worst
FailureInfoGivenToUsers_best
RatioUsersNotGettingInfo
RatioUsersNotGettingInfo_current
NotificationOrderEffect
OrderTrackingGranularityEffect
UserInformationEffect
Failed Order Notification
Order Tracking
Available User Information
NotificationOrderEffect[Current: no notification]
NotificationOrderEffect[User notified of failed order]
OrderTrackingGranularityEffect[Current: order level]
OrderTrackingGranularityEffect[Granule-level order tracking]
UserInformationEffect[Current: no link to user info]
UserInformationEffect[Link to user information]

(A) Partial AND/OR refinement graph for the model variable FailureInfoGivenToUsersUtility of NASA ECS decision model.



(B) Decision dependency graph for the NASA ECS decision model.

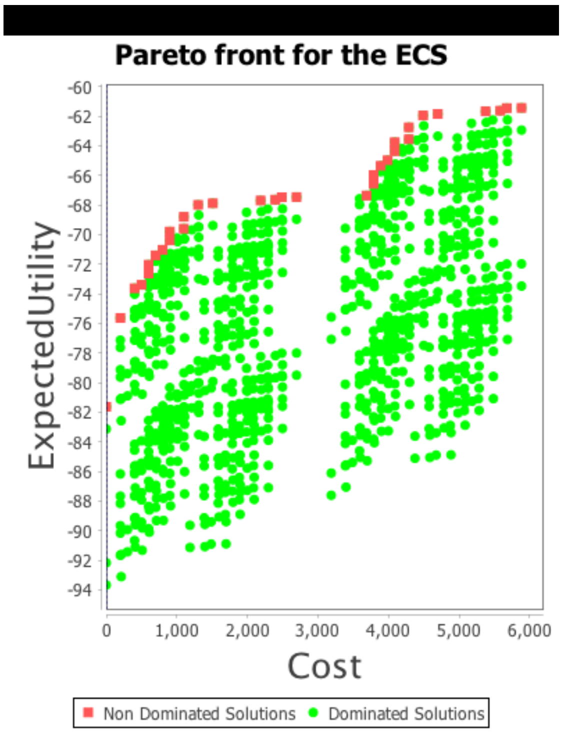


Figure B.2: Pareto front of the NASA's ECS model analysis.

Information Value Analysis

Objective:	**Max** ENB
EVTPI:	0

Parameter	EVPPI
LostRequestsRatio	0
CostOrderReassignment[Allow Order Reassignment]	0
CostForcedOrderCompletion[Allow Forced Order Completion]	0
CostOrderPersistenceStrategy[Store as soon as received]	0
CostOrderSegmentation[Orders are segmented]	0
CostHungOrderRecovery[Allow Order Retry]	0
CostFailedOrderNotification[User notified of failed order]	0
CostOrderTracking[Granule-level order tracking]	0
CostAvailableUserInformation[Link to user information]	0
CostOrderChunking[Order Chunking]	0
CostOrderBundling[Order Bundling]	0

Table B.2: Information Value Analysis for the NASA ECS System.

Comparison To Previous Analysis

Previous analysis involves the application of the Cost Benefit Analysis Method (CBAM) [149, 179] to the ECS project. This analysis has been presented in the 2002 Software

Engineering Institute (SEI) report. The analysis estimates individual impact of individual architectural strategies on the utility derived from the system, but does not consider the combined impacts of architectural strategies on goals. In our approach, we model the interactions between architectural strategies. i.e. the combined impact of architectural strategies. In addition, our analysis technique allows the selection of more than one architectural strategy and analyses uncertainty by computing the expected value of total and partial perfect information.

Optimisation Analysis	
Objective:	**Max**ExpectedUtility
Objective:	**Min**Cost
SolutionSpace	1024
Minimal SolutionSet	1024
Shortlisted	33
Nbr. Variables	75
Nbr. Parameters	11
Nbr. Decisions	10
Runtime(s)	2

ID	Order Reassignment	Forced Order Completion	Order Persistence Strategy	Order Segmentation	Hung Order Recovery	Failed Order Notification	Order Tracking	Available User Information	Order Chunking	Order Bundling	ExpectedUtility	Cost
1	Current: not possible to reassign order	Current: not possible to force order completion	Store as soon as received	Orders are segmented	Current: no order retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-67.35429264	3687.114892
2	Allow Order Reassignment	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Allow Order Retry	Current: no notification	Granule-level order tracking	Current: no link to user info	Order Chunking	No Order Bundling	-67.69651487	2186.319942
3	Allow Order Reassignment	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Allow Order Retry	User notified of failed order	Current: order level	Link to user information	Order Chunking	No Order Bundling	-67.88318153	1510.418841
4	Allow Order Reassignment	Allow Forced Order Completion	Store as soon as received	Orders are segmented	Allow Order Retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-61.95429264	4487.265338
5	Current: not possible to reassign order	Allow Forced Order Completion	Store as soon as received	Orders are segmented	Allow Order Retry	Current: no notification	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-65.97651487	3787.318577
6	Allow Order Reassignment	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Allow Order Retry	User notified of failed order	Granule-level order tracking	Current: no link to user info	Order Chunking	No Order Bundling	-67.47029264	2486.276777
7	Current: not possible to reassign order	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Allow Order Retry	Current: no notification	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-72.02651487	600.1639298
8	Allow Order Reassignment	Allow Forced Order Completion	Store as soon as received	Orders are segmented	Current: no order retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-63.55429264	4287.128492
9	Current: not possible to reassign order	Current: not possible to force order completion	Current: store when processed	Orders are segmented	Current: no order retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-73.39429264	499.9602443
10	Allow Order Reassignment	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Allow Order Retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-67.99429264	1300.110691
11	Current: not possible to reassign order	Allow Forced Order Completion	Store as soon as received	Orders are segmented	Allow Order Retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-63.75429264	4087.275413
12	Allow Order Reassignment	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Allow Order Retry	Current: no notification	Granule-level order tracking	Link to user information	Order Chunking	No Order Bundling	-67.64095931	2396.628092
13	Allow Order Reassignment	Allow Forced Order Completion	Store as soon as received	Orders are segmented	Allow Order Retry	User notified of failed order	Granule-level order tracking	Current: no link to user info	Order Chunking	No Order Bundling	-61.45429264	5673.431425
14	Current: not possible to reassign order	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Current: no order retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-71.39429264	699.9839196
15	Allow Order Reassignment	Allow Forced Order Completion	Store as soon as received	Orders are segmented	Allow Order Retry	User notified of failed order	Current: order level	Link to user information	Order Chunking	No Order Bundling	-61.84318153	4697.573489
16	Allow Order Reassignment	Current: not possible to force order completion	Current: store when processed	Orders are segmented	Current: no order retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-70.39429264	899.9501696
17	Current: not possible to reassign order	Current: not possible to force order completion	Current: store when processed	Orders are segmented	Current: no order retry	Current: no notification	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-75.62651487	200.0034088
18	Current: not possible to reassign order	Current: not possible to force order completion	Current: store when processed	Current: no order segmentation	Current: no order retry	Current: no notification	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-81.62651487	0
19	Allow Order Reassignment	Current: not possible to force order completion	Current: store when processed	Orders are segmented	Current: no order retry	Current: no notification	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-72.62651487	599.9933341
20	Allow Order Reassignment	Current: not possible to force order completion	Store as soon as received	Orders are segmented	Allow Order Retry	Current: no notification	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-64.97651487	3987.284827
21	Allow Order Reassignment	Current: not possible to force order completion	Current: store when processed	Orders are segmented	Allow Order Retry	Current: no notification	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-71.02651487	800.1301799
22	Allow Order Reassignment	Allow Forced Order Completion	Store as soon as received	Orders are segmented	Allow Order Retry	User notified of failed order	Granule-level order tracking	Link to user information	Order Chunking	No Order Bundling	-61.44318153	5883.739575
23	Allow Order Reassignment	Current: not possible to force order completion	Store as soon as received	Orders are segmented	Current: no order retry	Current: no notification	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-66.57651487	3787.147982
24	Allow Order Reassignment	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Allow Order Retry	User notified of failed order	Granule-level order tracking	Link to user information	Order Chunking	No Order Bundling	-67.45918153	2696.584928
25	Allow Order Reassignment	Current: not possible to force order completion	Current: store when processed	Orders are segmented	Allow Order Retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-68.79429264	1100.087015
26	Current: not possible to reassign order	Allow Forced Order Completion	Store as soon as received	Orders are segmented	Current: no order retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-65.35429264	3887.138567
27	Allow Order Reassignment	Current: not possible to force order completion	Store as soon as received	Orders are segmented	Current: no order retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-64.35429264	4087.104817
28	Allow Order Reassignment	Allow Forced Order Completion	Store as soon as received	Orders are segmented	Allow Order Retry	Current: no notification	Granule-level order tracking	Link to user information	Order Chunking	No Order Bundling	-61.62095931	5583.78274
29	Allow Order Reassignment	Current: not possible to force order completion	Store as soon as received	Orders are segmented	Allow Order Retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-62.75429264	4287.241663
30	Allow Order Reassignment	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Current: no order retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-69.59429264	1099.973845
31	Allow Order Reassignment	Allow Forced Order Completion	Store as soon as received	Orders are segmented	Allow Order Retry	Current: no notification	Granule-level order tracking	Current: no link to user info	Order Chunking	No Order Bundling	-61.67651487	5373.474589
32	Current: not possible to reassign order	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Allow Order Retry	User notified of failed order	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-69.79429264	900.1207654
33	Current: not possible to reassign order	Allow Forced Order Completion	Current: store when processed	Orders are segmented	Current: no order retry	Current: no notification	Current: order level	Current: no link to user info	Order Chunking	No Order Bundling	-73.62651487	400.0270841

FIGURE B.3: Analysis results for the cost-benefit analysis model

Appendix C

Modelling and Analysing Feature Selection of Drupal (PHP Framework)

The optimisation objectives are given below:

```
Objective  Min ExpectedComplexity = EV(Complexity);

Objective  Max ExpectedTestAssertions = EV(TestAssertions);

Objective  Max ExpectedNumberOfInstallations = EV(TotalNumberOfInstallations);

Objective  Min ExpectedNumberOfDeVelopers = EV (TotalNumberOfDevelopers);
```

Modelling Complexity

```
Complexity = FeatureNodeComplexity
                 + FeatureSystemComplexity
                 + FeatureFilterComplexity
                 + FeatureFieldComplexity
                 + FeatureUserComplexity
                 + OtherFeaturesComplexity;

FeatureNodeComplexity = ComplexityForFeatureNode+ FeatureNodeSubFeatureComplexity;
ComplexityForFeatureNode = deterministic(0.27);
FeatureNodeSubFeatureComplexity = decision-subset(+) ("Node") {
    "Blog" : deterministic(0.16);
    "Forum" : deterministic(0.24);
}

FeatureSystemComplexity = deterministic(0.31);
FeatureFilterComplexity = deterministic(0.17);
```

```
FeatureFieldComplexity = ComplexityForFeatureField + FeatureTextComplexity + Feature-
FieldSQLStorageComplexity + FeatureOptionsComplexity;

ComplexityForFeatureField = deterministic(0.41);
FeatureTextComplexity= deterministic(0.29);
FeatureFieldSQLStorageComplexity = deterministic(0.3);

FeatureOptionsComplexity = decision("Field Options"){
    "Options" : deterministic(0.17);
    "Without Options" : deterministic(0);
}
```

FeatureUserComplexity = **deterministic**(0.26);

OtherFeaturesComplexity = **decision-subset**(+)(”Drupal Non-Mandatory Features”){

“Path” : **deterministic**(0.14);

“Image” : **deterministic**(0.29);

“Field UI” : **deterministic**(0.28);

“File” : **deterministic**(0.67);

“Comment” : **deterministic**(0.23);

“Views” : FeatureViewsComplexity + FeatureViewsUIComplexity ;

“Libraries API” : **deterministic**(0.55);

“IMCE” : **deterministic**(0.47);

“Ctools” : FeatureCToolsComplexity + CToolsSubFeaturesComplexity;

“Token”: **deterministic**(0.51);

“Taxonomy” : **deterministic**(0.23);

“Date” : FeatureDateComplexity + DateSubFeaturesComplexity;

“WebForm” : **deterministic**(0.51);

“Link” : **deterministic**(0.63);

“EntityAPI” : FeatureEntityAPIComplexity + EntityAPISubFeatureComplexity ;

“CKEditor” : **deterministic**(0.59);

“Captcha” : FeatureCaptchaComplexity + CaptchaSubFeatureComplexity;

“Features” : **deterministic**(0.56);

“Panels” : FeaturePanelsComplexity + PanelSubFeatureComplexity;

“Pathauto” : **deterministic**(0.23);

“JQuery” : **deterministic**(0.26);

“GoogleAnalytics” : **deterministic**(0.29);

“Rules” : FeatureRuleComplexity + RuleSubFeatureComplexity;

“BackUpMigration” : **deterministic**(0.37);

}

```
FeatureViewsComplexity = deterministic(0.41);
FeatureViewsUIComplexity = decision(”Views UI SubFeature”){
    “Views UI”: deterministic(0.37);
    “Without UI”: deterministic(0);
}

FeatureCToolsComplexity = deterministic(0.52);
CToolsSubFeaturesComplexity = decision-subset(+)(”Ctool SubFeature”){
    “Ctools access ruleset” : deterministic(0.19);
    “Ctools custom content” : deterministic(0.3);
    “Views content” : deterministic(0.46);
}

FeatureDateComplexity = deterministic(0.44);
DateSubFeaturesComplexity = decision-subset(+)(”Date SubFeature”){
    “Date API” : deterministic(0.6);
    “Date views” : deterministic(0.44);
    “Date popups” : deterministic(0.36);
}

FeatureEntityAPIComplexity = deterministic(1);
EntityAPISubFeatureComplexity = decision(”Entity API SubFeatures”){
    “Entity Tokens” : deterministic(1.09);
    “Without Entity Tokens” : deterministic(0);
}
```

```
FeatureCaptchaComplexity = deterministic(0.19);
CaptchaSubFeatureComplexity = decision(”Captcha SubFeature”){
    “Image Captcha” : deterministic(0.28);
    “Without Image Captcha” : deterministic(0);
}

FeaturePanelsComplexity =deterministic(0.35);
PanelSubFeatureComplexity = decision-subset(+)(”Panels SubFeatures”){
    “Panel Nodes” : deterministic(0.35);
    “Panels IPE” : deterministic(0.23);
}

FeatureRuleComplexity = deterministic(0.49);
RuleSubFeatureComplexity = decision-subset(+)(”Rules SubFeatures”){
    “Rules Scheduler” : deterministic(0.15);
    “Rules UI” : deterministic(0.39);
}
```

Modelling Test Assertions

```
TestAssertions = FeatureNodeTestAssertions + FeatureSystemTestAsser-
tions
              + FeatureFilterTestAssertions
              + FeatureFieldTestAssertions
              + FeatureUserTestAssertions
              + OtherFeaturesTestAssertions;
```

```
FeatureNodeTestAssertions = TestAssertionsForFeatureNode+ FeatureNodeSubFeatureTestAssertions;
TestAssertionsForFeatureNode = deterministic(1391);
FeatureNodeSubFeatureTestAssertions = decision-subset(+)(”Node”) {
    “Blog” : deterministic(244);
    “Forum” : deterministic(677);
}

FeatureSystemTestAssertions = deterministic(2138);
FeatureFilterTestAssertions = deterministic(958);

FeatureFieldTestAssertions = TestAssertionsForFeatureField + FeatureTextTestAssertions +
FeatureFieldSQLStorageTestAssertions + FeatureOptionsTestAssertions;

TestAssertionsForFeatureField = deterministic(870);
FeatureTextTestAssertions = deterministic(444);
FeatureFieldSQLStorageTestAssertions = deterministic(94);

FeatureOptionsTestAssertions = decision(”Field Options”){
    “Options” : deterministic(227);
    “Without Options” : deterministic(0);
}
```

```
FeatureUserTestAssertions = deterministic(1335);
OtherFeaturesTestAssertions = decision-subset(+)(”Drupal Non-Mandatory Features”){
    “Path” : deterministic(330);
    “Image” : deterministic(667);
    “Field UI” : deterministic(287);
    “File” : deterministic(2293);
    “Comment” : deterministic(3287);
    “Views” : FeatureViewsTestAssertions + FeatureViewsUITestAssertions ;
    “Libraries API” : deterministic(135);
    “IMCE” : deterministic(0);
    “Ctools” : FeatureCToolsTestAssertions + CToolsSubFeaturesTestAssertions;
    “Token”: deterministic(347);
    “Taxonomy” : deterministic(677);
    “Date” : FeatureDateTestAssertions + DateSubFeaturesTestAssertions;
    “WebForm” : deterministic(456);
    “Link” : deterministic(1275);
    “EntityAPI” : FeatureEntityAPITestAssertions + EntityAPISubFeatureTestAssertions ;
    “CKEditor” : deterministic(0);
    “Captcha” : FeatureCaptchaTestAssertions + CaptchaSubFeatureTestAssertions;
    “Features” : deterministic(16);
    “Panels” : FeaturePanelsTestAssertions + PanelSubFeatureTestAssertions;
    “Pathauto” : deterministic(316);
    “JQuery” : deterministic(0);
    “GoogleAnalytics” : deterministic(200);
    “Rules” : FeatureRuleTestAssertions + RuleSubFeatureTestAssertions;
    “BackUpMigration” : deterministic(0);
}
FeatureViewsTestAssertions = deterministic(1089);
FeatureViewsUITestAssertions = decision(”Views UI SubFeature”){
    “Views UI”: deterministic(538);
    “Without UI”: deterministic(0);
}
```

```
FeatureCToolsTestAssertions = **deterministic**(1);
CToolsSubFeaturesTestAssertions = **decision-subset**(+)(”Ctool SubFeature”){
    “Ctools access ruleset” : **deterministic**(0);
    “Ctools custom content” : **deterministic**(0);
    “Views content” : **deterministic**(0);
}

FeatureDateTestAssertions = **deterministic**(1);
DateSubFeaturesTestAssertions = **decision-subset**(+)(”Date SubFeature”){
    “Date API” : **deterministic**(106);
    “Date views” : **deterministic**(0);
    “Date popups” : **deterministic**(0);
}

FeatureEntityAPITestAssertions = **deterministic**(1);
EntityAPISubFeatureTestAssertions = **decision**(”Entity API SubFeatures”){
    “Entity Tokens” : **deterministic**(6);
    “Without Entity Tokens” : **deterministic**(0);
}
```

```
FeatureCaptchaTestAssertions = deterministic(851);
CaptchaSubFeatureTestAssertions = decision(”Captcha SubFeature”){
    “Image Captcha” : deterministic(1);
    “Without Image Captcha” : deterministic(0);
}

FeaturePanelsTestAssertions =deterministic(0);
PanelSubFeatureTestAssertions = decision-subset(+)(”Panels SubFeatures”){
    “Panel Nodes” : deterministic(0);
    “Panels IPE” : deterministic(0);
}

FeatureRuleTestAssertions = deterministic(285);
RuleSubFeatureTestAssertions = decision-subset(+)(”Rules SubFeatures”){
    “Rules Scheduler” : deterministic(7);
    “Rules UI” : deterministic(0);
}
```

Modelling Total Number Of Installations

```
TotalNumberOfInstallations = FeatureNodeTotalNumberOfInstallations
                           + FeatureSystemTotalNumberOfInstallations
                           + FeatureFilterNumberOfInstallations
                           + FeatureFieldNumberOfInstallations
                           + FeatureUserNumberOfInstallations
                           + OtherFeaturesNumberOfInstallations;
```

```
FeatureNodeTotalNumberOfInstallations = TotalNumberOfInstallationsForFeatureNode+
FeatureNodeSubFeatureTotalNumberOfInstallations;

TotalNumberOfInstallationsForFeatureNode = deterministic(5259525);
FeatureNodeSubFeatureTotalNumberOfInstallations = decision-subset(+)(”Node”) {
    “Blog” : deterministic(5259525);
    “Forum” : deterministic(5259525);
}
```

```
FeatureSystemTotalNumberOfInstallations= deterministic(5259525);
FeatureFilterTotalNumberOfInstallations= deterministic(5259525);

FeatureFieldTotalNumberOfInstallations= TotalNumberOfInstallationsForFeatureField + Fea-
tureTextTotalNumberOfInstallations+ FeatureFieldSQLStorageTotalNumberOfInstallations+
FeatureOptionsTestAssertions;

TotalNumberOfInstallationsForFeatureField= deterministic(5259525);
FeatureTextTotalNumberOfInstallations= deterministic(5259525);
FeatureFieldSQLStorageTotalNumberOfInstallations= deterministic(5259525);

FeatureOptionsTotalNumberOfInstallations= decision(”Field Options”){
    “Options” : deterministic(5259525);
    “Without Options” : deterministic(0);
}
FeatureUserTotalNumberOfInstallations= deterministic(5259525);
```

```
OtherFeaturesNumberOfInstallations = decision-subset(+)(”Drupal Non-Mandatory Features”){
    “Path” : deterministic(5259525);
    “Image” : deterministic(5259525);
    “Field UI” : deterministic(5259525);
    “File” : deterministic(5259525);
    “Comment” : deterministic(5259525);
    “Views” : FeatureViewsTotalNumberOfInstallations+ FeatureViewsUITotalNumberOfInstallations;
    “Libraries API” : deterministic(516333);
    “IMCE” : deterministic(392705);
    “Ctools” : FeatureCToolsTotalNumberOfInstallations+ CToolsSubFeaturesTestAssertions;
    “Token”: deterministic(715563);
    “Taxonomy” : deterministic(677);
    “Date” : FeatureDateTotalNumberOfInstallations+ DateSubFeaturesTestAssertions;
    “WebForm” : deterministic(402163);
    “Link” : deterministic(286892);
    “EntityAPI” : FeatureEntityAPITotalNumberOfInstallations+ EntityAPISubFeatureTotalNumberOfInstallations;
    “CKEditor” : deterministic(280919);
    “Captcha” : FeatureCaptchaTotalNumberOfInstallations+ CaptchaSubFeatureTestAssertions;
    “Features” : deterministic(209653);
    “Panels” : FeaturePanelsTotalNumberOfInstallations+ PanelSubFeatureTestAssertions;
    “Pathauto” : deterministic(622478);
    “JQuery” : deterministic(286556);
    “GoogleAnalytics” : deterministic(348278);
    “Rules” : FeatureRuleTotalNumberOfInstallations+ RuleSubFeatureTestAssertions;
    “BackUpMigration” : deterministic(281797);
}
FeatureViewsTotalNumberOfInstallations= deterministic(1);
FeatureViewsUITotalNumberOfInstallations= decision(”Views UI SubFeature”){
    “Views UI”: deterministic(802467);
    “Without UI”: deterministic(0);
}
```

```
FeatureCToolsTotalNumberOfInstallations= **deterministic**(1);
CToolsSubFeaturesTotalNumberOfInstallations= decision-subset(+)(”Ctool SubFeature”){
    “Ctools access ruleset” : deterministic(747248);
    “Ctools custom content” : deterministic(747248);
    “Views content” : deterministic(747248);
}

FeatureDateTotalNumberOfInstallations= deterministic(412324);
DateSubFeaturesTotalNumberOfInstallations= decision-subset(+)(”Date SubFeature”){
    “Date API” : deterministic(412324);
    “Date views” : deterministic(412324);
    “Date popups” : deterministic(412324);
}
```

```
FeatureEntityAPITotalNumberOfInstallations= deterministic(407569);
EntityAPISubFeatureTotalNumberOfInstallations= decision(”Entity API SubFeatures”){
    “Entity Tokens” : deterministic(407569);
    “Without Entity Tokens” : deterministic(0);
}

FeatureCaptchaTotalNumberOfInstallations= deterministic(226295);
CaptchaSubFeatureTotalNumberOfInstallations= decision(”Captcha SubFeature”){
    “Image Captcha” : deterministic(226295);
    “Without Image Captcha” : deterministic(0);
}

FeaturePanelsTotalNumberOfInstallations=deterministic(206805);
PanelSubFeatureTotalNumberOfInstallations= decision-subset(+)(”Panels SubFeatures”){
    “Panel Nodes” : deterministic(206805);
    “Panels IPE” : deterministic(206805);
}
FeatureRuleTotalNumberOfInstallations= deterministic(238388);
RuleSubFeatureTotalNumberOfInstallations= decision-subset(+)(”Rules SubFeatures”){
    “Rules Scheduler” : deterministic(238388);
    “Rules UI” : deterministic(238388);
}
```

