Making Trade-Offs among Security and Other Requirements during System Design

by

Golnaz Elahi

A thesis submitted in conformity with the requirements for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Copyright © 2012 by Golnaz Elahi
Abstract

Making Trade-Offs among Security and Other Requirements during System Design

Golnaz Elahi

Doctor of Philosophy

Graduate Department of Computer Science

University of Toronto

2012

Employing a design solution can satisfy some requirements while having negative side-effects on some other software requirements and project objectives. Ultimately, selecting a design solution among multiple options involves making trade-offs among competing requirements. These trade-offs, especially at the early stages of software development, are often hard to identify or quantify, and can be subjective. Security is one critical requirement among many, which can cause critical trade-offs and severe costs. Damages from security attacks can be overwhelming and the costs increase every year. The threat of vulnerabilities and their exploitation by potential adversaries calls for careful analysis of security risks and trade-offs that security solutions impose, from the viewpoints of both defenders and attackers.

Since software developers and analysts are usually not security experts, detecting potential threats within software systems can be problematic. Even when threats are known, the risk factors, either the probability of a successful attack or the resulting damage of a successful attack, are not always known or numerically measurable. In this situation, selecting proper security solutions can be challenging, when mitigating impacts and side-effects of solutions are often not quantifiable.

This thesis addresses such challenges in identifying and making trade-offs among security and other system requirements and stakeholders’ goals. This work introduces a framework for identifying and modeling security risks and requirements trade-offs. The
central idea in this thesis is analyzing security requirements on the basis of predicting software vulnerabilities, weaknesses or flaws that can be exploited to break into the system. Vulnerabilities and exploitation scenarios are specified within goal-oriented requirements models of the system. This approach enables analysis of vulnerability exploitations and their impacts on the running system. The structure of goal-oriented security requirements models enables tracing the ultimate impacts of the exploitations on high-level goals of stakeholders and design objectives.

In order to evaluate the risk of vulnerabilities, this framework intertwines the Common Vulnerability Scoring System (CVSS) with security requirements risk assessment. The proposed framework provides a decision aid method that takes into the account risks, competing requirements, security solutions, their impacts on risks, and their side-effects on other requirements, to aid decision makers to select a solution among alternative security solutions. The proposed decision analysis method helps analysts to make requirements trade-offs systematically, in the absence of quantitative data, or when a mixture of both quantitative and qualitative data are available.
Acknowledgements

I am in debt to my supervisor, Eric Yu, for introducing me to the research area of goal-oriented modeling, and for helping me to define and refine my ideas on the topic of security trade-offs modeling and analysis. This work would have not been possible without Eric’s support and his dedication to the projects. My special thanks go to John Mylopoulos and Steve Easterbrook, my PhD committee members, for providing constructive feedbacks on my ideas and results. I would like to thank Lin Liu from Software Institute in Tsinghua University: for her feedback on my research and for constant support in all ups and downs I faced in the last few years.

There are many individuals, institutes, and companies that directly and indirectly had a constructive influence on my work: I appreciate the detailed review and excellent comments of external examiner of this thesis, Dr. Andreas Opdahl. I would like to thank Dimitrios Hatzinakos for his valuable feedback.

I would like to thank Yuan Xiang Gu from Irdeto, Ottawa, for introducing me to the area of White-Box security. I gratefully acknowledge the financial support received from Irdeto, Ottawa as part of a MITACS accelerate PhD program. I would like to thank Nicola Zannone for his tremendous impact on my research and being a great mentor. I would like to thank Zeev Lieber from SlashID corporation, Maria Carmela Annosi from Ericsson Software Research in Italy, Rohit Sethi from Security Compass, Roger Browne at Ministry of Transportation, Ontario (MTO), ONE-ITS project members specially Professor Tamer El-Diraby and Mahmoud Osman Abou-Beih, Jorge Silva at the
Adaptive Technology Resource Centre at the University of Toronto, and Greg Wilson from University of Toronto.