Modelling Total Number Of Developers

```
TotalNumberOfDevelopers = FeatureNodeTotalNumberOfDevelopers
                        + FeatureSystemTotalNumberOfDevelopers
                        + FeatureFilterNumberOfDevelopers
                        + FeatureFieldNumberOfDevelopers
                        + FeatureUserNumberOfDevelopers
                        + OtherFeaturesNumberOfDevelopers;
```

```
FeatureNodeTotalNumberOfDevelopers = TotalNumberOfDevelopersForFeatureNode+ FeatureNodeSubFeatureTotalNumberOfDevelopers;

TotalNumberOfDevelopersForFeatureNode = deterministic(94);
FeatureNodeSubFeatureTotalNumberOfDevelopers = decision-subset(+)(”Node”) {
    “Blog” : deterministic(94);
    “Forum” : deterministic(94);
}

FeatureSystemTotalNumberOfDevelopers= deterministic(94);
FeatureFilterTotalNumberOfDevelopers= deterministic(94);

FeatureFieldTotalNumberOfDevelopers=TotalNumberOfDevelopersForFeatureField + FeatureTextTotalNumberOfDevelopers+ FeatureFieldSQLStorageTotalNumberOfDevelopers+ FeatureOptionsTotalNumberOfDevelopers;

TotalNumberOfDevelopersForFeatureField= deterministic(94);
FeatureTextTotalNumberOfDevelopers= deterministic(94);
FeatureFieldSQLStorageTotalNumberOfDevelopers= deterministic(94);


FeatureOptionsTotalNumberOfDevelopers= decision(”Field Options”){
    “Options” : deterministic(94);
    “Without Options” : deterministic(0);
}
```

```
FeatureUserTotalNumberOfDevelopers= deterministic(94);
OtherFeaturesNumberOfDevelopers = decision-subset(+)(”Drupal Non-Mandatory Fea-
tures”){
    “Path” : deterministic(94);
    “Image” : deterministic(94);
    “Field UI” : deterministic(94);
    “File” : deterministic(94);
    “Comment” : deterministic(94);
    “Views” : FeatureViewsTotalNumberOfDevelopers+ FeatureViewsUITotalNumberOfDevel-
opers;
    “Libraries API” : deterministic(7);
    “IMCE” : deterministic(13);
    “Ctools” : FeatureCToolsTotalNumberOfDevelopers+ CToolsSubFeaturesTestAssertions;
    “Token”: deterministic(31);
    “Taxonomy” : deterministic(94);
    “Date” : FeatureDateTotalNumberOfDevelopers+ DateSubFeaturesTestAssertions;
    “WebForm” : deterministic(46);
    “Link” : deterministic(31);
    “EntityAPI” : FeatureEntityAPITotalNumberOfDevelopers+ EntityAPISubFeatureTotal-
NumberOfDevelopers;
    “CKEditor” : deterministic(29);
    “Captcha” : FeatureCaptchaTotalNumberOfDevelopers+ CaptchaSubFeatureTestAsser-
tions;
    “Features” : deterministic(36);
    “Panels” : FeaturePanelsTotalNumberOfDevelopers+ PanelSubFeatureTestAssertions;
    “Pathauto” : deterministic(33);
    “JQuery” : deterministic(17);
    “GoogleAnalytics” : deterministic(21);
    “Rules” : FeatureRuleTotalNumberOfDevelopers+ RuleSubFeatureTestAssertions;
    “BackUpMigration” : deterministic(7);
}
FeatureViewsTotalNumberOfDevelopers= deterministic(178);
FeatureViewsUITotalNumberOfDevelopers= decision(”Views UI SubFeature”){
    “Views UI”: deterministic(178);
    “Without UI”: deterministic(0);
}
```

```
FeatureCToolsTotalNumberOfDevelopers= deterministic(75);
CToolsSubFeaturesTotalNumberOfDevelopers= decision-subset(+)(”Ctool SubFeature”){
    “Ctools access ruleset” : deterministic(75);
    “Ctools custom content” : deterministic(75);
    “Views content” : deterministic(75);
}

FeatureDateTotalNumberOfDevelopers= deterministic(42);
DateSubFeaturesTotalNumberOfDevelopers= decision-subset(+)(”Date SubFeature”){
    “Date API” : deterministic(42);
    “Date views” : deterministic(42);
    “Date popups” : deterministic(42);
}

FeatureEntityAPITotalNumberOfDevelopers= deterministic(45);
EntityAPISubFeatureTotalNumberOfDevelopers= decision(”Entity API SubFeatures”){
    “Entity Tokens” : deterministic(45);
    “Without Entity Tokens” : deterministic(0);
}
```

```
FeatureCaptchaTotalNumberOfDevelopers= deterministic(43);
CaptchaSubFeatureTotalNumberOfDevelopers= decision(”Captcha SubFeature”){
    “Image Captcha” : deterministic(43);
    “Without Image Captcha” : deterministic(0);
}

FeaturePanelsTotalNumberOfDevelopers=deterministic(43);
PanelSubFeatureTotalNumberOfDevelopers= decision-subset(+)(”Panels SubFeatures”){
    “Panel Nodes” : deterministic(43);
    “Panels IPE” : deterministic(43);
}

FeatureRuleTotalNumberOfDevelopers= deterministic(52);
RuleSubFeatureTotalNumberOfDevelopers= decision-subset(+)(”Rules SubFeatures”){
    “Rules Scheduler” : deterministic(52);
    “Rules UI” : deterministic(52);
}
```

Modelling Total Number of Changes

```
TotalNumberOfChanges = FeatureNodeTotalNumberOfChanges
                     + FeatureSystemTotalNumberOfChanges
                     + FeatureFilterNumberOfChanges
                     + FeatureFieldNumberOfChanges
                     + FeatureUserNumberOfChanges
                     + OtherFeaturesNumberOfChanges;
```

```
FeatureNodeTotalNumberOfChanges = TotalNumberOfChangesForFeatureNode+ FeatureN-
odeSubFeatureTotalNumberOfChanges;

TotalNumberOfChangesForFeatureNode = **deterministic**(9);
FeatureNodeSubFeatureTotalNumberOfChanges = **decision-subset**(+) (”Node”) {
    “Blog” : **deterministic**(0);
    “Forum” : **deterministic**(3);
}

FeatureSystemTotalNumberOfChanges= **deterministic**(19);
FeatureFilterTotalNumberOfChanges= **deterministic**(1);

FeatureFieldTotalNumberOfChanges= TotalNumberOfChangesForFeatureField + FeatureText-
TotalNumberOfChanges+ FeatureFieldSQLStorageTotalNumberOfChanges+ FeatureOption-
sTotalNumberOfChanges;

TotalNumberOfChangesForFeatureField= **deterministic**(6);
FeatureTextTotalNumberOfChanges= **deterministic**(0);
FeatureFieldSQLStorageTotalNumberOfChanges= **deterministic**(1);

FeatureOptionsTotalNumberOfChanges= **decision**(”Field Options”){
    “Options” : **deterministic**(0);
    “Without Options” : **deterministic**(0);
}
FeatureUserTotalNumberOfChanges= **deterministic**(7);
```

OtherFeaturesNumberOfChanges = **decision-subset**(+)(”Drupal Non-Mandatory Features”){

“Path” : **deterministic**(0);

“Image” : **deterministic**(9);

“Field UI” : **deterministic**(4);

“File” : **deterministic**(1);

“Comment” : **deterministic**(2);

“Views” : FeatureViewsTotalNumberOfChanges+ FeatureViewsUITotalNumberOfChanges;

“Libraries API” : **deterministic**(7);

“IMCE” : **deterministic**(9);

“Ctools” : FeatureCToolsTotalNumberOfChanges+ CToolsSubFeaturesTestAssertions;

“Token”: **deterministic**(10);

“Taxonomy” : **deterministic**(2);

“Date” : FeatureDateTotalNumberOfChanges+ DateSubFeaturesTestAssertions;

“WebForm” : **deterministic**(46);

“Link” : **deterministic**(11);

“EntityAPI” : FeatureEntityAPITotalNumberOfChanges+ EntityAPISubFeatureTotal-NumberOfChanges;

“CKEditor” : **deterministic**(40);

“Captcha” : FeatureCaptchaTotalNumberOfChanges+ CaptchaSubFeatureTestAssertions;

“Features” : **deterministic**(72);

“Panels” : FeaturePanelsTotalNumberOfChanges+ PanelSubFeatureTestAssertions;

“Pathauto” : **deterministic**(2);

“JQuery” : **deterministic**(1);

“GoogleAnalytics” : **deterministic**(14);

“Rules” : FeatureRuleTotalNumberOfChanges+ RuleSubFeatureTestAssertions;

“BackUpMigration” : **deterministic**(90);

}

FeatureViewsTotalNumberOfChanges= **deterministic**(27);

FeatureViewsUITotalNumberOfChanges= **decision**(”Views UI SubFeature”){

“Views UI”: **deterministic**(0);

“Without UI”: **deterministic**(0);

}

```
FeatureCToolsTotalNumberOfChanges= deterministic(32);
CToolsSubFeaturesTotalNumberOfChanges= decision-subset(+)(”Ctool SubFeature”){
    “Ctools access ruleset” : deterministic(0);
    “Ctools custom content” : deterministic(1);
    “Views content” : deterministic(5);
}

FeatureDateTotalNumberOfChanges= deterministic(9);
DateSubFeaturesTotalNumberOfChanges= decision-subset(+)(”Date SubFeature”){
    “Date API” : deterministic(11);
    “Date views” : deterministic(6);
    “Date popups” : deterministic(4);
}

FeatureEntityAPITotalNumberOfChanges= deterministic(14);
EntityAPISubFeatureTotalNumberOfChanges= decision(”Entity API SubFeatures”){
    “Entity Tokens” : deterministic(1);
    “Without Entity Tokens” : deterministic(0);
}

FeatureCaptchaTotalNumberOfChanges= deterministic(15);
CaptchaSubFeatureTotalNumberOfChanges= decision(”Captcha SubFeature”){
    “Image Captcha” : deterministic(0);
    “Without Image Captcha” : deterministic(0);
}
```

```
FeaturePanelsTotalNumberOfChanges=deterministic(34);
PanelSubFeatureTotalNumberOfChanges= decision-subset(+)(”Panels SubFeatures”){
    “Panel Nodes” : deterministic(2);
    “Panels IPE” : deterministic(20);
}


FeatureRuleTotalNumberOfChanges= deterministic(5);
RuleSubFeatureTotalNumberOfChanges= decision-subset(+)(”Rules SubFeatures”){
    “Rules Scheduler” : deterministic(4);
    “Rules UI” : deterministic(1);
}
```

Modelling constraints

Constraint "Node": "Forum" **requires**"Drupal Non-Mandatory Features" : "Taxonomy";

Constraint "Node": "Forum" **requires**"Field Options" : "Options";

Constraint "Node": "Forum" **requires**"Drupal Non-Mandatory Features" : "Comment";

Constraint "Drupal Non-Mandatory Features": "Image" **requires**"Drupal Non-Mandatory Features" : "File";

Constraint "Drupal Non-Mandatory Features": "Views" **requires**"Drupal Non-Mandatory Features" : "Ctools";

Constraint "Views UI SubFeature": "Views UI" **requires**"Drupal Non-Mandatory Features" : "Ctools";

Constraint "Ctools": "Views content" **requires**"Drupal Non-Mandatory Features" : "Views";

Constraint "Drupal Non-Mandatory Features": "Taxonomy" **requires**"Field Options" : "Options";

Constraint "Date SubFeature": "Date views" **requires**"Drupal Non-Mandatory Features" : "Ctools";

Constraint "Date SubFeature": "Date views" **requires**"Drupal Non-Mandatory Features" : "Views";

Constraint "Drupal Non-Mandatory Features": "Panels" **requires**"Drupal Non-Mandatory Features" : "Ctools";

Constraint "Panels": "Panels IPE" **requires**"Drupal Non-Mandatory Features" : "Ctools";

Constraint "Panels": "Panel Nodes" **requires**"Drupal Non-Mandatory Features" : "Ctools";

Constraint "Drupal Non-Mandatory Features": "Pathauto" **requires**"Drupal Non-Mandatory Features" : "Token";

Constraint "Drupal Non-Mandatory Features": "Pathauto" **requires**"Drupal Non-Mandatory Features" : "Path";

Constraint "Drupal Non-Mandatory Features": "Rules" **requires**"Drupal Non-Mandatory Features" : "EntityAPI";

Constraint "Drupal Non-Mandatory Features": "Rules" **requires**"Entity API SubFeatures" : "Entity Tokens";

Constraint "Rules SubFeatures": "Rules Scheduler" **requires**"Drupal Non-Mandatory Features" : "EntityAPI";

Constraint "Rules SubFeatures": "Rules Scheduler" **requires**"Entity API SubFeatures" : "Entity Tokens";

Constraint "Rules SubFeatures": "Rules UI" **requires**"Drupal Non-Mandatory Features" : "EntityAPI";

Constraint "Rules SubFeatures": "Rules UI" **requires**"Entity API SubFeatures" : "Entity Tokens";

Analysis Result

The RADAR analysis of Drupal (PHP framework) model is presented in Fig.C.1. The problems was analysed using NSGAII using the 1+λ optimisation approach [130] and algorithmic parameters similar to the settings used in [285]: population size of 100, crossover probability of 0.9, mutation probability of 0.1 and maximum number of fitness evaluation of 50000. The optimisation results shows RADAR shortlists 28 optimal alternatives each representing different trade-offs between minimising code complexity; maximising the number of test assertions; maximising the number of installations and minimising the number of developers.

Optimisation Analysis

Objective:	**Min** ExpectedComplexity
Objective:	**Max** ExpectedTestAssertions
Objective:	**Max** ExpectedNumberOfInstallations
Objective:	**Min** ExpectedNumberOfDevelopers
Optimisation Approach	1+λ
Algorithms Settings	Algorithm (NSGAII), Pop. Size (100), P_c (0.8), P_m (0.1), Fitness Evaluation (50000)
	100
Shortlisted	28
Nbr. Variables	135
Nbr. Decisions	10
Runtime(s)	43

ID	Node	Field	Drupal Non-Mandatory Features	Views UI	Ctool	Date	Entity API	Captcha	Panels	Rules	Expected	Expected	ExpectedNumber	ExpectedNumber
	Node	Options		SubFeature	SubFeature	SubFeature	SubFeatures	SubFeature	SubFeatures	SubFeatures	Complexity	TestAssertions	Of Installations	Of Developers
1	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	—	—	11	19751	40865633	1516
2	Blog	With Options	Field UI;File;Comment;Token;Date;WebForm;EntityAPI;Captcha;Rules	Without UI	Ctools access ruleset;Ctools custom content	—	Entity Tokens	Image Captcha	—	—	8	15516	34367022	869
3	—	With Options	Path;Field UI;File;Comment;IMCE;Token;WebForm;Pathauto;JQuery;GoogleAnalytics;BackUpMigration	Without UI	Ctools access ruleset;Ctools custom content	—	Entity Tokens	Image Captcha	—	—	6	14973	34606690	732
4	Forum	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	—	—	11	20184	40865633	1516
5	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Without UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	—	—	11	19213	40063166	1338
6	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Without Image Captcha	—	—	11	19750	40865632	1515
7	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Without Image Captcha	Panels IPE	—	11	19750	40865632	1515
8	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	Panel Nodes;Panels IPE	—	11	19751	40865633	1516
9	Forum	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Without UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	—	—	11	19646	40063166	1338
10	Forum	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Without UI	Ctools access ruleset;Ctools custom content	Date API	Entity Tokens	Without Image Captcha	Panels IPE	Rules Scheduler	11	19758	40063278	1450
11	—	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Without UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	—	—	10	18969	34803641	1244
12	—	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Without UI	Ctools access ruleset	—	Entity Tokens	Without Image Captcha	—	—	10	18968	34803640	1243
13	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Without UI	Ctools access ruleset	—	Entity Tokens	Without Image Captcha	—	—	10	19212	40063165	1337
14	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	—	Rules Scheduler	11	19758	40865640	1523
15	Forum	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Without Image Captcha	—	—	11	20183	40865632	1515
16	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	Panels IPE	Rules Scheduler	11	19758	40865640	1523
17	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Without UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	Panel Nodes;Panels IPE	—	11	19213	40063166	1338
18	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Without Image Captcha	Panel Nodes;Panels IPE	—	11	19750	40865632	1515
19	—	With Options	Path;Field UI;File;Comment;IMCE;Token;WebForm;Pathauto;JQuery;GoogleAnalytics;BackUpMigration	Without UI	Ctools access ruleset;Ctools custom content	—	Entity Tokens	Without Image Captcha	—	—	6	14973	34606690	732
20	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Without UI	Ctools access ruleset	Date API	Entity Tokens	Image Captcha	—	—	11	19319	40063272	1444
21	Forum	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	Panels IPE	—	11	20184	40865633	1516
22	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Without UI	Ctools access ruleset	—	Entity Tokens	Without Image Captcha	Panels IPE	—	10	19212	40063165	1337
23	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	Panels IPE	—	11	19751	40865633	1516
24	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Views UI	Ctools access ruleset	Date API	Entity Tokens	Image Captcha	—	—	12	19857	40865739	1622
25	Blog	With Options	Image;Field UI;File;Comment;Views;Libraries API;Ctools;Taxonomy;Date;WebForm;Link;EntityAPI;Captcha;GoogleAnalytics;Rules	Without UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	—	Rules Scheduler	11	19220	40063173	1345
26	—	With Options	Path;Field UI;File;Comment;IMCE;Token;WebForm;Pathauto;JQuery;GoogleAnalytics;BackUpMigration	Without UI	Ctools access ruleset	—	Entity Tokens	Image Captcha	—	—	6	14973	34606690	732
27	Blog	Without Options	Field UI;File;Comment;Token;Date;WebForm;EntityAPI;Captcha;Rules	Without UI	Ctools access ruleset;Ctools custom content	—	Entity Tokens	Image Captcha	—	—	8	15289	34367022	869
28	Blog	With Options	Field UI;File;Comment;Token;Date;WebForm;EntityAPI;Captcha;Rules	Without UI	Ctools access ruleset;Ctools custom content	—	Entity Tokens	Without Image Captcha	—	—	8	15515	34367021	868

Figure C.1: Analysis results for the Drupal

Appendix D

Modelling and Analysing An E-Commerce System

The optimisation objectives are given below:

```
Objective  Min ExpectedCost = EV(Cost);

Objective  Min ExpectedTotalDefect = EV(TotalDefects);

Objective  Max TotalFeaturesUsedBefore = EV(FeaturesUsedBefore);

Objective  Max TotalFeatureCount = EV(FeatureCount);
```

We model the Cost below:

```
Cost = CostOfFeatureWebServer + CostOfAchievingOtherServices;

CostOfFeatureWebServer = CostOfProtocolAndLogging +CostOfFeatureContent;


CostOfProtocolAndLogging = decision-subset(+)(”Protocol And Logging”){

    ”Logging”: CostOfFeatureLogging;

    ”Protocols”: CostOfFeatureProtocols;

}
```

```
CostOfFeatureLogging = decision(”Feature Logging”){
    ”DB”: CostOfUsingDB;
    ”File”: CostOfUsingFile;
}
CostOfUsingDB = deterministic(10.308317459185346);
CostOfUsingFile = deterministic(5.901282664841281);
CostOfFeatureProtocols = decision-subset(+) (”Feature Protocol”){
    ”NTTP”: deterministic(13.682644828346174);
    ”FTP”: deterministic(13.008853827785588);
    ”HTTPS” : deterministic(10.048601003623034);
}
CostOfFeatureContent = CostOfFeatureStatic + CostOfFeatureActive;
CostOfFeatureStatic = deterministic(11.101621731632662);
CostOfFeatureActive = decision-subset(+) (”Active Features”){
    ”Active”: deterministic(12.247116996854373);
    ”ASP”: deterministic(8.620782055044714);
    ”PHP”: deterministic(10.641905327748809);
    ”JSP”: deterministic(12.95291397374664);
    ”CGI”: deterministic(12.141681946598066);
}
CostOfAchievingOtherServices = decision-subset(+)(”Other Services”){
    ”Additional Services”: CostOfFeatureAdditionalServices;
    ”Security”: CostOfFeatureSecurity;
    ”Persistence”: CostOfFeaturePersistence;
    ”Performance”: CostOfFeaturePerformance;
}
CostOfFeatureAdditionalServices = decision-subset(+)(”Feature Additional Services”){
    ”Additional Services”: deterministic(7.5092053767634654);
    ”Site Statistics”: CostOfFeatureSiteStatistics;
    ”Site Search”: CostOfFeatureSiteSearch;
    ”Ad Server”: CostOfFeatureAdServer;
}
```

```
CostOfFeatureSiteStatistics = CostOfFeatureStatistics
                                  + CostOfFeatureBasicStat
                                  + CostOfFeatureAdvancedStat;
CostOfFeatureStatistics =deterministic(14.689398134577047);
CostOfFeatureBasicStat = deterministic(6.166499393904992);
CostOfFeatureAdvancedStat = decision(”Feature Advanced Stat”){
    ”Without Advanced Stat”: deterministic(0);
    ”With Advanced Stat”: deterministic(10.419039139443939);
}
CostOfFeatureSiteSearch = decision-subset(+)(”Feature Site Search”){
    ”Site Search”: CostOfSiteSearch;
    ”Images”: CostOfFeatureImages;
    ”Text”: CostOfFeatureText;
}
CostOfSiteSearch = deterministic(9.758947026333994);
CostOfFeatureImages = deterministic(8.148236814831348);
CostOfFeatureText = CostOfText + CostOfFeatureHTML + CostOfFeatureDynamic;
CostOfText = deterministic(13.465111749758071);
CostOfFeatureHTML = deterministic(10.95267855613079);

CostOfFeatureDynamic = decision(”Feature Dynamic”){
    ”Without Dynamic”: deterministic(0);
    ”With Dynamic”: deterministic(8.863099463028671);
}
CostOfFeatureAdServer = CostOfAdServer                    + CostOfFeatureReports
                   + CostOfFeatureBanners
                   + CostOfPopupsAndKeywordSupport;

CostOfAdServer = deterministic(7.7496457787452195);
CostOfFeatureReports = deterministic(6.4745651041969055);
CostOfFeatureBanners = CostOfBanners + CostOfFeatureImage + CostOfFeatureFlash;
CostOfBanners = deterministic(7.363245163135134);
CostOfFeatureImage = deterministic(13.428384485896377);
```

```
CostOfFeatureFlash = decision(”Flash”){
    ”Without Flash”: deterministic(0);
    ”With Flash”: deterministic(5.452286607902222);
}
CostOfPopupsAndKeywordSupport = decision-subset(+)(”Popups And Keyword Support”){
    ”Popups”: CostOfFeaturePopups;
    ”Keyword Support”: CostOfFeatureKeywordSupport;
}
CostOfFeatureKeywordSupport =deterministic(6.100473520805334);
CostOfFeaturePopups = deterministic(11.397415959703473);
CostOfFeaturePersistence = CostOfPersistence + CostOfPersistenceSubFeatures;
CostOfPersistence = deterministic(10.474662762049276);
CostOfPersistenceSubFeatures = decision(”Persistence Mechanism”){
    ”XML”: deterministic(14.054171683911981);
    ”Database”: deterministic(6.7377605081216725);
}
CostOfFeatureSecurity = CostOfSecurity + CostOfSecuritySubFeatures;
CostOfSecurity = deterministic(5.013527622729248);
CostOfSecuritySubFeatures = decision-subset(+)(”Security Mechanism”){
    ”Data Storage”: deterministic( 9.556632783363145);
    ”Data Transfer”: deterministic(5.222981863350846);
    ”User Authentication”: deterministic(12.234241931520607);
}
CostOfFeaturePerformance = CostOfPerformance + CostOfPerformanceSubFeatures;
CostOfPerformance = deterministic(5.013527622729248);
CostOfPerformanceSubFeatures = decision(”Feature Performance”){
    ”Milli Sec”: deterministic(11.690308919683972);
    ”Seconds”: deterministic(9.08978755801254);
    ”Minutes”: deterministic(8.283453001056765);
}
```

We model the total feature defects below:

```
TotalDefects = DefectCountOnFeatureWebServer + DefectCountOnAchievingOtherServices;
DefectCountOnFeatureWebServer = DefectCountOnProtocolAndLogging
                        +DefectCountOnFeatureContent;

DefectCountOnProtocolAndLogging = decision-subset(+)(”Protocol And Logging”){
    ”Logging”: DefectCountOnFeatureLogging;
    ”Protocols” : DefectCountOnFeatureProtocols;
}
DefectCountOnFeatureLogging = decision(”Feature Logging”){
    ”DB” : DefectCountOnUsingDB;
    ”File” : DefectCountOnUsingFile;
}
DefectCountOnUsingDB = deterministic(5);
DefectCountOnUsingFile = deterministic(0);
DefectCountOnFeatureProtocols = decision-subset(+) (”Feature Protocol”){
    ”NTTP” : deterministic(6);
    ”FTP” : deterministic(6);
    ”HTTPS” : deterministic(0);
}
DefectCountOnFeatureContent = DefectCountOnFeatureStatic + DefectCountOnFeatureAc-
tive;
DefectCountOnFeatureStatic = deterministic(3);
DefectCountOnFeatureActive = decision-subset(+) (”Active Features”){
    ”Active” : deterministic(0);
    ”ASP”: deterministic(0);
    ”PHP”: deterministic(3);
    ”JSP”: deterministic(6);
    ”CGI”: deterministic(5);
}
DefectCountOnAchievingOtherServices = decision-subset(+)(”Other Services”){
    ”Additional Services”: DefectCountOnFeatureAdditionalServices;
    ”Security” : DefectCountOnFeatureSecurity;
    ”Persistence” : DefectCountOnFeaturePersistence;
    ”Performance” : DefectCountOnFeaturePerformance;
}
```

```
DefectCountOnFeatureAdditionalServices = decision-subset(+)(”Feature Additional Ser-
vices”){
    ”Additional Services” : deterministic(6);
    ”Site Statistics” : DefectCountOnFeatureSiteStatistics;
    ”Site Search” : DefectCountOnFeatureSiteSearch;
    ”Ad Server” : DefectCountOnFeatureAdServer;
} DefectCountOnFeatureSiteStatistics = DefectCountOnFeatureStatistics
                             + DefectCountOnFeatureBasicStat
                             + DefectCountOnFeatureAdvancedStat;

DefectCountOnFeatureStatistics =deterministic(0);
DefectCountOnFeatureBasicStat = deterministic(8);
DefectCountOnFeatureAdvancedStat = decision(”Feature Advanced Stat”){
    ”Without Advanced Stat” : deterministic(0);
    ”With Advanced Stat” : deterministic(5);
}
DefectCountOnFeatureSiteSearch = decision-subset(+)(”Feature Site Search”){
    ”Site Search” : DefectCountOnSiteSearch;
    ”Images” : DefectCountOnFeatureImages;
    ”Text” : DefectCountOnFeatureText;
}
DefectCountOnSiteSearch = deterministic(6);
DefectCountOnFeatureImages = deterministic(5);
DefectCountOnFeatureText = DefectCountOnText
                     + DefectCountOnFeatureHTML
                     + DefectCountOnFeatureDynamic;
DefectCountOnText = deterministic(6);
DefectCountOnFeatureHTML = deterministic(4);
DefectCountOnFeatureDynamic = decision(”Feature Dynamic”){
    ”Without Dynamic” : deterministic(0);
    ”With Dynamic” : deterministic(4);
}
DefectCountOnFeatureAdServer = DefectCountOnAdServer
                 + DefectCountOnFeatureReports
                 + DefectCountOnFeatureBanners
                 + DefectCountOnPopupsAndKeywordSupport;
```

```
DefectCountOnAdServer = deterministic(6);
DefectCountOnFeatureReports = deterministic(4);
DefectCountOnFeatureBanners = DefectCountOnBanners
                    + DefectCountOnFeatureImage
                    + DefectCountOnFeatureFlash;
DefectCountOnBanners = deterministic(2);
DefectCountOnFeatureImage = deterministic(4);

DefectCountOnFeatureFlash = decision(”Flash”){
    ”Without Flash” : deterministic(0);
    ”With Flash” : deterministic(5);
}
DefectCountOnPopupsAndKeywordSupport = decision-subset(+)(”Popups And Keyword
Support”){
    ”Popups” : DefectCountOnFeaturePopups;
    ”Keyword Support” : DefectCountOnFeatureKeywordSupport;
}
DefectCountOnFeatureKeywordSupport =deterministic(5);
DefectCountOnFeaturePopups = deterministic(0);
DefectCountOnFeaturePersistence = DefectCountOnPersistence + DefectCountOnPersistence-
SubFeatures;
DefectCountOnPersistence = deterministic(0);

DefectCountOnPersistenceSubFeatures = decision(”Persistence Mechanism”){
    ”XML” : deterministic(0);
    ”Database” : deterministic(5);
}
DefectCountOnFeatureSecurity = DefectCountOnSecurity + DefectCountOnSecuritySubFea-
tures;
DefectCountOnSecurity = deterministic(0);
DefectCountOnSecuritySubFeatures = decision-subset(+)(”Security Mechanism”){
    ”Data Storage” : deterministic( 3);
    ”Data Transfer” : deterministic(4);
    ”User Authentication” : deterministic(4);
}
DefectCountOnFeaturePerformance = DefectCountOnPerformance + DefectCountOnPerfor-
manceSubFeatures;
DefectCountOnPerformance = deterministic(0);
DefectCountOnPerformanceSubFeatures = decision(”Feature Performance”){
    ”Milli Sec” : deterministic(0);
    ”Seconds” : deterministic(6);
    ”Minutes” : deterministic(6);
}
```

```
FeatureCount = FeatureWebServerCount + CountOfOtherServices;
FeatureWebServerCount = ProtocolAndLoggingCount +FeatureContentCount;
ProtocolAndLoggingCount = **decision-subset**(+)(”Protocol And Logging”){
    ”Logging”: FeatureLoggingCount;
    ”Protocols” : FeatureProtocolsCount;
}
FeatureLoggingCount = **decision**(”Feature Logging”){
    ”DB” : CountOfUsingDB;
    ”File” : CountOfUsingFile;
}
CountOfUsingDB = **deterministic**(1);
CountOfUsingFile = **deterministic**(1);
FeatureProtocolsCount = **decision-subset**(+) (”Feature Protocol”){
    ”NTTP” : **deterministic**(1);
    ”FTP” : **deterministic**(1);
    ”HTTPS” : **deterministic**(1);
}
FeatureContentCount = FeatureStaticCount + FeatureActiveCount;
FeatureStaticCount = **deterministic**(1);
FeatureActiveCount = **decision-subset**(+) (”Active Features”){
    ”Active” : **deterministic**(1);
    ”ASP”: **deterministic**(1);
    ”PHP”: **deterministic**(1);
    ”JSP”: **deterministic**(1);
    ”CGI”: **deterministic**(1);
}
CountOfOtherServices = **decision-subset**(+)(”Other Services”){
    ”Additional Services”: FeatureAdditionalServicesCount;
    ”Security” : FeatureSecurityCount;
    ”Persistence” : FeaturePersistenceCount;
    ”Performance” : FeaturePerformanceCount;
}
```

```
FeatureAdditionalServicesCount = decision-subset(+)("Feature Additional Services"){
    "Additional Services" : deterministic(1);
    "Site Statistics" : FeatureSiteStatisticsCount;
    "Site Search" : FeatureSiteSearchCount;
    "Ad Server" : FeatureAdServerCount;
}
FeatureSiteStatisticsCount = FeatureStatisticsCount                    + FeatureBasicStat-
Count                          + FeatureAdvancedStatCount;

FeatureStatisticsCount =deterministic(1);
FeatureBasicStatCount = deterministic(1);
FeatureAdvancedStatCount = decision("Feature Advanced Stat"){
    "Without Advanced Stat" : deterministic(1);
    "With Advanced Stat" : deterministic(1);
}
FeatureSiteSearchCount = decision-subset(+)("Feature Site Search"){
    "Site Search" : SiteSearchCount;
    "Images" : FeatureImagesCount;
    "Text" : FeatureTextCount;
}

SiteSearchCount = deterministic(1);
FeatureImagesCount = deterministic(1);
FeatureTextCount = TextCount + FeatureHTMLCount + FeatureDynamicCount;
TextCount = deterministic(1);
FeatureHTMLCount = deterministic(1);
FeatureDynamicCount = decision("Feature Dynamic"){
    "Without Dynamic" : deterministic(1);
    "With Dynamic" : deterministic(1);
}
```

We model the number of feature counts i.e., the number of features as below:

```
FeatureAdServerCount = AdServerCount
                + FeatureReportsCount
                + FeatureBannersCount
                + PopupsAndKeywordSupportCount;

AdServerCount = deterministic(1);
FeatureReportsCount = deterministic(1);
FeatureBannersCount = BannersCount + FeatureImageCount + FeatureFlashCount;
BannersCount = deterministic(1);
FeatureImageCount = deterministic(1);

FeatureFlashCount = decision(”Flash”){
    ”Without Flash” : deterministic(1);
    ”With Flash” : deterministic(1);
}

PopupsAndKeywordSupportCount = decision-subset(+)(”Popups And Keyword Support”){
”Popups” : FeaturePopupsCount;
”Keyword Support” : FeatureKeywordSupportCount;
}

FeatureKeywordSupportCount =deterministic(1);
FeaturePopupsCount = deterministic(1);
FeaturePersistenceCount = PersistenceCount + PersistenceSubFeaturesCount;
PersistenceCount = deterministic(1);
PersistenceSubFeaturesCount = decision(”Persistence Mechanism”){
    ”XML” : deterministic(1);
    ”Database” : deterministic(1);
}

FeatureSecurityCount = SecurityCount + SecuritySubFeaturesCount;
SecurityCount = deterministic(1);
SecuritySubFeaturesCount = decision-subset(+)(”Security Mechanism”){
    ”Data Storage” : deterministic( 1);
    ”Data Transfer” : deterministic(1);
    ”User Authentication” : deterministic(1);
}

FeaturePerformanceCount = PerformanceCount + PerformanceSubFeaturesCount;

PerformanceCount = deterministic(1);

PerformanceSubFeaturesCount = decision(”Feature Performance”){

    ”Milli Sec” : deterministic(1);

    ”Seconds” : deterministic(1);

    ”Minutes” : deterministic(1);
```

Since we want to maximise the number of features used before as they are likely more stable big-free than a new features. Hence, we model whether a feature has been used before as below:

```
FeaturesUsedBefore = FeatureWebServerUsedBefore + UsedBeforeOfOtherServices;
FeatureWebServerUsedBefore = ProtocolAndLoggingUsedBefore +FeatureContentUsedBefore;
ProtocolAndLoggingUsedBefore = decision-subset(+)("Protocol And Logging"){
    "Logging": FeatureLoggingUsedBefore;
    "Protocols" : FeatureProtocolsUsedBefore;
}
FeatureLoggingUsedBefore = decision("Feature Logging"){
    "DB" : DBUsedBefore;
    "File" : FileUsedBefore;
}
DBUsedBefore = deterministic(1);
FileUsedBefore = deterministic(0);
FeatureProtocolsUsedBefore = decision-subset(+) ("Feature Protocol"){
    "NTTP" : deterministic(1);
    "FTP" : deterministic(1);
    "HTTPS" : deterministic(0);
}
FeatureContentUsedBefore = FeatureStaticUsedBefore + FeatureActiveUsedBefore;
FeatureStaticUsedBefore = deterministic(1);
FeatureActiveUsedBefore = decision-subset(+) ("Active Features"){
    "Active" : deterministic(0);
    "ASP": deterministic(0);
    "PHP": deterministic(1);
    "JSP": deterministic(1);
    "CGI": deterministic(1);
}
UsedBeforeOfOtherServices = decision-subset(+)("Other Services"){
    "Additional Services": FeatureAdditionalServicesUsedBefore;
    "Security" : FeatureSecurityUsedBefore;
    "Persistence" : FeaturePersistenceUsedBefore;
    "Performance" : FeaturePerformanceUsedBefore;
}
```

```
FeatureAdditionalServicesUsedBefore = decision-subset(+)("Feature Additional Services"){
    "Additional Services" : deterministic(1);
    "Site Statistics" : FeatureSiteStatisticsUsedBefore;
    "Site Search" : FeatureSiteSearchUsedBefore;
    "Ad Server" : FeatureAdServerUsedBefore;
}

FeatureSiteStatisticsUsedBefore = FeatureStatisticsUsedBefore + FeatureBasicStatUsedBefore
+ FeatureAdvancedStatUsedBefore;
FeatureStatisticsUsedBefore =deterministic(0);
FeatureBasicStatUsedBefore = deterministic(1);
FeatureAdvancedStatUsedBefore = decision("Feature Advanced Stat"){
    "Without Advanced Stat" : deterministic(0);
    "With Advanced Stat" : deterministic(1);
}
FeatureSiteSearchUsedBefore = decision-subset(+)("Feature Site Search"){
    "Site Search" : SiteSearchUsedBefore;
    "Images" : FeatureImagesUsedBefore;
    "Text" : FeatureTextUsedBefore;
}
SiteSearchUsedBefore = deterministic(1);
FeatureImagesUsedBefore = deterministic(1);
FeatureTextUsedBefore = TextUsedBefore + FeatureHTMLUsedBefore + FeatureDynami-
cUsedBefore;
TextUsedBefore = deterministic(1);
FeatureHTMLUsedBefore = deterministic(1);
FeatureDynamicUsedBefore = decision("Feature Dynamic"){
    "Without Dynamic" : deterministic(0);
    "With Dynamic" : deterministic(1);
}

FeatureAdServerUsedBefore = AdServerUsedBefore
                  + FeatureReportsUsedBefore
                  + FeatureBannersUsedBefore
                  + PopupsAndKeywordSupportUsedBefore;
```

```
AdServerUsedBefore = deterministic(1);
FeatureReportsUsedBefore = deterministic(1);
FeatureBannersUsedBefore = BannersUsedBefore
                + FeatureImageUsedBefore
                + FeatureFlashUsedBefore;
BannersUsedBefore = deterministic(1);

FeatureImageUsedBefore = deterministic(1);
FeatureFlashUsedBefore = decision(”Flash”){
    ”Without Flash” : deterministic(0);
    ”With Flash” : deterministic(1);
}
PopupsAndKeywordSupportUsedBefore = decision-subset(+)(”Popups And Keyword Sup-
port”){
    ”Popups” : FeaturePopupsUsedBefore;
    ”Keyword Support” : FeatureKeywordSupportUsedBefore;
}
FeatureKeywordSupportUsedBefore =deterministic(1);
FeaturePopupsUsedBefore = deterministic(0);
FeaturePersistenceUsedBefore = PersistenceUsedBefore + PersistenceSubFeaturesUsedBefore;
PersistenceUsedBefore = deterministic(0);
PersistenceSubFeaturesUsedBefore = decision(”Persistence Mechanism”){
    ”XML” : deterministic(0);
    ”Database” : deterministic(1);
}
FeatureSecurityUsedBefore = SecurityUsedBefore + SecuritySubFeaturesUsedBefore;
SecurityUsedBefore = deterministic(0);
SecuritySubFeaturesUsedBefore = decision-subset(+)(”Security Mechanism”){
    ”Data Storage” : deterministic( 1);
    ”Data Transfer” : deterministic(1);
    ”User Authentication” : deterministic(1);
}
FeaturePerformanceUsedBefore = PerformanceUsedBefore + PerformanceSubFeaturesUsedBe-
fore;
PerformanceUsedBefore = deterministic(0);
PerformanceSubFeaturesUsedBefore = decision(”Feature Performance”){
    ”Milli Sec” : deterministic(0);
    ”Seconds” : deterministic(1);
    ”Minutes” : deterministic(1);
}
```

```
Constraint "Security Mechanism" : "Data Transfer" requires"Feature Protocol": "HTTPS";

Constraint "Feature Logging": "DB" requires"Persistence Mechanism" : "Database";

Constraint "Feature Protocol": "HTTPS" excludes"Feature Performance" : "Milli Sec";

Constraint "Feature Logging" : "File" requires"Feature Protocol" : "FTP";

Constraint "Popups And Keyword Support" : "Keyword Support" requires"Feature Site
Search" : "Text";

Constraint "Feature Dynamic" : "With Dynamic" requires"Active Features": "Active";
```

Analysis Result

The RADAR analysis of the Web Portal decision model is presented in Fig.D.1. The problems was analysed using NSGAII using the 1+λ optimisation approach [130] and algorithmic parameters similar to the settings used in [285]: population size of 100, crossover probability of 0.9, mutation probability of 0.1 and maximum number of fitness evaluation of 50000. The optimisation results shows RADAR shortlists 76 optimal alternatives each representing different trade-offs between minimising the cost and maximising total feature count and total feature used before; minimising total defect count.