I would like to thank my friends and colleagues at the department of computer science at U of T, software institute at Tsinghua University, and specially labmates at software engineering lab at U of T: Mehrdad SabetZadeh, Shiva Nejati, Jocelyn Simmonds, Jennifer Horkoff, Hanieh Bastani, Yiqiao Wang, Elizabeth Lam, Mihaela Bobaru, Nan Niu, Maryam Fazel Zarandi, Neil Ernst, Michalis Famelis, Hesam Chiniforooshan, Justin Ward, Aws Albarghouthi, Mo Sadoghi, Sara Mostafavi, Eric Hsu, Alicia Grubb, Niloofar Razavi, Zachary Kincaid, Jonathan Lung, Jorge Aranda, Zhang He, and Tong Li. Five years of my PhD studies would have been impossible without Jocelyn, Jennifer, and Amir.

And on a personal note, I like to thank my parents, my brother, and my life partner, Amir. I am grateful for their support, love, and encouragements. My mother made sure to call me almost every day and gave me the extra strength I needed to keep going. My father is my inspiration in my life. My brother is my true mentor, and Amir is my source of energy, happiness, and encouragements. Thank you all.
# Contents

## 1 Introduction and Motivation

1.1 Illustrative Example: Digital Rights Management .................................. 2  
1.2 Challenges .................................................................................. 5  
1.3 Research Objectives .................................................................. 8  
1.4 Contributions ........................................................................... 9  
1.4.1 Component One: Security Requirements Modeling ................. 10  
1.4.2 Component Two: Risk Assessment ......................................... 13  
1.4.3 Component Three: Decision Analysis ................................. 14  
1.5 Organization ........................................................................... 15

## 2 Background and Related Work

2.1 Modeling for Security Requirements Analysis ................................. 17  
2.1.1 Black Side: Vulnerability and Attack Modeling .................... 17  
2.1.2 White Side: Security Requirements and Countermeasures Modeling 20  
2.1.3 Limitations of Existing Modeling Approaches ..................... 22  
2.2 Vulnerability Modeling and Analysis Approaches ............................. 23  
2.2.1 Vulnerability Catalogs ....................................................... 24  
2.2.2 Modeling Vulnerabilities for Security Requirements Engineering . 24  
2.2.3 Comparison of the Vulnerability Modeling Notations ............. 25  
2.3 Security Requirements Frameworks ............................................. 26
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.3.1 Agent and Goal-Oriented Requirements Frameworks</td>
<td>26</td>
</tr>
<tr>
<td>2.3.2 Trust-Based Requirements Frameworks</td>
<td>28</td>
</tr>
<tr>
<td>2.3.3 UML-Based Requirements Frameworks</td>
<td>29</td>
</tr>
<tr>
<td>2.4 Security Risk Assessment Methods</td>
<td>30</td>
</tr>
<tr>
<td>2.4.1 Limitations of Current Risk Assessment Methods</td>
<td>33</td>
</tr>
<tr>
<td>2.5 Multi-Criteria Decision Analysis Methods</td>
<td>34</td>
</tr>
<tr>
<td>2.5.1 Requirements Decision Analysis</td>
<td>38</td>
</tr>
<tr>
<td>2.5.2 Challenges in (Security) Requirements Decisions</td>
<td>40</td>
</tr>
<tr>
<td>3 A Framework for Security Requirements Trade-off Analysis</td>
<td>46</td>
</tr>
<tr>
<td>3.1 The Modeling Process</td>
<td>47</td>
</tr>
<tr>
<td>3.2 The Risk Assessment Process</td>
<td>50</td>
</tr>
<tr>
<td>3.3 The Decision Analysis Process</td>
<td>52</td>
</tr>
<tr>
<td>4 Modeling Security Requirements to Support Trade-off Analysis</td>
<td>54</td>
</tr>
<tr>
<td>4.1 The Meta-Model of Modeling Notation</td>
<td>56</td>
</tr>
<tr>
<td>4.1.1 Relevant Concepts</td>
<td>56</td>
</tr>
<tr>
<td>4.1.2 The Extended $i^*$ Meta-Model</td>
<td>58</td>
</tr>
<tr>
<td>4.2 Modeling Vulnerabilities</td>
<td>64</td>
</tr>
<tr>
<td>4.2.1 Eliciting and Modeling Initial Requirements</td>
<td>64</td>
</tr>
<tr>
<td>4.2.2 Modeling Vulnerabilities</td>
<td>66</td>
</tr>
<tr>
<td>4.2.3 Vulnerability Propagation</td>
<td>68</td>
</tr>
<tr>
<td>4.3 Modeling Exploitation Scenarios (Attacks)</td>
<td>69</td>
</tr>
<tr>
<td>4.4 Modeling Countermeasures</td>
<td>71</td>
</tr>
<tr>
<td>4.5 The Iterative Nature of the Modeling Process</td>
<td>73</td>
</tr>
<tr>
<td>4.6 Case Studies and Evaluation</td>
<td>77</td>
</tr>
<tr>
<td>4.6.1 Exploratory Study</td>
<td>77</td>
</tr>
<tr>
<td>4.6.2 Modeling Case Studies</td>
<td>82</td>
</tr>
</tbody>
</table>
5 Risk Assessment