Optimisation Analysis

Objective:	**Min** MinExpectedCost
Objective:	**Max** MinExpectedTotalDefect
Objective:	**Max** MaxTotalFeaturesUsedBefore
Objective:	**Min** MaxTotalFeatureCount
Optimisation Approach	1+λ
Algorithms Settings	Algorithm (NSGAII), Pop. Size (100), P_c (0.8), P_m (0.1), Fitness Evaluation (50000)
Shortlisted	76
Nbr. Variables	168
Nbr. Decisions	14
Runtime	43

ID	Protocol And Logging	Feature Logging	Feature Protocol	Active Features	Other Services	Feature Additional Services	Feature Advanced Stat	Feature Site Search	Feature Dynamic	Flash	Popups And Keyword Support	Persistence Mechanism	Security Mechanism	Feature Performance	ExpectedCost	ExpectedTotalDefect	TotalFeaturesUsedBefore	TotalFeatureCount
1	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	—	Minutes	44	16	4	4
2	Logging;Protocols	File	FTP;HTTPS	Active;ASP;JSP	Additional Services;Security;Persistence;Performance	Additional Services;Ad Server	With Advanced Stat	Images	Without Dynamic	Without Flash	—	Database	Data Storage;Data Transfer	Minutes	167	55	12	20
3	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Minutes	44	16	4	4
4	Logging	File	FTP;HTTPS	ASP	Security;Persistence	Site Search	With Advanced Stat	Site Search	Without Dynamic	Without Flash	—	Database	—	Seconds	48	8	2	6
5	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	Without Flash	Keyword Support	Database	Data Storage;User Authentication	Minutes	44	16	4	4
6	Logging	DB	NTTP;HTTPS	Active;PHP	Security	Ad Server	Without Advanced Stat	Site Search;Text	Without Dynamic	Without Flash	Popups;Keyword Support	Database	Data Storage;Data Transfer	Minutes	64	18	5	7
7	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Keyword Support	Database	Data Storage;User Authentication	Minutes	44	16	4	4
8	Protocols	DB	—	ASP	Additional Services;Security	Ad Server	Without Advanced Stat	Site Search;Text	Without Dynamic	With Flash	Keyword Support	Database	Data Storage;User Authentication	Minutes	93	36	9	11
9	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Keyword Support	Database	User Authentication	Minutes	44	16	4	4
10	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	With Flash	Keyword Support	Database	—	Minutes	44	16	4	4
11	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Minutes	34	11	3	3
12	Logging	File	NTTP;FTP	ASP	Security	Additional Services;Site Search	Without Advanced Stat	Site Search	Without Dynamic	With Flash	—	Database	—	Seconds	31	3	1	4
13	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Keyword Support	Database	—	Seconds	44	16	4	4
14	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	Without Advanced Stat	Text	Without Dynamic	Without Flash	Keyword Support	Database	—	Seconds	44	16	4	4
15	—	File	NTTP;FTP;HTTPS	PHP;CGI	—	Additional Services;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Keyword Support	Database	Data Storage;User Authentication	Minutes	34	11	3	3
16	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Text	Without Dynamic	With Flash	—	Database	Data Storage	Seconds	44	16	4	4
17	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	Without Flash	—	Database	Data Storage	Minutes	44	16	4	4
18	Logging	File	FTP;HTTPS	Active;ASP	Security;Performance	Additional Services;Site Search	Without Advanced Stat	Site Search	Without Dynamic	With Flash	—	Database	—	Minutes	56	9	2	7
19	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	Without Advanced Stat	Text	Without Dynamic	Without Flash	Keyword Support	Database	—	Minutes	44	16	4	4
20	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Keyword Support	Database	—	Seconds	34	11	3	3
21	Logging	File	FTP;HTTPS	ASP	Security;Persistence	Site Search	With Advanced Stat	Site Search	Without Dynamic	Without Flash	—	XML	Data Storage;Data Transfer	Minutes	70	10	3	8
22	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Keyword Support	Database	Data Storage;User Authentication	Seconds	34	11	3	3
23	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Keyword Support	Database	Data Storage;User Authentication	Minutes	34	11	3	3
24	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	Without Advanced Stat	Text	Without Dynamic	Without Flash	Keyword Support	Database	—	Seconds	34	11	3	3
25	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Text	Without Dynamic	Without Flash	—	Database	Data Storage	Minutes	34	11	3	3
26	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Text	Without Dynamic	With Flash	Keyword Support	Database	Data Storage;User Authentication	Minutes	34	11	3	3
27	—	File	NTTP;FTP;HTTPS	PHP;CGI	—	Additional Services;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	Without Flash	—	Database	User Authentication	Seconds	34	11	3	3
28	Logging	File	FTP;HTTPS	Active;ASP	Security;Performance	Site Statistics	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	Data Storage;Data Transfer	Minutes	71	16	4	9
29	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	Without Flash	Keyword Support	Database	Data Storage;User Authentication	Minutes	34	11	3	3
30	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Minutes	44	16	4	4
31	—	File	NTTP;FTP;HTTPS	PHP;CGI	—	Ad Server	Without Advanced Stat	Images	Without Dynamic	Without Flash	—	XML	Data Storage	Minutes	34	11	3	3
32	Logging	File	FTP;HTTPS	ASP	Additional Services;Security	Ad Server	With Advanced Stat	Site Search;Text	Without Dynamic	Without Flash	—	Database	Data Storage	Minutes	75	22	6	10
33	Logging	File	NTTP;FTP	ASP	Security	Additional Services;Site Search	Without Advanced Stat	Site Search	Without Dynamic	Without Flash	—	Database	Data Storage	Seconds	40	6	2	5
34	—	DB	—	PHP;CGI	Security	Ad Server	With Advanced Stat	Text	Without Dynamic	With Flash	Popups;Keyword Support	Database	Data Storage	Minutes	48	14	4	5
35	Logging	File	FTP;HTTPS	ASP	Security;Persistence	Site Search	With Advanced Stat	Site Search	Without Dynamic	With Flash	Popups	Database	—	Minutes	48	8	2	6
36	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	Without Flash	—	Database	—	Seconds	44	16	4	4
37	—	DB	—	PHP;CGI	—	Additional Services;Site Statistics;Site Search	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Seconds	34	11	3	3
38	—	File	NTTP;FTP;HTTPS	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Minutes	34	11	3	3
39	Logging	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Keyword Support	Database	Data Storage	Minutes	44	16	4	4
40	Logging	File	FTP;HTTPS	Active;ASP	Security;Performance	Ad Server	With Advanced Stat	Site Search;Text	Without Dynamic	With Flash	Popups;Keyword Support	Database	Data Storage;Data Transfer	Minutes	71	16	4	9
41	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Milli Sec	44	16	4	4
42	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	With Flash	Keyword Support	Database	Data Storage	Minutes	44	16	4	4
43	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	Without Flash	Keyword Support	Database	Data Storage	Minutes	44	16	4	4
44	—	File	NTTP;FTP;HTTPS	PHP;CGI	—	Additional Services;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups;Keyword Support	Database	Data Storage;Data Transfer	Minutes	34	11	3	3
45	—	File	NTTP;FTP;HTTPS	PHP;CGI	—	Ad Server	Without Advanced Stat	Site Search	Without Dynamic	Without Flash	—	Database	—	Seconds	34	11	3	3
46	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	Without Advanced Stat	Text	Without Dynamic	With Flash	Keyword Support	Database	—	Minutes	44	16	4	4
47	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Seconds	44	16	4	4
48	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Minutes	34	11	3	3
49	—	DB	—	PHP;CGI	—	Additional Services;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Minutes	34	11	3	3
50	Logging	File	FTP;HTTPS	Active;ASP	Security;Performance	Additional Services;Site Search	Without Advanced Stat	Site Search	Without Dynamic	With Flash	—	XML	—	Minutes	56	9	2	7
51	Logging	File	FTP;HTTPS	Active;ASP	Security;Performance	Ad Server	With Advanced Stat	Site Search;Text	With Dynamic	With Flash	Keyword Support	Database	—	Minutes	56	9	2	7
52	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	—	Seconds	44	16	4	4
53	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	Without Flash	—	Database	Data Storage	Seconds	44	16	4	4
54	Logging	File	FTP;HTTPS	ASP	Security;Persistence	Site Search	With Advanced Stat	Site Search;Text	Without Dynamic	Without Flash	—	Database	Data Storage	Minutes	57	11	3	7
55	—	File	NTTP;FTP;HTTPS	PHP;CGI	—	Site Statistics	With Advanced Stat	Text	Without Dynamic	Without Flash	—	XML	—	Seconds	34	11	3	3
56	—	File	NTTP;FTP;HTTPS	PHP;CGI	—	Ad Server	Without Advanced Stat	Site Search	Without Dynamic	With Flash	Popups	Database	—	Minutes	34	11	3	3
57	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images	Without Dynamic	Without Flash	—	Database	Data Storage	Minutes	34	11	3	3
58	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Seconds	34	11	3	3
59	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	With Flash	Popups	Database	User Authentication	Minutes	34	11	3	3
60	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Text	Without Dynamic	With Flash	Popups	Database	User Authentication	Minutes	44	16	4	4
61	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	Without Flash	—	Database	Data Storage	Minutes	44	16	4	4
62	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	Without Flash	—	Database	User Authentication	Minutes	44	16	4	4
63	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	Without Advanced Stat	Text	Without Dynamic	Without Flash	Keyword Support	Database	Data Storage	Minutes	44	16	4	4
64	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	Without Advanced Stat	Text	Without Dynamic	Without Flash	Keyword Support	Database	User Authentication	Minutes	44	16	4	4
65	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	Without Flash	—	Database	Data Storage	Seconds	34	11	3	3
66	—	File	NTTP;FTP;HTTPS	PHP;CGI	—	Ad Server	Without Advanced Stat	Images	Without Dynamic	Without Flash	—	Database	—	Seconds	34	11	3	3
67	Logging;Protocols	DB	—	PHP;CGI	—	Ad Server	With Advanced Stat	Images;Text	Without Dynamic	Without Flash	Popups	Database	Data Storage	Minutes	44	16	4	4
68	Logging;Protocols	DB	—	PHP;CGI	—	Additional Services;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Milli Sec	44	16	4	4
69	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	—	Minutes	34	11	3	3
70	Logging;Protocols	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	Without Flash	Popups	Database	Data Storage	Minutes	44	16	4	4
71	Logging	File	FTP;HTTPS	Active;ASP	Security;Performance	Ad Server	With Advanced Stat	Site Search;Text	With Dynamic	With Flash	—	XML	—	Minutes	56	9	2	7
72	Logging	File	FTP;HTTPS	ASP	Security;Persistence	Site Search	With Advanced Stat	Site Search;Text	Without Dynamic	With Flash	Keyword Support	Database	—	Minutes	48	8	2	6
73	—	DB	—	PHP;CGI	—	Site Statistics;Site Search;Ad Server	With Advanced Stat	Site Search	Without Dynamic	With Flash	Popups	Database	—	Minutes	34	11	3	3
74	Logging	File	FTP;HTTPS	ASP	Security;Performance	Site Statistics	Without Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	Data Storage;Data Transfer	Minutes	59	16	4	8
75	Logging;Protocols	DB	—	PHP;CGI	—	Additional Services;Site Statistics;Ad Server	Without Advanced Stat	Text	Without Dynamic	Without Flash	Keyword Support	Database	—	Minutes	44	16	4	4
76	—	DB	NTTP;FTP;HTTPS	PHP;CGI	—	Additional Services;Site Search;Ad Server	With Advanced Stat	Site Search;Images;Text	Without Dynamic	With Flash	Popups	Database	Data Storage	Minutes	34	11	3	3

FIGURE D.1: Analysis results for the Web Portal

Appendix E

Modelling and Analysing Configuration Decisions in AWS

Modelling Optimisation Objectives

The optimisation objectives is given below:

Objective **Max** ExpectedFeatureCountMonth = **EV**(FeatureCount);

Objective **Min** ExpectedCostMonth = **EV**(TotalCostMonth);

Objective **Max** ExpectedInstanceCores = **EV**(TotalInstanceCores);

Objective **Max** ExpectedInstanceECU = **EV**(TotalInstanceECU);

Objective **Max** ExpectedInstanceRAM = **EV**(TotalInstanceRAM);

Objective **Min** ExpectedCostHour = **EV**(TotalCostHour);

Objective **Max** ExpectedInstanceSSDBacked = **EV**(TotalInstanceSSDBacked);

In the above expression, the first objective is a maximisation of the features count to be implemented; minimisation of the feature costs; maximisation of the number of cores; maximisation of the RAM usage; minimisation of cost of implementing the features; and maximisation of the number of SSD used.

We model the feature count as below:

```
FeatureCount = FeatureOSCount
                            + FeatureBlockStorageCount
                            + FeatureInstanceCount
                            + FeatureDedicationCount
                            + FeatureLocationCount
                            + FeaturePurchaseCount;

FeatureOSCount = decision(”OS”){
     “Windows Based” : FeatureWindowBasedOSCount + FeatureWindowsBasedServerCount;
     “Linux Based” : FeatureLinuxBasedOSCount ;
}
FeatureWindowBasedOSCount = deterministic(1);
FeatureWindowsBasedServerCount = FeatureWindowsServerCount + FeatureSQLServerCount;
FeatureWindowsServerCount = deterministic(1);
FeatureSQLServer =decision-subset(+)(”Feature SQL Server”){
     “SQL Server” : FeatureSQLServerOptionCount;
}
```

```
FeatureSQLServerOptionCount = decision(”SQL Server”){
    “Web” : deterministic(1);
    “Std” : deterministic(1);
}
FeatureLinuxBasedOSCount = decision(”Linux OS”){
    “Suse” : deterministic(1);
    “Amazon Linux” : deterministic(1);
    “RedHat” : deterministic(1);
}
FeatureBlockStorageCount = decision-subset(+)(”Feature Block Storage”){
    “Block Storage” : FeatureBlockStorageOptionCount;
}
FeatureBlockStorageOptionCount = decision(”Block Storage”){
    “Std” : deterministic(1);
    “SSD” : deterministic(1);
}
FeatureInstanceCount = decision(”Instance”){
    “Memory Opt” : deterministic(1);
    “General” : deterministic(1);
    “Compute Opt” : deterministic(1);
    “Storage Opt” : deterministic(1);
    “GPU” : deterministic(1);
}
```

```
FeatureDedicationCount = decision(”Dedication”){
    “Public” : deterministic(1);
    “Ded” : deterministic(1);
}
FeatureLocationCount = decision(”Location”){
    “Sau Paulo”: deterministic(1);
    “North America” : FeatureNorthAmericaCount;
    “Asia Oceania” : FeatureAsiaOceaniaCount;
    “IR” : deterministic(1);
} FeatureNorthAmericaCount = decision(”North America”){
    “VA” : deterministic(1);
    “CA” : deterministic(1);
    “OR” : deterministic(1);
}
FeatureAsiaOceaniaCount = decision(”Asia Oceania”){
    “Sin” : deterministic(1);
    “JP” : deterministic(1);
    “Aus” : deterministic(1);
}
FeaturePurchaseCount = decision(”Purchase”){
    “On Demand” : deterministic(1);
    “Reserved” : FeatureReservedCount;
}
FeatureReservedCount = ReservedFeatureCount + FeatureReservedSubFeatureCount;
FeatureReservedSubFeatureCount = FeatureYearCount + FeatureModeCount;
FeatureYearCount = decision(”Years”){
    “1Year” : deterministic(1);
    “3Years” : deterministic(1);
}
FeatureModeCount = decision(”Mode”){
    “Light” : deterministic(1);
    “Heavy” : deterministic(1);
    “Med” : deterministic(1);
}
```

We model the monthly cost as:

```
TotalCostMonth = FeatureOSCostMonth
                  + FeatureBlockStorageCostMonth
                  + FeatureDedicationCostMonth
                  + FeatureLocationCostMonth
                  + FeaturePurchaseCostMonth;

FeatureOSCostMonth = decision(”OS”){
   “Windows Based” : FeatureWindowBasedOSCostMonth + FeatureWindowsBasedServer-
CostMonth;
   “Linux Based” : FeatureLinuxBasedOSCostMonth ;
}
FeatureWindowBasedOSCostMonth =triangular(800, 1000, 1200);
FeatureWindowsBasedServerCostMonth = FeatureWindowsServerCostMonth + Fea-
tureSQLServerCostMonth; FeatureWindowsServerCostMonth = triangular(400,500,600);
FeatureSQLServerCostMonth =decision-subset(+)(”Feature SQL Server”){
   “SQL Server” : FeatureSQLServerOptionCostMonth;
}
FeatureSQLServerOptionCostMonth = decision(”SQL Server”){
   “Web” : triangular(1300, 1500, 1700);
   “Std” : triangular(2000, 2500, 3000);
}
FeatureLinuxBasedOSCostMonth = decision(”Linux OS”){
   “Suse” : triangular(1500, 2000, 2500);
   “Amazon Linux” : triangular(2000, 2500, 3000);
   “RedHat” : triangular(1000, 1500, 2000);
}
FeatureBlockStorageCostMonth = decision-subset(+)(”Feature Block Storage”){
   “Block Storage” : FeatureBlockStorageOptionCostMonth;
}
```

```
FeatureBlockStorageOptionCostMonth = decision(”Block Storage”){
    “Std” : triangular(4500, 5000, 5500);
    “SSD” : triangular(9000, 10000, 11000);
}
FeatureDedicationCostMonth = decision(”Dedication”){
    “Public” : triangular(8000, 10000, 12000);
    “Ded” : triangular(15000, 20000, 25000);
}
FeatureLocationCostMonth = decision(”Location”){
    “Sau Paulo”: deterministic(100);
    “North America” : FeatureNorthAmericaCostMonth;
    “Asia Oceania” : FeatureAsiaOceaniaCostMonth;
    “IR” : triangular(300, 500, 700);
}
FeatureNorthAmericaCostMonth = decision(”North America”){
    “VA” : triangular(100, 150, 200);
    “CA” : triangular(150, 180, 200);
    “OR” : triangular(150, 200, 250);
}
FeatureAsiaOceaniaCostMonth = decision(”Asia Oceania”){
    “Sin” : triangular(200, 250, 300);
    “JP” : triangular(200, 270, 300);
    “Aus” : triangular(100, 300, 500);
}
```

```
FeaturePurchaseCostMonth = decision(”Purchase”){
    “On Demand” : triangular(800, 1000, 1200);
    “Reserved” : FeatureReservedCostMonth;
}
FeatureReservedCostMonth = ReservedFeatureCostMonth + FeatureReservedSubFeatureCost-
Month;
FeatureReservedSubFeatureCostMonth = FeatureYearCostMonth + FeatureModeCostMonth;
FeatureYearCostMonth = decision(”Years”){
    “1Year” : triangular(800, 1000, 1200);
    “3Years” : triangular(1000, 1500, 2000);
}
FeatureModeCostMonth = decision(”Mode”){
    “Light” : triangular(1000, 1500, 2000);
    “Heavy” : triangular(6000, 7000, 8000);
    “Med” : triangular(3000, 3500, 4000);
}
```

```
TotalInstanceCores = decision(”Instance”){
    “Memory Opt” : deterministic(8);
    “General” : deterministic(16);
    “Compute Opt” : deterministic(8);
    “Storage Opt” : deterministic(16);
    “GPU” : deterministic(32);
}
TotalInstanceECU = decision(”Instance”){
    “Memory Opt” : deterministic(12);
    “General” : deterministic(36);
    “Compute Opt” : deterministic(48);
    “Storage Opt” : deterministic(60);
    “GPU” : deterministic(108);
}
TotalInstanceRAM = decision(”Instance”){
    “Memory Opt” : deterministic(32);
    “General” : deterministic(160);
    “Compute Opt” : deterministic(64);
    “Storage Opt” : deterministic(128);
    “GPU” : deterministic(250);
}
TotalCostHour = decision(”Instance”){
    “Memory Opt” : deterministic(5);
    “General” : deterministic(15);
    “Compute Opt” : deterministic(10);
    “Storage Opt” : deterministic(8);
    “GPU” : deterministic(20);
}
TotalInstanceBacked = decision(”Instance”){
    “Memory Opt” : deterministic(15);
    “General” : deterministic(30);
    “Compute Opt” : deterministic(60);
    “Storage Opt” : deterministic(120);
    “GPU” : deterministic(240);
}
```

Analysis Result

The RADAR analysis of the AWS decision model is presented in Fig.E.1. The problems was analysed using NSGAII using the 1+λ optimisation approach [130] and algorithmic parameters similar to the settings used in [285]: population size of 100, crossover probability of 0.9, mutation probability of 0.1 and maximum number of fitness evaluation of 50000. The optimisation results shows RADAR shortlists 68 optimal alternatives: each including the "public" option of the Dedication decision and "sau paulo" option of Location decision. But when these options are selected, the shortlist includes different combinations of other options; each combination representing a different tradeoffs between maximising the expected feature count; minimising the cost per month and per hour and maximising the number of instance cores, ECUS and SSD, RAMS.

Optimisation Analysis

Objective:	**Max** ExpectedFeatureCountMonth
Objective:	**Min** ExpectedCostMonth
Objective:	**Max** ExpectedFeatureCountMonth
Objective:	**Min** ExpectedCostMonth
Objective:	**Max** ExpectedFeatureCountMonth
Optimisation Approach:	1+λ
Algorithm Settings:	Algorithm (NSGAII), Pop. Size (100), P_c (0.9), P_m (0.1), Fitness Evaluations (50000)
Shortlisted:	68
Nbr. Decisions:	14
Nbr. Variables:	68
Runtime(s) :	21

ID	OS	SQL Server	Linux OS	Feature Block Storage	Block Storage	Instance	Dedication	Location	North America	Asia Oceania	Purchase	Years	Mode	ExpectedFeatureCountMonth	ExpectedCostMonth	ExpectedInstanceCores	ExpectedInstanceECU	ExpectedInstanceRAM	ExpectedCostHour	ExpectedInstanceSSDBacked
1	Windows Based	Web	Amazon Linux	—	Std	GPU	Public	Sau Paulo	CA	Sin	On Demand	3Years	Med	6	12596	32	108	250	20	240
2	Windows Based	Web	RedHat	—	Std	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	1Year	Med	6	12596	8	12	32	5	15
3	Windows Based	Web	Amazon Linux	Block Storage	Std	Storage Opt	Public	Sau Paulo	CA	JP	Reserved	1Year	Light	8	19094	16	60	128	8	120
4	Linux Based	Web	RedHat	—	Std	Memory Opt	Public	Sau Paulo	VA	JP	On Demand	3Years	Med	5	12595	8	12	32	5	15
5	Windows Based	Std	Amazon Linux	—	SSD	Storage Opt	Public	Sau Paulo	VA	Aus	On Demand	1Year	Light	6	12596	16	60	128	8	120
6	Windows Based	Web	RedHat	—	Std	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	3Years	Heavy	6	12596	8	12	32	5	15
7	Windows Based	Std	Amazon Linux	—	SSD	Storage Opt	Public	Sau Paulo	VA	Sin	On Demand	3Years	Med	6	12596	16	60	128	8	120
8	Windows Based	Web	RedHat	—	Std	General	Public	Sau Paulo	VA	JP	On Demand	1Year	Med	6	12596	16	36	160	15	30
9	Windows Based	Std	Amazon Linux	—	Std	Memory Opt	Public	Sau Paulo	OR	JP	On Demand	3Years	Light	6	12596	8	12	32	5	15
10	Linux Based	Web	RedHat	—	Std	Memory Opt	Public	Sau Paulo	CA	JP	On Demand	3Years	Heavy	5	12595	8	12	32	5	15
11	Windows Based	Std	Amazon Linux	—	SSD	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	3Years	Heavy	6	12596	8	12	32	5	15
12	Windows Based	Std	Amazon Linux	—	SSD	General	Public	Sau Paulo	CA	Aus	On Demand	3Years	Light	6	12596	16	36	160	15	30
13	Windows Based	Std	Amazon Linux	—	SSD	Storage Opt	Public	Sau Paulo	OR	Sin	On Demand	3Years	Med	6	12596	16	60	128	8	120
14	Windows Based	Std	Amazon Linux	—	SSD	Storage Opt	Public	Sau Paulo	OR	Aus	On Demand	1Year	Light	6	12596	16	60	128	8	120
15	Linux Based	Web	RedHat	—	Std	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	3Years	Heavy	5	12595	8	12	32	5	15
16	Windows Based	Std	Suse	—	Std	GPU	Public	Sau Paulo	VA	JP	On Demand	1Year	Med	6	12596	32	108	250	20	240
17	Windows Based	Std	Amazon Linux	Block Storage	Std	Storage Opt	Public	Sau Paulo	CA	JP	Reserved	1Year	Light	8	19094	16	60	128	8	120
18	Windows Based	Web	Amazon Linux	—	SSD	Storage Opt	Public	Sau Paulo	VA	Aus	On Demand	1Year	Light	6	12596	16	60	128	8	120
19	Windows Based	Web	Amazon Linux	—	SSD	General	Public	Sau Paulo	CA	Aus	On Demand	3Years	Light	6	12596	16	36	160	15	30
20	Windows Based	Web	RedHat	—	Std	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	3Years	Med	6	12596	8	12	32	5	15
21	Linux Based	Web	RedHat	—	Std	Storage Opt	Public	Sau Paulo	CA	Sin	On Demand	3Years	Med	5	12595	16	60	128	8	120
22	Windows Based	Web	Suse	—	Std	GPU	Public	Sau Paulo	CA	Sin	On Demand	3Years	Med	6	12596	32	108	250	20	240
23	Windows Based	Std	RedHat	—	SSD	General	Public	Sau Paulo	CA	Aus	On Demand	3Years	Med	6	12596	16	36	160	15	30
24	Windows Based	Std	Amazon Linux	—	Std	GPU	Public	Sau Paulo	OR	Aus	On Demand	1Year	Heavy	6	12596	32	108	250	20	240
25	Windows Based	Web	RedHat	—	Std	General	Public	Sau Paulo	VA	Sin	On Demand	1Year	Med	6	12596	16	36	160	15	30
26	Windows Based	Std	Amazon Linux	—	SSD	Storage Opt	Public	Sau Paulo	CA	JP	On Demand	3Years	Heavy	6	12596	16	60	128	8	120
27	Windows Based	Web	RedHat	—	Std	Storage Opt	Public	Sau Paulo	VA	Aus	On Demand	1Year	Light	6	12596	16	60	128	8	120
28	Windows Based	Std	RedHat	—	SSD	General	Public	Sau Paulo	VA	JP	On Demand	1Year	Med	6	12596	16	36	160	15	30
29	Windows Based	Web	RedHat	—	SSD	Storage Opt	Public	Sau Paulo	CA	JP	On Demand	3Years	Heavy	6	12596	16	60	128	8	120
30	Windows Based	Std	Amazon Linux	—	Std	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	1Year	Med	6	12596	8	12	32	5	15
31	Windows Based	Std	Amazon Linux	—	SSD	Storage Opt	Public	Sau Paulo	VA	Aus	On Demand	3Years	Light	6	12596	16	60	128	8	120
32	Windows Based	Web	RedHat	—	Std	General	Public	Sau Paulo	CA	JP	Reserved	1Year	Light	7	14095	16	36	160	15	30
33	Windows Based	Web	RedHat	—	SSD	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	3Years	Med	6	12596	8	12	32	5	15
34	Windows Based	Std	Amazon Linux	—	SSD	Storage Opt	Public	Sau Paulo	CA	JP	On Demand	3Years	Light	6	12596	16	60	128	8	120
35	Windows Based	Web	Amazon Linux	—	Std	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	1Year	Light	6	12596	8	12	32	5	15
36	Windows Based	Web	RedHat	—	SSD	Storage Opt	Public	Sau Paulo	CA	Aus	On Demand	3Years	Light	6	12596	16	60	128	8	120
37	Windows Based	Web	Amazon Linux	Block Storage	Std	Memory Opt	Public	Sau Paulo	CA	JP	Reserved	1Year	Light	8	19094	8	12	32	5	15
38	Windows Based	Web	Amazon Linux	—	SSD	Storage Opt	Public	Sau Paulo	VA	Sin	On Demand	3Years	Light	6	12596	16	60	128	8	120
39	Windows Based	Std	Suse	—	Std	GPU	Public	Sau Paulo	VA	JP	On Demand	3Years	Light	6	12596	32	108	250	20	240
40	Windows Based	Web	Amazon Linux	—	SSD	General	Public	Sau Paulo	CA	Aus	On Demand	1Year	Med	6	12596	16	36	160	15	30
41	Windows Based	Web	RedHat	—	Std	Storage Opt	Public	Sau Paulo	VA	Aus	On Demand	3Years	Heavy	6	12596	16	60	128	8	120
42	Windows Based	Web	RedHat	—	Std	Memory Opt	Public	Sau Paulo	VA	Sin	On Demand	1Year	Med	6	12596	8	12	32	5	15
43	Windows Based	Std	Amazon Linux	—	Std	Storage Opt	Public	Sau Paulo	VA	Aus	On Demand	1Year	Light	6	12596	16	60	128	8	120
44	Windows Based	Std	Amazon Linux	—	SSD	Storage Opt	Public	Sau Paulo	CA	Sin	On Demand	3Years	Heavy	6	12596	16	60	128	8	120
45	Windows Based	Std	Amazon Linux	—	Std	Memory Opt	Public	Sau Paulo	OR	Aus	On Demand	3Years	Med	6	12596	8	12	32	5	15
46	Windows Based	Web	RedHat	—	SSD	General	Public	Sau Paulo	CA	Aus	On Demand	3Years	Med	6	12596	16	36	160	15	30
47	Windows Based	Std	Amazon Linux	—	Std	GPU	Public	Sau Paulo	CA	Sin	On Demand	3Years	Med	6	12596	32	108	250	20	240
48	Windows Based	Web	Suse	—	Std	Memory Opt	Public	Sau Paulo	OR	JP	On Demand	3Years	Light	6	12596	8	12	32	5	15
49	Windows Based	Std	RedHat	—	Std	GPU	Public	Sau Paulo	CA	Sin	On Demand	3Years	Med	6	12596	32	108	250	20	240
50	Windows Based	Web	Suse	—	SSD	Storage Opt	Public	Sau Paulo	VA	Aus	On Demand	1Year	Light	6	12596	16	60	128	8	120
51	Windows Based	Std	Amazon Linux	—	SSD	General	Public	Sau Paulo	VA	Sin	On Demand	1Year	Med	6	12596	16	36	160	15	30
52	Windows Based	Web	Suse	—	Std	GPU	Public	Sau Paulo	VA	JP	On Demand	1Year	Med	6	12596	32	108	250	20	240
53	Windows Based	Std	Amazon Linux	—	Std	GPU	Public	Sau Paulo	OR	Aus	On Demand	1Year	Light	6	12596	32	108	250	20	240
54	Windows Based	Std	RedHat	—	SSD	Storage Opt	Public	Sau Paulo	VA	Sin	On Demand	3Years	Med	6	12596	16	60	128	8	120
55	Windows Based	Web	Amazon Linux	—	Std	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	1Year	Med	6	12596	8	12	32	5	15
56	Windows Based	Std	Suse	—	Std	GPU	Public	Sau Paulo	VA	JP	On Demand	1Year	Light	6	12596	32	108	250	20	240
57	Windows Based	Std	RedHat	—	Std	GPU	Public	Sau Paulo	CA	JP	On Demand	3Years	Heavy	6	12596	32	108	250	20	240
58	Linux Based	Web	RedHat	—	Std	Memory Opt	Public	Sau Paulo	CA	Sin	On Demand	3Years	Med	5	12595	8	12	32	5	15
59	Windows Based	Std	Amazon Linux	—	SSD	General	Public	Sau Paulo	CA	Aus	On Demand	1Year	Med	6	12596	16	36	160	15	30
60	Windows Based	Std	Amazon Linux	—	SSD	General	Public	Sau Paulo	CA	Aus	On Demand	3Years	Med	6	12596	16	36	160	15	30
61	Windows Based	Web	Amazon Linux	—	Std	General	Public	Sau Paulo	VA	JP	On Demand	1Year	Med	6	12596	16	36	160	15	30
62	Windows Based	Web	RedHat	—	Std	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	1Year	Light	6	12596	8	12	32	5	15
63	Windows Based	Web	Amazon Linux	—	Std	GPU	Public	Sau Paulo	CA	Sin	On Demand	3Years	Heavy	6	12596	32	108	250	20	240
64	Windows Based	Web	Amazon Linux	—	SSD	GPU	Public	Sau Paulo	VA	Aus	On Demand	3Years	Light	6	12596	32	108	250	20	240
65	Windows Based	Web	RedHat	—	Std	General	Public	Sau Paulo	VA	Aus	On Demand	3Years	Heavy	6	12596	16	36	160	15	30
66	Windows Based	Std	Suse	Block Storage	Std	GPU	Public	Sau Paulo	VA	Aus	Reserved	1Year	Light	8	19094	32	108	250	20	240
67	Linux Based	Web	RedHat	—	Std	Memory Opt	Public	Sau Paulo	VA	Aus	On Demand	3Years	Med	5	12595	8	12	32	5	15
68	Windows Based	Web	Suse	—	Std	GPU	Public	Sau Paulo	CA	Aus	On Demand	3Years	Light	6	12596	32	108	250	20	240

Figure E.1: Optimisation analysis results for the AWS decision model.

Appendix F

Modelling and Analysing Feature Selection in Berkeley Relational Database System

The Optimisation objectives are given below:

```
Objective  Max ExpectedNetBenefit = EV(NB);

Objective  Min ExpectedFeatureSize = EV(TotalFeatureSize);
```

We model the net benefit as:

```
NB = Revenue - TotalFeatureCost;
```

```
Revenue = normalCI(1000, 1200);
```

```
TotalFeatureCost = FeatureIndexesCost

            + NonMandatoryFeaturesCost;
```

```
NonMandatoryFeaturesCost = decision-subset (+) ("Non Mandatory Feature"){
    "Statistics" : triangular(200,250,300);
    "Cryptography" : triangular(250, 300, 350);
    "Replication" : triangular(150,200,250);
    "Verification" : triangular(50,100,150);
    "Diagnostic" : triangular (250,300,350);
    "Sequence" : triangular(40,50,60);
}
```

```
FeatureIndexesCost = CostOfFeatureIndex + FeatureIndexesSubFeatureCost;
CostOfFeatureIndex = deterministic(10);
FeatureIndexesSubFeatureCost = FeatureBTreeCost + NonMandatoryIndexSubFeatureCosts;
FeatureBTreeCost = decision("B-Tree"){
"B-Tree Fast" : triangular(45,50,55);
"B-Tree Small" : triangular(50,75,100);
}
```

```
NonMandatoryIndexSubFeatureCosts = decision-subset(+)("Index Non Mandatory SubFea-
tures"){
    "Hash" : triangular(100, 125,150);
    "Queue" : triangular(40,50,60);
}
```

We model the feature size as:

```
TotalFeatureSize = FeatureIndexesSize
                        + NonMandatoryFeaturesSIze;
```

```
NonMandatoryFeaturesSIze = decision-subset(+) ("Non Mandatory Feature"){
    "Statistics" : deterministic(285);
    "Cryptography" : deterministic(19);
    "Replication" : deterministic(89);
    "Verification" : deterministic(50);
    "Diagnostic" : deterministic(191);
    "Sequence" : deterministic(50);
}
```

```
FeatureIndexesSIze = SIzeOfFeatureIndex + FeatureIndexesSubFeatureSIze;
SIzeOfFeatureIndex = deterministic(1);
FeatureIndexesSubFeatureSIze = FeatureBTreeSIze + NonMandatoryIndexSubFeatureSIzes;
FeatureBTreeSIze = decision ("B-Tree"){
    "B-Tree Fast" : deterministic(1800);
    "B-Tree Small" : deterministic(340);
}
NonMandatoryIndexSubFeatureSIzes = decision-subset(+)("Index Non Mandatory SubFea-
tures"){
"Hash" : deterministic(113);
"Queue" : deterministic(58);
}
```

The RADAR analysis of the BerkeleyDB decision model is presented in Fig.F.1. The problems was analysed using NSGAII using the 1+λ optimisation approach [130] and algorithmic parameters similar to the settings used in [285]: population size of 100, crossover probability of 0.9, mutation probability of 0.1 and maximum number of fitness evaluation of 50000. The optimisation results shows RADAR shortlists two optimal alternatives which is to select B-Tree Small or B-Tree Fast options.

Optimisation Analysis

Objective:	**Max** ExpectedNetBenefit
Objective:	**Min** ExpectedFeatureSize
Optimisation Approach	1+λ
Algorithm Name	NSGAII
Population Size	100
Crossover Probability	0.8
Mutation Probability	0.1
Max Nbr. Evaluations	50000
Shortlisted	2
Nbr. Variables	16
Nbr. Parameters	11
Nbr. Decisions	3
Runtime(s)	4

ID	Non Mandatory Feature	B-Tree	Index Non Mandatory SubFeatures	ExpectedNetBenefit	ExpectedFeatureSize
1	–	B-Tree Small	–	1015	341
2	–	B-Tree Fast	–	1040	1801

Figure F.1: Analysis results for the Berkeley DB

Appendix G

Modelling and Analysing the Requirements Selection of Microsoft Word Processor

Modelling the Optimisation Objectives

The optimisation objectives are given below:

```
Objective  Max ENB = EV(NB);
Objective  Min ProjectRisk = Pr(Failure);
```

In the above expression, the first objective is a maximisation of the expected net benefit (ENB) from selecting a subset of requirements, and the second objective is a minimisation of the project risk.

This net benefit is the difference between the benefit of implementing the requirements and the cost of the requirements. We define the NB below:

```
NB = Value - Cost;
```

The Value derived from the subset of requirements implemented is estimated considering uncertainty in the perceived values supplied by the stakeholders. We use probability distributions to capture stakeholders' value scores for each requirement.

We model the Value as below:

```
Value = decision-subset(+)("Next Release"){
        "NewFile": normalCI(10,20);
        "OpenFile" : normalCI(30,40);
        "CloseFile" : normalCI(10,30);
        "SaveFile" : normalCI(70,80);
        "Save As Different File" : normalCI(5,10);
        "SearchFile" : normalCI(30,50);
        "ProtectFile" : normalCI(10,50);
        "PrintPreview" : normalCI(5,10);
        "Print File" : normalCI(50,80);
        "Send To" : normalCI(10,80);
        "Set Properties" : normalCI(10,20,70);
        "Exit" : normalCI(30,90);
        "Undo Task" : normalCI(50,60);
        "Redo Task" : normalCI(20,60);
        "Cut" : normalCI(60,70);
        "Copy" : normalCI(60,70);
        "Paste" : normalCI(30,60);
        "Paste Special" : normalCI(20,40);
        "Go To" : normalCI(10,20);
        "Find" : normalCI(40,50);
        "Replace" : normalCI(50,60);
        "Select All" : normalCI(10,90);
        "Default" : normalCI(10,30);
        "Print Layout" : normalCI(30,40);
        "Web Layout" : normalCI(10,40);
        "Zoom" : normalCI(10,50);
        "Header Footer" : normalCI(30,90);
        "Page Numbers" : normalCI(20,90);
        "Date Time" : normalCI(50,90);
        "Symbol" : normalCI (30,90);
        "Bookmark" : normalCI(20,60);
        "Hyper Link" : normalCI(20,90);
        "Font" : normalCI(40,90);
        "Paragraph" : normalCI(30,90);
        "Bullets Numbering" : normalCI(30,90);
        "Change Case" : normalCI(10,80);
        "Background" : normalCI(10,20);
        "Spell Check" : normalCI(60,90);
        "Check Grammer" : normalCI(10,90);
        "Speech" : normalCI(10,20);
        "Mail Merge" : normalCI(10,70);
        "Macro" : normalCI(10,60);
        "Set Options" : normalCI(10,80);
        "Insert Table" : normalCI(40,90);
        "Delete Table" : normalCI(50,90);
        "Table Format" : normalCI(50,90);
        "Sort" : normalCI(30,90);
        "Import Data" : normalCI(30,60);
        "Help" : normalCI(20,90);
        "Search" : normalCI(10,90);
  }
```

In the above expression, each option represents a feature (requirements) to be implemented. The total Value derived from these features is the sum of the assigned value corresponding to a selected requirement.

The total Cost (£) is a function of the requirements elicitation cost (ElicitationCost), the design cost (DesignCost), the development cost (DevCost), the cost accrued from external tool development (ExtToolDev), the cost due to software testing (CoTesting-Cost).

```
Cost = ElicitationCost
                 + DesignCost
                 + DevCost
                 + ExtToolDevCost
                 + TestingCost;
```

Modelling Risk Of Exceeding Budget

```
ProjectFailure = 1 − ( (1 − RiskExceedingElicitationCost) ×
                        (1 − RiskExceedingDesignCost) ×
                        (1 − RiskExceedingDevCost) ×
                        (1 − RiskExceedingExtToolDevCost) ×
                        (1 − RiskExceedingTestingCost);
```

We model the risk of exceeding the elicitation cost as below:

```
RiskExceedingElicitationCost = ElicitationCost > ElicitationBudget;
ElicitationBudget = deterministic(150);
```

We model the risk of exceeding the design cost as below:

```
RiskExceedingDesignCost = DesignCost > DesignBudget;
DesignBudget = deterministic(150);
```

We model the risk of exceeding the development cost as below:

```
RiskExceedingDevCost = DevCost > DevelopmentBudget;

DevelopmentBudget = deterministic(200);
```

The risk of exceeding external tool development cost is given as:

```
RiskExceedingExtToolDevCost = ExtToolDevCost > ExternalToolDevelopmentBudget;

ExternalToolDevelopmentBudget = deterministic(75);
```

The risk of exceeding testing cost is given below:

```
RiskExceedingTestingCost = TestingCost > TestingBudget;

TestingBudget = deterministic(150);
```

We model the Requirements Elicitation Cost as:

```
ElicitationCost = decision-subset(+)(”Next Release”){
    ”NewFile”: triangular(14,15,16);
        ”OpenFile” : triangular(17,18, 19);
        ”CloseFile” : triangular(2,3, 4);
        ”SaveFile” : triangular(12,15,17);
        ”Save As Different File” : triangular(15, 16,17);
        ”SearchFile” : triangular(19, 20, 21);
        ”ProtectFile” : triangular(4,5,6);
        ”PrintPreview” : triangular(13,15, 17);
        ”Print File” : triangular(11, 12, 13);
        ”Send To” : triangular(9, 10, 11);
        ”Set Properties” : triangular(3,5,7);
        ”Exit” : triangular(2,3,4);
        ”Undo Task” : triangular(6,7,8);
        ”Redo Task” : triangular(4,6,8);
        ”Cut” : triangular(5,6,7);
        ”Copy” : triangular(3,5,7);
        ”Paste” : triangular(6,7, 8);
        ”Paste Special” : triangular(11,12,13);
        ”Go To” : triangular(6,7,9);
        "Find" : triangular(3,6,9);
        "Replace" : triangular(3,5,7);
        "Select All" : triangular(3,5,7);
        "Default" : triangular(2,4,6);
        "Print Layout" : triangular(2,4,6);
        "Web Layout" : triangular(2,4,6);
        "Zoom" : triangular(4,6,8);
        "Header Footer" : triangular(1,2,4);
        "Page Numbers" : triangular(1,2, 4);
        "Date Time" : triangular(1,2, 4);
        "Symbol" : triangular(1,2, 4);
        "Bookmark" : deterministic(1);
        "Hyper Link" : deterministic(1);
        "Font" : triangular(1,3,5);
        "Paragraph" : triangular(3,5,7);
        "Bullets Numbering" : triangular(2,4,6);
        "Change Case" : triangular(2,4,6);
        "Background" : triangular(2,4,6);
        "Spell Check" : triangular(6,8,10);
        "Check Grammer" : triangular(7,9,11);
        "Speech" : triangular(1,3,5);
        "Mail Merge" : triangular(13,15,17);
        "Macro" : triangular(10,12,14);
        "Set Options" : triangular(1,3,5);
        "Insert Table" : triangular(3,5,7);
        "Delete Table" : triangular(2,4,6);
        "Table Format" : triangular(5,7,9);
        "Sort" : triangular(4,6,8);
        "Import Data" : triangular(7,9,11);
        "Help" : triangular(1,2,4);
        "Search" : deterministic(1);
}
```

The Design Cost is given as:

```
DesignCost = decision-subset(+)("Next Release"){

        "NewFile": triangular(15,17,19);

```

```
"OpenFile" : triangular(18, 20,22);
"CloseFile" : triangular(3, 5,7);
"SaveFile" : triangular(15,17,19);
"Save As Different File" : triangular(12, 14, 16);
"SearchFile" : triangular(16,18, 20);
"ProtectFile" : triangular(5,7,9);
"PrintPreview" : triangular(11,13,15);
"Print File" : triangular(8,10, 12);
"Send To" : triangular(7, 9, 11);
"Set Properties" : triangular(2,4,6);
"Exit" : triangular(1,2,4);
"Undo Task" : triangular(3,5,7);
"Redo Task" : triangular(4,7,8);
"Cut" : triangular(4,7,8);
"Copy" : triangular(1,3,5);
"Paste" : triangular(2,4, 6);
"Paste Special" : triangular (8,10,12);
"Go To" : triangular(3,5,7);
"Find" : triangular(4,5,7);
"Replace" : triangular(2,4,6);
"Select All" : triangular(2,4,6);
"Default" : triangular(1,3,5);
"Print Layout" : triangular(1,3,5);
"Web Layout" : triangular(1,3,5);
"Zoom" : triangular(4,7,8);
"Header Footer" : triangular(1,3,5);
"Page Numbers" : triangular(1,3, 5);
"Date Time" : triangular(1,3, 5);
"Symbol" : triangular(1,3, 5);
"Bookmark" : deterministic(1);
"Hyper Link" : deterministic(1);
"Font" : triangular(1,4,5);
"Paragraph" : triangular(3,5,7);
"Bullets Numbering" : triangular(2,5,6);
"Change Case" : triangular(2,5,6);
"Background" : triangular(2,5,6);
"Spell Check" : triangular(6,9,10);
```

```
        ”Speech” : **triangular**(1,4,5);
        ”Mail Merge” : **triangular**(13,14,17);
        ”Macro” : **triangular**(10,13,14);
        ”Set Options” : **triangular**(1,4,5);
        ”Insert Table” : **triangular**(3,6,7);
        ”Delete Table” : **triangular**(2,5,6);
        ”Table Format” : **triangular**(5,6,9);
        ”Sort” : **triangular**(4,5,8);
        ”Import Data” : **triangular**(7,8,11);
        ”Help” : **triangular**(1,3,4);
        ”Search” : **deterministic**(2);
}
```

The Developement Cost is given as:

```
DevCost = **decision**-subset (+)(”Next Release”){
        ”NewFile”: **triangular**(19,22,25);
        ”OpenFile” : **triangular**(22,25,27);
        ”CloseFile” : deterministic (1);
        ”SaveFile” : **triangular**(17,18,19);
        ”Save As Different File” : **triangular**(17,18,19);
        ”SearchFile” : **triangular**(20,25, 27);
        ”ProtectFile” : **triangular**(2,4,6);
        ”PrintPreview” : **triangular**(11,13,15);
        ”Print File” : **triangular**(8,10, 12);
        ”Send To” : **triangular**(7, 8, 11);
        ”Set Properties” : **triangular**(4,6,8);
        ”Exit” : **triangular**(2,4,6);
```

```
”Undo Task” : triangular(7,8,10);
”Redo Task” : triangular(7,8,10);
”Cut” : triangular(7,8,10);
”Copy” : triangular(1,4,5);
”Paste” : triangular(4, 6,8);
”Paste Special” : triangular(8,10,12);
”Go To” : triangular(3,6,7);
”Find” : triangular(4,5,7);
”Replace” : triangular(2,4,6);
”Select All” : triangular(2,4,6);
”Default” : triangular(1,3,5);
”Print Layout” : triangular(1,3,5);
”Web Layout” : triangular(1,3,5);
”Zoom” : triangular(4,5,8);
”Header Footer” : triangular(1,3,5);
”Page Numbers” : triangular(1,3, 5);
”Date Time” : triangular(1,3, 5);
”Symbol” : triangular(1,3, 5);
”Bookmark” : deterministic(2);
”Hyper Link” : deterministic(2);
”Font” : triangular(1,4,5);
”Paragraph” : triangular(3,6,7);
”Bullets Numbering” : triangular(2,5,6);
”Change Case” : triangular(2,5,6);
”Background” : triangular(2,5,6);
”Spell Check” : triangular(9,10,12);
”Check Grammer” : triangular(10,11,13);
”Speech” : triangular(5,6,8);
”Mail Merge” : triangular(10, 13,14);
”Macro” : triangular(8,10,13);
”Set Options” : triangular(1,2,5);
”Insert Table” : triangular(3,4,7);
”Delete Table” : triangular(2,3,6);
”Table Format” : triangular(5,6,9);
”Sort” : triangular(4,7,8);
”Import Data” : triangular(7,8,11);
”Help” : triangular(1,3,4);
```

We model the Testing Cost as:

```
TestingCost = decision-subset(+)("Next Release"){
        "NewFile": triangular(10,12,15);
        "OpenFile" : triangular(12,13,17);
        "CloseFile" : deterministic(2);
        "SaveFile" : triangular(11,13,15);
        "Save As Different File" : triangular(15, 17,18);
        "SearchFile" : triangular(16,18, 20);
        "ProtectFile" : triangular(2,3,6);
        "PrintPreview" : triangular(9, 11,13);
        "Print File" : triangular(8,10, 12);
        "Send To" : triangular(7, 8, 11);
        "Set Properties" : triangular(2, 4,6);
        "Exit" : triangular(2,3,6);
        "Undo Task" : triangular(3,5,10);
        "Redo Task" : triangular(2,4,8);
        "Cut" : triangular(3,4, 7);
        "Copy" : triangular(1,3,5);
        "Paste" : triangular(4, 6,8);
        "Paste Special" : triangular(8,11,12);
        "Go To" : triangular(3,5,7);
        "Find" : triangular(2,3,7);
        "Replace" : triangular(2,3,6);
        "Select All" : triangular(2,3,6);
        "Default" : triangular(1,2,5);
        "Print Layout" : triangular (1,3,5);
        "Web Layout" : triangular(1,3,5);
        "Zoom" : triangular(4,7,8);
        "Header Footer" : deterministic(1);
        "Page Numbers" : deterministic(1);
        "Date Time" : deterministic(1);
        "Symbol" : deterministic(1);
        "Bookmark" : deterministic(2);
        "Hyper Link" : deterministic(1);
        "Font" : triangular(1,2,5);
        "Paragraph" : triangular(3,6,7);
        "Bullets Numbering" : triangular(2,5,6);
        "Change Case" : triangular(2,5,6);
        "Background" : triangular(2,5,6);
        "Spell Check" : triangular(8,9,10);
        "Check Grammer" : triangular(6,8, 10);
        "Speech" : triangular (2,4, 5);
        "Mail Merge" : triangular(10, 13,14);
        "Macro" : triangular(8,11,13);
        "Set Options" : triangular(1,4,5);
        "Insert Table" : triangular(3,4,7);
        "Delete Table" : triangular(2,3,6);
        "Table Format" : triangular(5,6,9);
        "Sort" : triangular(4,5,8);
        "Import Data" : triangular(7,8,11);
        "Help" : triangular(1,3,4);
        "Search" : deterministic(2);
}
```

We model the constraint relationships between the requirements as below:

```
Constraint "Next Release" : "OpenFile" couples"Next Release" : "SaveFile";
Constraint "Next Release" : "Save As Different File" couples"Next Release" : "SaveFile";
Constraint "Next Release" : "Cut" couples"Next Release" : "Paste";
Constraint "Next Release" : "Cut" couples"Next Release" : "Paste Special";
Constraint "Next Release" : "Header Footer" couples"Next Release" : "Page Numbers";
Constraint "Next Release" : "Insert Table" couples"Next Release" : "Table Format";
Constraint "Next Release" : "Insert Table" couples"Next Release" : "Delete Table";
Constraint "Next Release" : "Help" couples"Next Release" : "Search";
```

Constraint ”Next Release” : ”OpenFile” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”Save As Different File” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”SearchFile” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”ProtectFile” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”PrintPreview” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”Print File” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”Send To” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”Set Properties” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”Exit” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”Print Layout” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”Web Layout” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”Zoom” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”Header Footer” **requires**”Next Release” : ”NewFile”;

Constraint ”Next Release” : ”Spell Check” **requires**”Next Release” : ”NewFile”;

Constraint "Next Release" : "Check Grammer" **requires**"Next Release" : "NewFile";

Constraint "Next Release" : "CloseFile" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "SearchFile" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "ProtectFile" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "PrintPreview" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Print File" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Send To" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Set Properties" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Go To" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Find" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Print Layout" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Web Layout" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Zoom" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Header Footer" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Page Numbersr" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Date Time" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Symbol" **requires**"Next Release" : "OpenFile";

Constraint "Next Release" : "Exit" **requires**"Next Release" : "CloseFile";

Constraint "Next Release" : "Exit" **requires**"Next Release" : "SaveFile";

Constraint ”Next Release” : ”Exit” **requires**”Next Release” : ”Save As Different File”;

Constraint ”Next Release” : ”Exit” **requires**”Next Release” : ”SearchFile”;

Constraint ”Next Release” : ”Exit” **requires**”Next Release” : ”ProtectFile”;

Constraint ”Next Release” : ”Exit” **requires**”Next Release” : ”PrintPreview”;

Constraint ”Next Release” : ”Exit” **requires**”Next Release” : ”Print File”;

Constraint ”Next Release” : ”Exit” **requires**”Next Release” : ”Send To”;

Constraint ”Next Release” : ”Send To” **requires**”Next Release” : ”ProtectFile”;

Constraint ”Next Release” : ”Redo Task” **requires**”Next Release” : ”Undo Task”;

Constraint ”Next Release” : ”Paste” **requires**”Next Release” : ”Go To”;

Constraint ”Next Release” : ”Replace” **requires**”Next Release” : ”Go To”;

Constraint ”Next Release” : ”Print File” **requires**”Next Release” : ”Print Layout”;

Constraint ”Next Release” : ”Print Layout” **requires**”Next Release” : ”Header Footer”;

Constraint ”Next Release” : ”Web Layout” **requires**”Next Release” : ”Header Footer”;

Constraint ”Next Release” : ”Bookmark” **requires**”Next Release” : ”Page Numbers”;

Constraint ”Next Release” : ”Hyper Link” **requires**”Next Release” : ”Page Numbers”;

Constraint ”Next Release” : ”Bookmark” **requires**”Next Release” : ”Date Time”;

Constraint ”Next Release” : ”Hyper Link” **requires**”Next Release” : ”Date Time”;

Constraint ”Next Release” : ”Paragraph” **requires**”Next Release” : ”Font”;

```
Constraint "Next Release" : "Bullets Numbering" requires "Next Release" : "Font";
Constraint "Next Release" : "Change Case" requires"Next Release" : "Font";
Constraint "Next Release" : "Background" requires"Next Release" : "Font";
Constraint "Next Release" : "Background" requires"Next Release" : "Paragraph";
Constraint "Next Release" : "Background" requires"Next Release" : "Bullets Numbering";
Constraint "Next Release" : "Background" requires"Next Release" : "Change Case";
Constraint "Next Release" : "Check Grammer" requires"Next Release" : "Spell Check";
Constraint "Next Release" : "Speech" requires"Next Release" : "Check Grammer";
Constraint "Next Release" : "Spell Check" requires"Next Release" : "Set Options";
Constraint "Next Release" : "Check Grammer" requires"Next Release" : "Set Options";
Constraint "Next Release" : "Speech" requires"Next Release" : "Set Options";
Constraint "Next Release" : "Mail Merge" requires"Next Release" : "Set Options";
Constraint "Next Release" : "Macro" requires"Next Release" : "Set Options";
```

Analysis Result

Optimisation Analysis

The RADAR analysis of the RADAR models developed for the Microsoft Word Processor presented in Fig. G.1 show that a solution is found to be optimal out of a total of 2^{50} solutions. The solution include the following features: NewFile, OpenFile, CloseFile, SaveFile, Save As Different File, ProtectFile, PrintPreview, Set Properties, Cut, Paste, Paste Special, Go To, Find, Select All, Zoom, Header Footer, Symbol, Font, Paragraph, Bullets Numbering, Change Case, Spell Check, Check Grammar, Set Options, Insert Table, Delete Table, Table Format and Sort.

The EVTPI is £6.14m and EVPPI for all model parameters is approximately equal to zero. This means that none of the model parameters is worth further data collection or analysis.

Optimisation Analysis

Objective:	**Max** ENB
Objective:	**Min** ProjectRisk
Solution Space:	2^{50}
Optimisation Approach:	$1+\lambda$
Algorithm Name:	NSGAII
Population Size:	100
Crossover Probability:	0.8
Mutation Probability:	0.1
Nbr. Fitness Evaluations:	50000
Shortlisted:	1
Nbr. Decisions:	1
Nbr. Variables:	29
Runtime(s) :	46

	Feature in the Next Release
1	NewFile
2	OpenFile
3	CloseFile
4	SaveFile
5	Save As Different File
6	ProtectFile
7	PrintPreview
8	Set Properties
9	Cut
10	Paste
11	Paste Special
12	Go To
13	Find
14	Select All
15	Zoom
16	Header Footer
17	Symbol
18	Font
19	Paragraph
20	Bullets Numbering
21	Change Case
22	Spell Check
23	Check Grammer
24	Set Options
25	Insert Table
26	Delete Table
27	Table Format
28	Sort
ENB	409
ProjectRisk	0

FIGURE G.1: Optimisation Analysis and Information Value analysis results of requirements subset selection problem for the Microsoft Word Processor.

Bibliography

[1] User Requirements Notation (URN). https://www.itu.int/rec/T-REC-Z.151-201210-I/en, 2012. [Online; accessed 20-Sepetember-2016].

[2] jUCMNav. http://jucmnav.softwareengineering.ca/ucm/bin/view/ProjetSEG/WebHome, 2016. [Online; accessed 06-October-2016].

[3] Amy Affleck, Aneesh Krishna, and Narasimaha R Achuthan. Optimal selection of operationalizations for non-functional requirements. In *Proceedings of the Ninth Asia-Pacific Conference on Conceptual Modelling-Volume 143*, pages 69–78. Australian Computer Society, Inc., 2013.

[4] Amy Affleck, Aneesh Krishna, and Narasimaha R Achuthan. Non-functional requirements framework: A mathematical programming approach. *The Computer Journal*, 58(5):1122–1139, 2015.

[5] Ahmed Al-Emran and Dietmar Pfahl. Operational planning, re-planning and risk analysis for software releases. In *Product-Focused Software Process Improvement*, pages 315–329. Springer, 2007.

[6] Ahmed Al-Emran, Puneet Kapur, Dietmar Pfahl, and Guenther Ruhe. Studying the impact of uncertainty in operational release planning–an integrated method and its initial evaluation. *Information and Software Technology*, 52(4):446–461, 2010.

[7] Ahmed Al-Emran, Dietmar Pfahl, and Guenther Ruhe. Decision support for product release planning based on robustness analysis. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 157–166. IEEE, 2010.

[8] Tariq Al-Naeem, Ian Gorton, Muhammed Ali Babar, Fethi Rabhi, and Boualem Benatallah. A quality-driven systematic approach for architecting distributed software applications. In *Proceedings of the 27th international conference on Software engineering*, pages 244–253. ACM, 2005.

[9] Thamer AlBourae, Guenther Ruhe, and Mahmood Moussavi. Lightweight replanning of software product releases. In *Software Product Management, 2006. IWSPM'06. International Workshop on*, pages 27–34. IEEE, 2006.

[10] Antonio J Alencar, Carlos AS Franco, Eber A Schmitz, and Alexandre L Correa. A statistical approach for the maximization of the financial benefits yielded by a large set of mmfs and aes. *Computing and Informatics*, 32(6):1147–1169, 2014.

[11] Beatrice Alenljung. Envisioning a future decision support system for requirements engineering: a holistic and human-centred perspective. 2008.

[12] Ian Alexander. A taxonomy of stakeholders, human roles in system development. *Issues and Trends in Technology and Human Interaction*, pages 25–71, 2006.

[13] Rajeev Alur and Thomas A Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.

[14] Daniel Amyot, Sepideh Ghanavati, Jennifer Horkoff, Gunter Mussbacher, Liam Peyton, and Eric Yu. Evaluating goal models within the goal-oriented requirement language. *International Journal of Intelligent Systems*, 25(8):841–877, 2010.

[15] Debra Anderson, Thane Frivold, and Alfonso Valdes. Next-generation intrusion detection expert system (nides): A summary. 1995.

[16] Suzana Andova, Holger Hermanns, and Joost-Pieter Katoen. Discrete-time rewards model-checked. In *International Conference on Formal Modeling and Analysis of Timed Systems*, pages 88–104. Springer, 2003.

[17] Michal Antkiewicz and Krzysztof Czarnecki. Featureplugin: feature modeling plug-in for eclipse. In *Proceedings of the 2004 OOPSLA workshop on eclipse technology eXchange*, pages 67–72. ACM, 2004.

[18] Michal Antkiewicz, Kacper Bak, Alexandr Murashkin, Rafael Olaechea, Jia Hui Jimmy Liang, and Krzysztof Czarnecki. Clafer tools for product line engineering. In *Proceedings of the 17th International Software Product Line Conference co-located workshops*, pages 130–135. ACM, 2013.

[19] Ana I Anton. Goal identification and refinement in the specification of software-based information systems. 1997.

[20] Annie I Anton. Goal-based requirements analysis. In *Requirements Engineering, 1996., Proceedings of the Second International Conference on*, pages 136–144. IEEE, 1996.

[21] Annie I Antón and Colin Potts. The use of goals to surface requirements for evolving systems. In *Software Engineering, 1998. Proceedings of the 1998 International Conference on*, pages 157–166. IEEE, 1998.

[22] Annie I Antón, W Michael McCracken, and Colin Potts. Goal decomposition and scenario analysis in business process reengineering. In *International Conference on Advanced Information Systems Engineering*, pages 94–104. Springer, 1994.

[23] Annie I Anton, Ryan A Carter, Aldo Dagnino, John H Dempster, and Devon F Siege. Deriving goals from a use-case based requirements specification. *Requirements Engineering*, 6(1):63–73, 2001.

[24] Annie I Antón, Julia Brande Earp, and Angela Reese. Analyzing website privacy requirements using a privacy goal taxonomy. In *Requirements Engineering, 2002. Proceedings. IEEE Joint International Conference on*, pages 23–31. IEEE, 2002.

[25] Giuliano Antoniol, Massimiliano Di Penta, and Mark Harman. A robust search-based approach to project management in the presence of abandonment, rework, error and uncertainty. In *Software Metrics, 2004. Proceedings. 10th International Symposium on*, pages 172–183. IEEE, 2004.

[26] Jorge Aranda and Steve Easterbrook. Anchoring and adjustment in software estimation. In *10th European Software Engineering Conference and 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering (ES-EC/FSE 2013)*, pages 346–355. ACM, 2005.

[27] Andrea Arcuri and Lionel Briand. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Software Engineering (ICSE), 2011 33rd International Conference on*, pages 1–10. IEEE, 2011.

[28] Aybüke Aurum and Claes Wohlin. The fundamental nature of requirements engineering activities as a decision-making process. *Information and Software Technology*, 45(14):945–954, 2003.

[29] Muhammad Ali Babar, Liming Zhu, and Ross Jeffery. A framework for classifying and comparing software architecture evaluation methods. In *Software Engineering Conference, 2004. Proceedings. 2004 Australian*, pages 309–318. IEEE, 2004.

[30] Anthony J. Bagnall, Victor J. Rayward-Smith, and Ian M Whittley. The next release problem. *Information and software technology*, 43(14):883–890, 2001.

[31] Paul Baker, Mark Harman, Kathleen Steinhofel, and Alexandros Skaliotis. Search based approaches to component selection and prioritization for the next release problem. In *Software Maintenance, 2006. ICSM'06. 22nd IEEE International Conference on*, pages 176–185. IEEE, 2006.

[32] Len Bass. *Software architecture in practice.* Pearson Education India, 2007.

[33] Richard Ernest Bellman and Stuart E Dreyfus. Applied dynamic programming. 1962.

[34] Nelly Bencomo. Quantun: Quantification of uncertainty for the reassessment of requirements. In *Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, pages 236–240. IEEE, 2015.

[35] Nelly Bencomo, Jon Whittle, Pete Sawyer, Anthony Finkelstein, and Emmanuel Letier. Requirements reflection: requirements as runtime entities. In *Software Engineering, 2010 ACM/IEEE 32nd International Conference on*, volume 2, pages 199–202. IEEE, 2010.

[36] Barry Boehm, Prasanta Bose, Ellis Horowitz, and Ming June Lee. Software requirements negotiation and renegotiation aids: A theory-w based spiral approach. In *Software Engineering, 1995. ICSE 1995. 17th International Conference on*, pages 243–243. IEEE, 1995.

[37] Richard J Bolton and David J Hand. Statistical fraud detection: A review. *Statistical science*, pages 235–249, 2002.

[38] Ronald J Brachman and Hector J Levesque. *Readings in knowledge representation.* Morgan Kaufmann Publishers Inc., 1985.

[39] Klaus Briess, Herbert Jahn, Eckehard Lorenz, Dieter Oertel, Wolfgang Skrbek, and Boris Zhukov. Fire recognition potential of the bi-spectral infrared detection (bird) satellite. *International Journal of Remote Sensing*, 24(4):865–872, 2003.

[40] Janis Bubenko, Colette Rolland, Pericles Loucopoulos, and Valeria DeAntonellis. Facilitating fuzzy to formal requirements modelling. In *Requirements Engineering, 1994., Proceedings of the First International Conference on*, pages 154–157. IEEE, 1994.

[41] Saheed A Busari. Towards search-based modelling and analysis of requirements and architecture decisions. In *Proceedings of the 32nd IEEE/ACM International Conference on Automated Software Engineering*, pages 1026–1029. IEEE Press, 2017.

[42] Saheed A Busari and Emmanuel Letier. Radar: A lightweight tool for requirements and architecture decision analysis. In *Proceedings of the 39th International Conference on Software Engineering*, pages 552–562. IEEE Press, 2017.

[43] Saheed A Busari and Emmanuel Letier. Scalability analysis of the radar decision support tool. *arXiv preprint arXiv:1702.02977*, 2017.

[44] Antoine Cailliau and Axel van Lamsweerde. Handling knowledge uncertainty in risk-based requirements engineering. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pages 106–115. IEEE, 2015.

[45] Murray Cantor. Calculating and improving roi in software and system programs. *Communications of the ACM*, 54(9):121–130, 2011.

[46] Pär Carlshamre, Kristian Sandahl, Mikael Lindvall, Björn Regnell, and J Natt och Dag. An industrial survey of requirements interdependencies in software product release planning. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 84–91. IEEE, 2001.

[47] Jaelson Castro, Manuel Kolp, and John Mylopoulos. Towards requirements-driven information systems engineering: the tropos project. *Information systems*, 27(6): 365–389, 2002.

[48] Tristan Caulfield and David Pym. Improving security policy decisions with models. *IEEE Security & Privacy*, 13(5):34–41, 2015.

[49] Tristan Caulfield and David Pym. Modelling and simulating systems security policy. In *SIMUTOOLS 2015-8th EAI International Conference on Simulation Tools and Techniques*. ICST, 2015.

[50] Tristan Caulfield and David Pym. Modelling and simulating systems security policy. In *8th International Conference on Simulation Tools and Techniques*, pages 9–18. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2015.

[51] Tristan Caulfield and David Pym. Improving security policy decisions with models. *IEEE Security and Privacy Magazine*, 13(5):34–41, 2015.

[52] Vaclav Cechticky, Alessandro Pasetti, O Rohlik, and Walter Schaufelberger. Xml-based feature modelling. In *International Conference on Software Reuse*, pages 101–114. Springer, 2004.

[53] Chieng-Yi Chang. Dynamic programming as applied to feature subset selection in a pattern recognition system. In *Proceedings of the ACM annual conference-Volume 1*, pages 94–103. ACM, 1972.

[54] Adhitya Chittur. Model generation for an intrusion detection system using genetic algorithms. *High School Honors Thesis, Ossining High School. In cooperation with Columbia Univ*, 2001.

[55] Lawrence Chung, Daniel Gross, and Eric Yu. Architectural design to meet stakeholder requirements. In *Software architecture*, pages 545–564. Springer, 1999.

[56] Lawrence Chung, Brian A Nixon, Eric Yu, and John Mylopoulos. *Non-functional requirements in software engineering*, volume 5. Springer Science & Business Media, 2012.

[57] Lawrence Chung, Tom Hill, Owolabi Legunsen, Zhenzhou Sun, Adip Dsouza, and Sam Supakkul. A goal-oriented simulation approach for obtaining good private

cloud-based system architectures. *Journal of Systems and Software*, 86(9):2242–2262, 2013.

[58] John Clarke, Jose Javier Dolado, Mark Harman, Rob Hierons, Bryan Jones, Mary Lumkin, Brian Mitchell, Spiros Mancoridis, Kearton Rees, Marc Roper, et al. Reformulating software engineering as a search problem. *IEE Proceedings-software*, 150(3):161–175, 2003.

[59] Carlos A Coello Coello, Clarisse Dhaenens, and Laetitia Jourdan. Multi-objective combinatorial optimization: Problematic and context. In *Advances in multi-objective nature inspired computing*, pages 1–21. Springer, 2010.

[60] Vittorio Cortellessa, Fabrizio Marinelli, and Pasqualina Potena. Automated selection of software components based on cost/reliability tradeoff. In *Software Architecture*, pages 66–81. Springer, 2006.

[61] PRISM Cost and Reward. SysML Specification. http://www.prismmodelchecker.org/manual/ThePRISMLanguage/CostsAndRewards, 2008. [Online; accessed 28-October-2016].

[62] Joseph Czyzyk, Michael P Mesnier, and Jorge J Moré. The neos server. *IEEE Computational Science and Engineering*, 5(3):68–75, 1998.

[63] Eduardo Maciel da Cunha Mattos, Marcelo Vieira, Eber Assis Schmitz, and Antonio Juarez Alencar. Applying game theory to the incremental funding method in software projects. *Journal of Software*, 9(6):1435–1443, 2014.

[64] Jacek Dabrowski. Towards an adaptive framework for goal-oriented strategic decision-making. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 538–543. IEEE, 2017.

[65] Anne Dardenne, Axel Van Lamsweerde, and Stephen Fickas. Goal-directed requirements acquisition. *Science of computer programming*, 20(1):3–50, 1993.

[66] Robert Darimont and Axel Van Lamsweerde. Formal refinement patterns for goal-driven requirements elaboration. In *ACM SIGSOFT Software Engineering Notes*, volume 21, pages 179–190. ACM, 1996.

[67] AL de Cerqueira Leite Duboc. *A framework for the characterization and analysis of software systems scalability.* PhD thesis, UCL (University College London), 2010.

[68] Leonardo De Moura and Nikolaj Bjørner. Z3: An efficient smt solver. In *International conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 337–340. Springer, 2008.

[69] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *Evolutionary Computation, IEEE Transactions on*, 6(2):182–197, 2002.

[70] Mark Denne and Jane Cleland-Huang. *Software by numbers: Low-risk, high-return development.* Prentice Hall Professional, 2003.

[71] Mark Denne and Jane Cleland-Huang. The incremental funding method: Data-driven software development. *IEEE software*, 21(3):39–47, 2004.

[72] Clarisse Dhaenens, Julien Lemesre, and El-Ghazali Talbi. K-ppm: A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200(1):45–53, 2010.

[73] L. Duboc, E. Letier, D. S. Rosenblum, and T. Wicks. A case study in eliciting scalability requirements. In *16th IEEE International Requirements Engineering Conference (RE 2008)*, pages 247–252, 2008.

[74] Leticia Duboc, Emmanuel Letier, David S Rosenblum, and Tony Wicks. A case study in eliciting scalability requirements. In *International Requirements Engineering, 2008. RE'08. 16th IEEE*, pages 247–252. IEEE, 2008.

[75] Leticia Duboc, Emmanuel Letier, and David S. Rosenblum. Systematic elaboration of scalability requirements through goal-obstacle analysis. *IEEE Transactions on Software Engineering*, 39(1):119–140, 2013. ISSN 0098-5589.

[76] Juan J Durillo, Yuanyuan Zhang, Enrique Alba, Mark Harman, and Antonio J Nebro. A study of the bi-objective next release problem. *Empirical Software Engineering*, 16(1):29–60, 2011.

[77] Bruno Dutertre and Leonardo De Moura. The yices smt solver. *Tool paper at http://yices. csl. sri. com/tool-paper. pdf*, 2(2):1–2, 2006.

[78] RW Eglese. Simulated annealing: a tool for operational research. *European journal of operational research*, 46(3):271–281, 1990.

[79] Nasser A El-Sherbeny. Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods. *Journal of King Saud University-Science*, 22(3):123–131, 2010.

[80] Renee Elio, Jim Hoover, Ioanis Nikolaidis, Mohammad Salavatipour, Lorna Stewart, and Ken Wong. About computing science research methodology, 2011.

[81] Richard Ellis-Braithwaite, Russell Lock, Ray Dawson, and Badr Haque. Modelling the strategic alignment of software requirements using goal graphs. *arXiv preprint arXiv:1211.6258*, 2012.

[82] John Ellson, Emden Gansner, Lefteris Koutsofios, Stephen C North, and Gordon Woodhull. Graphvizopen source graph drawing tools. In *International Symposium on Graph Drawing*, pages 483–484. Springer, 2001.

[83] Naeem Esfahani, Ehsan Kouroshfar, and Sam Malek. Taming uncertainty in self-adaptive software. In *Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering*, pages 234–244. ACM, 2011.

[84] Naeem Esfahani, Salim Malek, and Kaveh Razavi. Guidearch: guiding the exploration of architectural solution space under uncertainty. In *Software Engineering (ICSE), 2013 35th International Conference on*, pages 43–52. IEEE, 2013.

[85] Naeem Esfahani, Sam Malek, and Kaveh Razavi. Guidearch: guiding the exploration of architectural solution space under uncertainty. In *35th International Conference on Software Engineering (ICSE 2013)*, pages 43–52. IEEE, 2013.

[86] Davide Falessi, Giovanni Cantone, Rick Kazman, and Philippe Kruchten. Decision-making techniques for software architecture design: A comparative survey. *ACM Computing Surveys (CSUR)*, 43(4):33, 2011.

[87] Norman Fenton and Martin Neil. *Risk assessment and decision analysis with Bayesian networks*. Crc Press, 2012.

[88] Filomena Ferrucci, Mark Harman, Jian Ren, and Federica Sarro. Not going to take this anymore: multi-objective overtime planning for software engineering

projects. In *Proceedings of the 2013 International Conference on Software Engineering*, pages 462–471. IEEE Press, 2013.

[89] UK Financial Fraud Action. Plastic Card Frauds. https://www.financialfraudaction.org.uk/wp-content/uploads/2016/09/2015-h1-fraud-figures-release-final.pdf. [Online; accessed 28-November-2015].

[90] Anthony Finkelstein and John Dowell. A comedy of errors: the london ambulance service case study. In *Proceedings of the 8th International Workshop on Software Specification and Design*, page 2. IEEE Computer Society, 1996.

[91] Anthony Finkelstein, Mark Harman, S Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. fairness analysis in requirements assignments. In *International Requirements Engineering, 2008. RE'08. 16th IEEE*, pages 115–124. IEEE, 2008.

[92] Anthony Finkelstein, Mark Harman, S Afshin Mansouri, Jian Ren, and Yuanyuan Zhang. A search based approach to fairness analysis in requirement assignments to aid negotiation, mediation and decision making. *Requirements Engineering*, 14 (4):231–245, 2009.

[93] Carlos M Fonseca, Peter J Fleming, et al. Genetic algorithms for multiobjective optimization: Formulation discussion and generalization. In *ICGA*, volume 93, pages 416–423, 1993.

[94] Martin Fowler. *Domain-specific languages*. Pearson Education, 2010.

[95] Johan Fredriksson, Kristian Sandström, and Mikael Åkerholm. Optimizing resource usage in component-based real-time systems. In *Component-Based Software Engineering*, pages 49–65. Springer, 2005.

[96] Sanford Friedenthal, Alan Moore, and Rick Steiner. *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[97] Emden Gansner, Eleftherios Koutsofios, and Stephen North. Drawing graphs with dot, 2006.

[98] Jesus Garcia-Galan, Pablo Trinidad, Omer F Rana, and Antonio Ruiz-Cortes. Automated configuration support for infrastructure migration to the cloud. *Future Generation Computer Systems*, 55:200–212, 2016.

[99] Thomas D Garvey and Teresa F Lunt. Model-based intrusion detection. In *Proceedings of the 14th national computer security conference*, volume 17, 1991.

[100] Simos Gerasimou, Giordano Tamburrelli, and Radu Calinescu. Search-based synthesis of probabilistic models for quality-of-service software engineering. In *30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015)*, pages 319–330. IEEE, 2015.

[101] Simos Gerasimou, Giordano Tamburrelli, and Radu Calinescu. Search-based synthesis of probabilistic models for quality-of-service software engineering. In *30th IEEE/ACM International Conference on Automated Software Engineering (ASE 2015)*. York, 2015.

[102] Sushmito Ghosh and Douglas L Reilly. Credit card fraud detection with a neural-network. In *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*, volume 3, pages 621–630. IEEE, 1994.

[103] Paolo Giorgini, John Mylopoulos, Eleonora Nicchiarelli, and Roberto Sebastiani. Reasoning with goal models. In *International Conference on Conceptual Modeling*, pages 167–181. Springer, 2002.

[104] Martin Glinz. On non-functional requirements. In *Requirements Engineering Conference, 2007. RE'07. 15th IEEE International*, pages 21–26. IEEE, 2007.

[105] Fred Glover. Tabu search-part i. *ORSA Journal on computing*, 1(3):190–206, 1989.

[106] Fred Glover and Manuel Laguna. *Tabu Search**. Springer, 2013.

[107] Des Greer and Günther Ruhe. Software release planning: an evolutionary and iterative approach. *Information and Software Technology*, 46(4):243–253, 2004.

[108] Daniel Gross and Eric Yu. Evolving system architecture to meet changing business goals: an agent and goal-oriented approach. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 316–317. IEEE, 2001.

[109] Lars Grunske. Identifying good architectural design alternatives with multi-objective optimization strategies. In *Proceedings of the 28th international conference on Software engineering*, pages 849–852. ACM, 2006.

[110] Jianmei Guo, Jules White, Guangxin Wang, Jian Li, and Yinglin Wang. A genetic algorithm for optimized feature selection with resource constraints in software product lines. *Journal of Systems and Software*, 84(12):2208–2221, 2011.

[111] Jianmei Guo, Edward Zulkoski, Rafael Olaechea, Derek Rayside, Krzysztof Czarnecki, Sven Apel, and Joanne M. Atlee. Scaling exact multi-objective combinatorial optimization by parallelization. In *29th ACM/IEEE International Conference on Automated Software Engineering (ASE 2014)*, pages 409–420, 2014.

[112] Jianmei Guo, Edward Zulkoski, Rafael Olaechea, Derek Rayside, Krzysztof Czarnecki, Sven Apel, and Joanne M Atlee. Scaling exact multi-objective combinatorial optimization by parallelization. In *Proceedings of the 29th ACM/IEEE international conference on Automated software engineering*, pages 409–420. ACM, 2014.

[113] John Hammersley. *Monte carlo methods.* Springer Science & Business Media, 2013.

[114] Jiawei Han, Jian Pei, and Micheline Kamber. *Data mining: concepts and techniques.* Elsevier, 2011.

[115] D. J. Hand, C. Whitrow, N. M. Adams, P. Juszczak, and D. Weston. Performance criteria for plastic card fraud detection tools. *The Journal of the Operational Research Society*, 59(7):956–962, 2008.

[116] DJ Hand, C Whitrow, NM Adams, P Juszczak, and D Weston. Performance criteria for plastic card fraud detection tools. *Journal of the Operational Research Society*, pages 956–962, 2008.

[117] Mark Harman. The current state and future of search based software engineering. In *2007 Future of Software Engineering*, pages 342–357. IEEE Computer Society, 2007.

[118] Mark Harman. Sbse: introduction and motivation. In *Search Based Software Engineering*, pages 16–16. Springer, 2011.

[119] Mark Harman and Bryan F Jones. Search-based software engineering. *Information and Software Technology*, 43(14):833–839, 2001.

[120] Mark Harman and Laurence Tratt. Pareto optimal search based refactoring at the design level. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1106–1113. ACM, 2007.

[121] Mark Harman, S Afshin Mansouri, and Yuanyuan Zhang. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. *Department of Computer Science, Kings College London, Tech. Rep. TR-09-03*, 2009.

[122] Mark Harman, Phil McMinn, Jerffeson Teixeira De Souza, and Shin Yoo. Search based software engineering: Techniques, taxonomy, tutorial. In *Empirical software engineering and verification*, pages 1–59. Springer, 2012.

[123] Mark Harman, Jens Krinke, Inmaculada Medina-Bulo, Francisco Palomo-Lozano, Jian Ren, and Shin Yoo. Exact scalable sensitivity analysis for the next release problem. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 23(2):19, 2014.

[124] Sara Hassan, Nelly Bencomo, and Rami Bahsoon. Minimizing nasty surprises with better informed decision-making in self-adaptive systems. In *Proceedings of the 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 134–144. IEEE Press, 2015.

[125] W. Heaven and E. Letier. Simulating and optimising design decisions in quantitative goal models. In *19th IEEE International Requirements Engineering Conference (RE 2011)*, pages 79–88. IEEE, 2011.

[126] William Heaven and Emmanuel Letier. Simulating and optimising design decisions in quantitative goal models. In *Requirements Engineering Conference (RE), 2011 19th IEEE International*, pages 79–88. IEEE, 2011.

[127] C. Henard, M. Papadakis, M. Harman, and Y. Le Traon. Combining multi-objective search and constraint solving for configuring large software product lines. In *37th IEEE International Conference on Software Engineering ICSE 2015)*, pages 517–528, 2015.

[128] Christopher Henard, Mike Papadakis, Mark Harman, and Yves Le Traon. Combining multi-objective search and constraint solving for configuring large software

product lines. In *Proceedings of the 37th IEEE/ACM International Conference on Software Engineering (ICSE 2015)*, 2015.

[129] Ann M Hickey and Alan M Davis. Requirements elicitation and elicitation technique selection: model for two knowledge-intensive software development processes. In *System Sciences, 2003. Proceedings of the 36th Annual Hawaii International Conference on*, pages 10–pp. IEEE, 2003.

[130] Robert M Hierons, Miqing Li, Xiaohui Liu, Sergio Segura, and Wei Zheng. Sip: Optimal product selection from feature models using many-objective evolutionary optimization. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 25(2):17, 2016.

[131] Andrew Hinton, Marta Kwiatkowska, Gethin Norman, and David Parker. Prism: A tool for automatic verification of probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 441–444. Springer, 2006.

[132] Jennifer Horkoff, Yijun Yu, and SK Eric. Openome: An open-source goal and agent-oriented model drawing and analysis tool. *iStar*, 766:154–156, 2011.

[133] R.A. Howard. Information value theory. *IEEE Transactions on Systems Science and Cybernetics*, 2(1):22–26, 1966.

[134] Ronald A Howard. *Readings on the principles and applications of decision analysis*, volume 1. Strategic Decisions Group, 1983.

[135] Douglas Hubbard. The IT measurement inversion. *CIO Enterprise Magazine*, 1999.

[136] D.W. Hubbard. *How to measure anything: Finding the value of intangibles in business*. Wiley, 2010.

[137] Bowen Hui, Sotirios Liaskos, and John Mylopoulos. Requirements analysis for customizable software: A goals-skills-preferences framework. In *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, pages 117–126. IEEE, 2003.

[138] Koral Ilgun, Richard A Kemmerer, and Phillip A Porras. State transition analysis: A rule-based intrusion detection approach. *IEEE transactions on software engineering*, 21(3):181–199, 1995.

[139] Daniel Jackson, H Estler, Derek Rayside, et al. The guided improvement algorithm for exact, general-purpose, many-objective combinatorial optimization. 2009.

[140] Stephan Jacobs. Introducing measurable quality requirements: a case study. In *Requirements Engineering, 1999. Proceedings. IEEE International Symposium on*, pages 172–179. IEEE, 1999.

[141] Daniel Kahneman. *Thinking, Fast and Slow.* Macmillan, 2011.

[142] Muhammad Rezaul Karim and Guenther Ruhe. Bi-objective genetic search for release planning in support of themes. In *International Symposium on Search Based Software Engineering*, pages 123–137. Springer, 2014.

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

[144] Joachim Karlsson, Stefan Olsson, and Kevin Ryan. Improved practical support for large-scale requirements prioritising. *Requirements Engineering*, 2(1):51–60, 1997.

[145] Christian Kastner, Thomas Thum, Gunter Saake, Janet Feigenspan, Thomas Leich, Fabian Wielgorz, and Sven Apel. Featureide: A tool framework for feature-oriented software development. In *Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on*, pages 611–614. IEEE, 2009.

[146] Rick Kazman, Gregory Abowd, Len Bass, and Paul Clements. Scenario-based analysis of software architecture. *IEEE software*, 13(6):47–55, 1996.

[147] Rick Kazman, Mark Klein, Mario Barbacci, Tom Longstaff, Howard Lipson, and Jeromy Carriere. The architecture tradeoff analysis method. In *Engineering of Complex Computer Systems, 1998. ICECCS'98. Proceedings. Fourth IEEE International Conference on*, pages 68–78. IEEE, 1998.

[148] Rick Kazman, Mark Klein, and Paul Clements. Atam: Method for architecture evaluation. Technical report, DTIC Document, 2000.

[149] Rick Kazman, Jai Asundi, and Mark Klein. Quantifying the costs and benefits of architectural decisions. In *23rd International Conference on Software Engineering (ICSE 2001)*, pages 297–306. IEEE Computer Society, 2001.

[150] Rick Kazman, Jai Asundi, and Mark Klein. Quantifying the costs and benefits of architectural decisions. In *Proceedings of the 23rd international conference on Software engineering*, pages 297–306. IEEE Computer Society, 2001.

[151] Rick Kazman, Jai Asundi, and Mark Klien. Making architecture design decisions: An economic approach. Technical Report CMU/SEI-2002-TR-035, Carnegie Mellon University. Software Engineering Institute, 2002.

[152] Taghi M Khoshgoftaar, Yi Liu, and Naeem Seliya. A multiobjective module-order model for software quality enhancement. *Evolutionary Computation, IEEE Transactions on*, 8(6):593–608, 2004.

[153] Alexander Knüppel, Thomas Thüm, Stephan Mennicke, Jens Meinicke, and Ina Schaefer. Is there a mismatch between real-world feature models and product-line research? In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 291–302. ACM, 2017.

[154] Warren L. G. Koontz, Patrenahalli M. Narendra, and Keinosuke Fukunaga. A branch and bound clustering algorithm. *IEEE Transactions on Computers*, (9): 908–915, 1975.

[155] Sameer Kumar and Promma Phrommathed. *Research methodology.* Springer, 2005.

[156] Marta Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In *International Conference on Computer Aided Verification (CAV 2011)*, pages 585–591. Springer Berlin Heidelberg, 2011.

[157] Kiran Lakhotia, Mark Harman, and Phil McMinn. A multi-objective approach to search-based test data generation. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1098–1105. ACM, 2007.

[158] Dries Langsweirdt, Nelis Boucké, and Yolande Berbers. Architecture-driven development of embedded systems with acol. In *ISORC Workshops*, pages 138–144, 2010.

[159] Alexei Lapouchnian. Goal-oriented requirements engineering: An overview of the current research. *University of Toronto*, page 32, 2005.

[160] Eugene L Lawler and David E Wood. Branch-and-bound methods: A survey. *Operations research*, 14(4):699–719, 1966.

[161] Julien Lemesre, Clarisse Dhaenens, and El-Ghazali Talbi. Parallel partitioning method (ppm): A new exact method to solve bi-objective problems. *Computers & operations research*, 34(8):2450–2462, 2007.

[162] Emmanuel Letier. *Reasoning about agents in goal-oriented requirements engineering.* PhD thesis, PhD thesis, Université catholique de Louvain, 2001.

[163] Emmanuel Letier and Axel Van Lamsweerde. Agent-based tactics for goal-oriented requirements elaboration. In *Proceedings of the 24th International Conference on Software Engineering*, pages 83–93. ACM, 2002.

[164] Emmanuel Letier and Axel Van Lamsweerde. Deriving operational software specifications from system goals. In *Proceedings of the 10th ACM SIGSOFT symposium on Foundations of software engineering*, pages 119–128. ACM, 2002.

[165] Emmanuel Letier and Axel Van Lamsweerde. Reasoning about partial goal satisfaction for requirements and design engineering. In *ACM SIGSOFT Software Engineering Notes*, volume 29, pages 53–62. ACM, 2004.

[166] Emmanuel Letier, David Stefan, and Earl T. Barr. Uncertainty, risk, and information value in software requirements and architecture. In *36th International Conference on Software Engineering (ICSE 2014)*, pages 883–894, 2014.

[167] Lingbo Li, Mark Harman, Emmanuel Letier, and Yuanyuan Zhang. Robust next release problem: handling uncertainty during optimization. In *Proceedings of the 2014 conference on Genetic and evolutionary computation*, pages 1247–1254. ACM, 2014.

[168] Lingbo Li, Mark Harman, Fan Wu, and Yuanyuan Zhang. The value of exact analysis in requirements selection. *IEEE Transactions on Software Engineering*, 43(6):580–596, 2017.

[169] Mian Li, Shapour Azarm, and Vikrant Aute. A multi-objective genetic algorithm for robust design optimization. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 771–778. ACM, 2005.

[170] Lin Liu, Eric Yu, and John Mylopoulos. Security and privacy requirements analysis within a social setting. In *Requirements Engineering Conference, 2003. Proceedings. 11th IEEE International*, pages 151–161. IEEE, 2003.

[171] Qin Ma and Sybren de Kinderen. Goal-based decision making. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 19–35. Springer, 2016.

[172] Sara Mahdavi-Hezavehi, Paris Avgeriou, and Danny Weyns. A classification framework of uncertainty in architecture-based self-adaptive systems with multiple quality requirements. In *Managing Trade-Offs in Adaptable Software Architectures*, pages 45–77. Elsevier, 2017.

[173] Sam Malek, Marija Mikic-Rakic, and Nenad Medvidovic. A style-aware architectural middleware for resource-constrained, distributed systems. *IEEE Transactions on Software Engineering*, 31(3):256–272, 2005.

[174] Indika Meedeniya, Irene Moser, Aldeida Aleti, and Lars Grunske. Architecture-based reliability evaluation under uncertainty. In *Proceedings of the joint ACM SIGSOFT conference–QoSA and ACM SIGSOFT symposium–ISARCS on Quality of software architectures–QoSA and architecting critical systems–ISARCS*, pages 85–94. ACM, 2011.

[175] Marcilio Mendonca, Moises Branco, and Donald Cowan. Splot: software product lines online tools. In *Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications*, pages 761–762. ACM, 2009.

[176] Marcilio Mendoncca, Thiago Tonelli Bartolomei, and Donald Cowan. Decision-making coordination in collaborative product configuration. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 108–113. ACM, 2008.

[177] Peter Midgley. Bicycle-sharing schemes: enhancing sustainable mobility in urban areas. *United Nations, Department of Economic and Social Affairs*, pages 1–12, 2011.

[178] Vajih Montaghami and Derek Rayside. Extending alloy with partial instances. *Abstract State Machines, Alloy, B, VDM, and Z*, pages 122–135, 2012.

[179] M. Moore, R. Kazman, M. Klein, and J. Asundi. Quantifying the value of architecture design decisions: lessons from the field. In *25th International Conference on Software Engineering (ICSE 2003)*, pages 557–562, 2003.

[180] John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and using nonfunctional requirements: A process-oriented approach. *Software Engineering, IEEE Transactions on*, 18(6):483–497, 1992.

[181] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. Representing knowledge about information systems in telos. In *Database Application Engineering with DAIDA*, pages 31–64. Springer, 1993.

[182] John Mylopoulos, Lawrence Chung, and Eric Yu. From object-oriented to goal-oriented requirements analysis. *Communications of the ACM*, 42(1):31–37, 1999.

[183] Patrenahalli M Narendra and Keinosuke Fukunaga. A branch and bound algorithm for feature subset selection. *Computers, IEEE Transactions on*, 100(9):917–922, 1977.

[184] Antonio J Nebro, Juan J Durillo, Francisco Luna, Bernabé Dorronsoro, and Enrique Alba. Mocell: A cellular genetic algorithm for multiobjective optimization. *International Journal of Intelligent Systems*, 24(7):726–746, 2009.

[185] Antonio J Nebro, Juan J Durillo, and Matthieu Vergne. Redesigning the jmetal multi-objective optimization framework. In *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 1093–1100. ACM, 2015.

[186] George L Nemhauser and Zev Ullmann. Discrete dynamic programming and capital allocation. *Management Science*, 15(9):494–505, 1969.

[187] Jürg Nievergelt. Exhaustive search, combinatorial optimization and enumeration: Exploring the potential of raw computing power. In *Sofsem 2000: theory and practice of informatics*, pages 18–35. Springer, 2000.

[188] Joost Noppen, Pim van den Broek, and Mehmet Akşit. Software development with imperfect information. *Soft computing*, 12(1):3–28, 2008.

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

[190] Jeho Oh, Don Batory, Margaret Myers, and Norbert Siegmund. Finding near-optimal configurations in product lines by random sampling. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, pages 61–71. ACM, 2017.

[191] A. O'Hagan, C.E. Buck, A. Daneshkhah, J.R. Eiser, P.H. Garthwaite, D.J. Jenkinson, J.E. Oakley, and T. Rakow. *Uncertain Judgements: Eliciting Experts' Probabilities*. Statistics in Practice. Wiley, 2006. ISBN 9780470033302.

[192] Rafael Olaechea, Steven Stewart, Krzysztof Czarnecki, and Derek Rayside. Modelling and multi-objective optimization of quality attributes in variability-rich software. In *Proceedings of the Fourth International Workshop on Nonfunctional System Properties in Domain Specific Modeling Languages*, page 2. ACM, 2012.

[193] Rafael Olaechea, Derek Rayside, Jianmei Guo, and Krzysztof Czarnecki. Comparison of exact and approximate multi-objective optimization for software product lines. In *Proceedings of the 18th International Software Product Line Conference-Volume 1*, pages 92–101. ACM, 2014.

[194] Olawole Oni. Towards a bayesian decision model for release planning in incremental development. In *Requirements Engineering Conference (RE), 2017 IEEE 25th International*, pages 520–525. IEEE, 2017.

[195] Olawole Oni and Emmanuel Letier. Optimizing the incremental delivery of software features under uncertainty. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 36–41. Springer, 2016.

[196] Matheus Paixão and Jerffeson Souza. A scenario-based robust model for the next release problem. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 1469–1476. ACM, 2013.

[197] Terence Parr. *The definitive ANTLR 4 reference*. Pragmatic Bookshelf, 2013.

[198] Terence Parr. *The definitive ANTLR 4 reference*. Pragmatic Bookshelf, 2013.

[199] Liliana Pasquale, Paola Spoletini, Mazeiar Salehie, Luca Cavallaro, and Bashar Nuseibeh. Automating trade-off analysis of security requirements. *Requirements Engineering*, pages 1–24, 2015.

[200] Diego Perez-Palacin and Raffaela Mirandola. Uncertainties in the modeling of self-adaptive systems: a taxonomy and an example of availability evaluation. In *Proceedings of the 5th ACM/SPEC international conference on Performance engineering*, pages 3–14. ACM, 2014.

[201] Antônio Mauricio Pitangueira, Rita Suzana P. Maciel, Márcio de Oliveira Barros, and Aline Santos Andrade. A systematic review of software requirements selection and prioritization using sbse approaches. In *5th International Symposium on Search Based Software Engineering (SSBSE 2013)*, pages 188–208. Springer Berlin Heidelberg, 2013.

[202] Antonio Mauricio Pitangueira, Paolo Tonella, Angelo Susi, Rita Suzana Maciel, and Marcio Barros. Risk-aware multi-stakeholder next release planning using multi-objective optimization. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pages 3–18. Springer, 2016.

[203] Klaus Pohl, Günter Böckle, and Frank J van Der Linden. *Software product line engineering: foundations, principles and techniques.* Springer Science & Business Media, 2005.

[204] Jakob Puchinger and Günther R Raidl. *Combining metaheuristics and exact algorithms in combinatorial optimization: A survey and classification.* Springer, 2005.

[205] Chaiyong Ragkhitwetsagul, Matheus Paixao, Manal Adham, Saheed Busari, Jens Krinke, and John H Drake. Searching for configurations in clone evaluation–a replication study. In *International Symposium on Search Based Software Engineering*, pages 250–256. Springer, 2016.

[206] Outi Räihä, Kai Koskimies, and Erkki Mäkinen. Generating software architecture spectrum with multi-objective genetic algorithms. In *Nature and Biologically Inspired Computing (NaBIC), 2011 Third World Congress on*, pages 29–36. IEEE, 2011.

[207] Andres J Ramirez, Adam C Jensen, and Betty HC Cheng. A taxonomy of uncertainty for dynamically adaptive systems. In *Proceedings of the 7th International Symposium on Software Engineering for Adaptive and Self-Managing Systems*, pages 99–108. IEEE Press, 2012.

[208] Norman Riegel and Joerg Doerr. A systematic literature review of requirements prioritization criteria. In *Requirements Engineering: Foundation for Software Quality*, pages 300–317. Springer, 2015.

[209] Suzanne Robertson and James Robertson. *Mastering the requirements process: Getting requirements right.* Addison-wesley, 2012.

[210] William N Robinson. *Integrating multiple specifications using domain goals*, volume 14. ACM, 1989.

[211] Colette Rolland, Carine Souveyet, and Camille Ben Achour. Guiding goal modeling using scenarios. *IEEE transactions on software engineering*, 24(12):1055–1071, 1998.

[212] Douglas T Ross and Kenneth E Schoman. Structured analysis for requirements definition. *IEEE transactions on Software Engineering*, (1):6–15, 1977.

[213] Jean-François Roy, Jason Kealey, and Daniel Amyot. Towards integrated tool support for the user requirements notation. In *International Workshop on System Analysis and Modeling*, pages 198–215. Springer, 2006.

[214] Nick Rozanski and Eóin Woods. *Software systems architecture: working with stakeholders using viewpoints and perspectives.* Addison-Wesley, 2011.

[215] Guenther Ruhe and Joseph Momoh. Strategic release planning and evaluation of operational feasibility. In *System Sciences, 2005. HICSS'05. Proceedings of the 38th Annual Hawaii International Conference on*, pages 313b–313b. IEEE, 2005.

[216] Günther Ruhe. *Product Release Planning: Methods, Tools and Applications.* CRC Press, 2010.

[217] Günther Ruhe and Des Greer. Quantitative studies in software release planning under risk and resource constraints. In *Empirical Software Engineering, 2003. ISESE 2003. Proceedings. 2003 International Symposium on*, pages 262–270. IEEE, 2003.

[218] Gunther Ruhe and Moshood Omolade Saliu. The art and science of software release planning. *Software, IEEE*, 22(6):47–53, 2005.

[219] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William E. Lorensen, et al. *Object-oriented modeling and design*, volume 199. Prentice-hall Englewood Cliffs, NJ, 1991.

[220] Respect-IT SA. KAOS/Objectiver. http://www.objectiver.com. [Online; accessed 28-November-2017].

[221] Thomas L Saaty. How to make a decision: the analytic hierarchy process. *European journal of operational research*, 48(1):9–26, 1990.

[222] Mehrdad Sabetzadeh, Davide Falessi, Lionel Briand, Stefano Di Alesio, Dag McGeorge, Vidar Åhjem, and Jonas Borg. Combining goal models, expert elicitation, and probabilistic simulation for qualification of new technology. In *High-Assurance Systems Engineering (HASE), 2011 IEEE 13th International Symposium on*, pages 63–72. IEEE, 2011.

[223] Mehrdad Sabetzadeh, Davide Falessi, Lionel Briand, and Stefano Di Alesio. A goal-based approach for qualification of new technologies: Foundations, tool support, and industrial validation. *Reliability Engineering & System Safety*, 119:52–66, 2013.

[224] Mohsen Sadatsafavi, Nick Bansback, Zafar Zafari, Mehdi Najafzadeh, and Carlo Marra. Need for speed: an efficient algorithm for calculation of single-parameter expected value of partial perfect information. *Value in Health*, 2013.

[225] Moshood Omolade Saliu and Guenther Ruhe. Bi-objective release planning for evolving software systems. In *Proceedings of the the 6th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering*, pages 105–114. ACM, 2007.

[226] Omolade Saliu and Guenther Ruhe. Supporting software release planning decisions for evolving systems. In *Software Engineering Workshop, 2005. 29th Annual IEEE/NASA*, pages 14–26. IEEE, 2005.

[227] Andrea Saltelli, Karen Chan, E Marian Scott, et al. *Sensitivity analysis*, volume 1. Wiley New York, 2000.

[228] Ana B Sánchez, Sergio Segura, José A Parejo, and Antonio Ruiz-Cortés. Variability testing in the wild: The drupal case study. *Software & Systems Modeling*, 16 (1):173–194, 2017.

[229] F. Sarro, F. Ferrucci, M. Harman, A. Manna, and J. Ren. Adaptive multi-objective evolutionary algorithms for overtime planning in software projects. *IEEE Transactions on Software Engineering*, 2017. doi: 10.1109/TSE.2017.2650914.

[230] Federica Sarro, Alessio Petrozziello, and Mark Harman. Multi-objective software effort estimation. In *Proceedings of the 38th International Conference on Software Engineering, ICSE 2016, Austin, TX, USA, May 14-22, 2016*, pages 619–630, 2016. doi: 10.1145/2884781.2884830.

[231] Pete Sawyer, Nelly Bencomo, Jon Whittle, Emmanuel Letier, and Anthony Finkelstein. Requirements-aware systems: A research agenda for re for self-adaptive systems. In *Requirements Engineering Conference (RE), 2010 18th IEEE International*, pages 95–103. IEEE, 2010.

[232] Abdel Salam Sayyad and Hany Ammar. Pareto-optimal search-based software engineering (posbse): A literature survey. In *Realizing Artificial Intelligence Synergies in Software Engineering (RAISE), 2013 2nd International Workshop on*, pages 21–27. IEEE, 2013.

[233] Abdel Salam Sayyad, Joseph Ingram, Tim Menzies, and Hany Ammar. Scalable product line configuration: A straw to break the camel's back. In *Automated Software Engineering (ASE), 2013 IEEE/ACM 28th International Conference on*, pages 465–474. IEEE, 2013.

[234] Abdel Salam Sayyad, Tim Menzies, and Hany Ammar. On the value of user preferences in search-based software engineering: a case study in software product lines. In *35th International Conference on Software Engineering (ICSE 2013)*, pages 492–501, 2013.

[235] Abdel Salam Sayyad, Tim Menzies, and Hany Ammar. On the value of user preferences in search-based software engineering: a case study in software product lines. In *Software engineering (ICSE), 2013 35th international conference on*, pages 492–501. IEEE, 2013.

[236] P-Y Schobbens, Patrick Heymans, and J-C Trigaux. Feature diagrams: A survey and a formal semantics. In *14th IEEE International Requirements Engineering Conference (RE 2006)*, pages 139–148. IEEE, 2006.

[237] Roberto Sebastiani, Paolo Giorgini, and John Mylopoulos. Simple and minimum-cost satisfiability for goal models. In *International Conference on Advanced Information Systems Engineering*, pages 20–35. Springer, 2004.

[238] Norbert Siegmund, Marko Rosenmuller, Martin Kuhlemann, Christian Kastner, Sven Apel, and Gunter Saake. Spl conqueror: Toward optimization of non-functional properties in software product lines. *Software Quality Journal*, 20(3-4): 487–517, 2012.

[239] Christopher L Simons, Ian C Parmee, and Rhys Gwynllyw. Interactive, evolutionary search in upstream object-oriented class design. *Software Engineering, IEEE Transactions on*, 36(6):798–816, 2010.

[240] David B Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the eleventh international conference on machine learning*, pages 293–301, 1994.

[241] Eric P Smith. Uncertainty analysis. *Encyclopedia of environmetrics*, 2002.

[242] Ian Sommerville and Gerald Kotonya. *Requirements engineering: processes and techniques*. John Wiley & Sons, Inc., 1998.

[243] K Srinivas, MP Gupta, et al. Software requirements selection using quantum-inspired elitist multi-objective evolutionary algorithm. In *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, pages 782–787. IEEE, 2012.

[244] Salvatore J Stolfo, Wenke Lee, Philip K Chan, Wei Fan, and Eleazar Eskin. Data mining-based intrusion detectors: an overview of the columbia ids project. *ACM SIGMOD Record*, 30(4):5–14, 2001.

[245] Felix Streichert, Holger Ulmer, et al. Javaeva: a java based framework for evolutionary algorithms. 2005.

[246] Alistair G Sutcliffe and Neil AM Maiden. Bridging the requirements gap: policies, goals and domains. In *Software Specification and Design, 1993., Proceedings of the Seventh International Workshop on*, pages 52–55. IEEE, 1993.

[247] SysML. SysML Specification. http://www.omgsysml.org, 2008. [Online; accessed 28-October-2016].

[248] Shan Tang, Xin Peng, Yijun Yu, and Wenyun Zhao. Goal-directed modeling of self-adaptive software architecture. In *Enterprise, Business-Process and Information Systems Modeling*, pages 313–325. Springer, 2009.

[249] Maurice H ter Beek, Alessandro Fantechi, Stefania Gnesi, and Franco Mazzanti. A collection of models of a bike-sharing case study. Technical report, Technical Report TR-QC-07-2014, QUANTICOL (May 2014), http://milner. inf. ed. ac. uk/wiki/files/y0R2Q6q/TRQC072014pdf. html, 2014.

[250] Maurice H Ter Beek, Alessandro Fantechi, and Stefania Gnesi. Applying the product lines paradigm to the quantitative analysis of collective adaptive systems. In *Proceedings of the 19th International Conference on Software Product Line*, pages 321–326. ACM, 2015.

[251] Stephen W Thomas, Hadi Hemmati, Ahmed E Hassan, and Dorothea Blostein. Static test case prioritization using topic models. *Empirical Software Engineering*, 19(1):182–212, 2014.

[252] Boris A Trakhtenbrot. A survey of russian approaches to perebor (brute-force searches) algorithms. *Annals of the History of Computing*, 6(4):384–400, 1984.

[253] Pablo Trinidad, David Benavides, Antonio Ruiz-Cortés, Sergio Segura, and Alberto Jimenez. Fama framework. In *Software Product Line Conference, 2008. SPLC'08. 12th International*, pages 359–359. IEEE, 2008.

[254] UML. UML Specification. http://www.uml.org, 2008. [Online; accessed 28-October-2016].

[255] Marjan van den Akker, Sjaak Brinkkemper, Guido Diepen, and Johan Versendaal. Software product release planning through optimization and what-if analysis. *Information and Software Technology*, 50(1):101–111, 2008.

[256] A van Lamsweerde and E Letier. Handling obstacles in goal-oriented software engineering. *IEEE Transactions on Software Engineering*, 26(10), 2000.

[257] Axel Van Lamsweerde. Divergent views in goal-driven requirements engineering. In *Joint Proceedings of the Sigsoft96 Workshops–Specifications96, ACM.* Citeseer, 1996.

[258] Axel Van Lamsweerde. Requirements engineering in the year 00: a research perspective. In *Proceedings of the 22nd international conference on Software engineering*, pages 5–19. ACM, 2000.

[259] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262. IEEE, 2001.

[260] Axel Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Requirements Engineering, 2001. Proceedings. Fifth IEEE International Symposium on*, pages 249–262. IEEE, 2001.

[261] Axel Van Lamsweerde. From system goals to software architecture. In *Formal Methods for Software Architectures*, pages 25–43. Springer, 2003.

[262] Axel Van Lamsweerde. Elaborating security requirements by construction of intentional anti-models. In *Proceedings of the 26th International Conference on Software Engineering*, pages 148–157. IEEE Computer Society, 2004.

[263] Axel van Lamsweerde. Goal-oriented requirements enginering: a roundtrip from research to practice [enginering read engineering]. In *Requirements Engineering Conference, 2004. Proceedings. 12th IEEE International*, pages 4–7. IEEE, 2004.

[264] Axel van Lamsweerde. *Requirements Engineering: From System Goals to UML Models to Software Specifications.* Wiley, 2009.

[265] Axel van Lamsweerde. Reasoning about alternative requirements options. In *Conceptual Modeling: Foundations and Applications*, pages 380–397. Springer, 2009.

[266] Axel Van Lamsweerde and Emmanuel Letier. From object orientation to goal orientation: A paradigm shift for requirements engineering. In *Radical Innovations of Software and Systems Engineering in the Future*, pages 325–340. Springer, 2004.

[267] Varsha Veerappa. *Clustering methods for requirements selection and optimisation.* PhD thesis, UCL (University College London), 2013.

[268] Junli Wang and Yubing Hou. Optimal web service selection based on multi-objective genetic algorithm. In *Computational Intelligence and Design, 2008. ISCID'08. International Symposium on*, volume 1, pages 553–556. IEEE, 2008.

[269] Shuai Wang, Shaukat Ali, Tao Yue, Yan Li, and Marius Liaaen. A practical guide to select quality indicators for assessing pareto-based search algorithms in search-based software engineering. In *Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on*, pages 631–642. IEEE, 2016.

[270] Bo Wei, Zhi Jin, Didar Zowghi, and Bin Yin. Implementation decision making for internetware driven by quality requirements. *Science China Information Sciences*, 57(7):1–19, 2014.

[271] Jules White, Brian Doughtery, and Douglas C Schmidt. Filtered cartesian flattening: An approximation technique for optimally selecting features while adhering to resource constraints. In *SPLC (2)*, pages 209–216, 2008.

[272] Jules White, Brian Dougherty, and Douglas C Schmidt. Selecting highly optimal architectural feature sets with filtered cartesian flattening. *Journal of Systems and Software*, 82(8):1268–1284, 2009.

[273] Karl Wiegers. *More about software requirements: thorny issues and practical advice.* Microsoft Press, 2005.

[274] Robert L. Winkler. *An introduction to Bayesian inference and decision / Robert L. Winkler.* Holt, Rinehart and Winston New York, 1972.

[275] Robert L Winkler. *An introduction to Bayesian inference and decision (2nd Edition).* Probabilistic Publishing, 2003.

[276] Eric Yu. Modelling strategic relationships for process reengineering. *Social Modeling for Requirements Engineering*, 11:2011, 2011.

[277] Eric Yu and Lin Liu. Modelling trust for system design using the i* strategic actors framework. In *Trust in Cyber-societies*, pages 175–194. Springer, 2001.

[278] Eric SK Yu. Towards modelling and reasoning support for early-phase requirements engineering. In *Requirements Engineering, 1997., Proceedings of the Third IEEE International Symposium on*, pages 226–235. IEEE, 1997.

[279] Lin Liu Eric Yu. From requirements to architectural design–using goals and scenarios. In *First International Workshop From Software Requirements to Architectures-STRAW*, volume 1, page 22, 2001.

[280] Yijun Yu, Haruhiko Kaiya, Nobukazu Yoshioka, Zhenjiang Hu, Hironori Washizaki, Yingfei Xiong, and Amin Hosseinian-Far. Goal modelling for security problem matching and pattern enforcement. *International Journal of Secure Software Engineering (IJSSE)*, 8(3):42–57, 2017.

[281] Yuanyuan Zhang. *Multi-Objective Search-based Requirements Selection and Optimisation.* PhD thesis, University of London, 2010.

[282] Yuanyuan Zhang and Mark Harman. Search based optimization of requirements interaction management. In *Search Based Software Engineering (SSBSE), 2010 Second International Symposium on*, pages 47–56. IEEE, 2010.

[283] Yuanyuan Zhang, Mark Harman, and S Afshin Mansouri. The multi-objective next release problem. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 1129–1137. ACM, 2007.

[284] Yuanyuan Zhang, Enrique Alba, Juan J Durillo, Sigrid Eldh, and Mark Harman. Today/future importance analysis. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pages 1357–1364. ACM, 2010.

[285] Yuanyuan Zhang, Mark Harman, and Soo Ling Lim. Empirical evaluation of search based requirements interaction management. *Information and Software Technology*, 55(1):126–152, 2013.

[286] Liu Zhuang, Guo HeQing, Li Dong, Han Tao, and Zhang Juan Juan. Solving multi-objective and fuzzy multi-attributive integrated technique for qos-aware web service selection. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 735–739. IEEE, 2007.

[287] Eckart Zitzler. *Evolutionary algorithms for multiobjective optimization: Methods and applications*, volume 63. Citeseer, 1999.

[288] Eckart Zitzler and Simon Künzli. Indicator-based selection in multiobjective search. In *Parallel Problem Solving from Nature-PPSN VIII*, pages 832–842. Springer, 2004.

[289] Eckart Zitzler, Marco Laumanns, Lothar Thiele, Eckart Zitzler, Eckart Zitzler, Lothar Thiele, and Lothar Thiele. Spea2: Improving the strength pareto evolutionary algorithm, 2001.

[290] Eckart Zitzler, Lothar Thiele, Marco Laumanns, Carlos M Fonseca, and Viviane Grunert Da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *Evolutionary Computation, IEEE Transactions on*, 7(2): 117–132, 2003.