5.1 From Vulnerabilities to Risks ........................................ 97
5.2 Adapting CVSS for Vulnerability-Centric Risks Assessment .......... 100
  5.2.1 Adaptation of CVSS for Risk Assessment .................... 101
  5.2.2 Tailoring CVSS for Un-trusted Environments .................. 105
  5.2.3 Aggregating CVSS Metrics .................................... 106
5.3 Risk Treatment ..................................................... 109
  5.3.1 Identifying and Analyzing Security Solutions ................. 110
5.4 Case Study ....................................................... 112
  5.4.1 Risk Assessment by CVSS .................................... 112
  5.4.2 Re-evaluating Risks in the Presence of Security Solutions .... 113
5.5 Discussion ....................................................... 114

6 Decision Analysis in the Absence of Numerical Data ................. 116

6.1 Comparing Alternatives for Analyzing Requirements Trade-offs .... 118
  6.1.1 Structuring the Decision Analysis Problem ................... 119
  6.1.2 Method Overview ............................................ 121
  6.1.3 The Decision Analysis Method based on Comparing Alternatives 127
  6.1.4 Identifying the Dominant Alternative by Even Swaps .......... 135
6.2 Semi-Automated Even Swaps with the Mixture of Qualitative and Quan-
titative Values ....................................................... 141
  6.2.1 Illustrative Example .......................................... 142
  6.2.2 The Algorithm Overview .................................... 144
  6.2.3 Selecting a pair of Alternatives for Even Swaps ............. 144
  6.2.4 Automatically Suggesting Even Swaps ....................... 147
6.3 Case Studies, Experiments, and Evaluation ........................ 153
  6.3.1 The Case Study at MTO .................................... 154
  6.3.2 Test Cases from Previous Work .............................. 156
6.3.3 Experiments .................................................. 160
6.4 Discussion ...................................................... 167
   6.4.1 Benefits of the Comparison-based Method .............. 167
   6.4.2 Threats to Validity and Usefulness of the Comparison-based Method 168
   6.4.3 Usefulness and Feasibility of Even Swaps ................. 169
   6.4.4 Decision Analysis Feedback on Models and Assessments .... 172

7 Evaluation and Contributions .................................... 175
   7.1 Case Study: Analysis of Web Application Vulnerabilities ...... 176
      7.1.1 Injection attack: OWASP Number One Vulnerability ....... 178
      7.1.2 Cross-Site Scripting: OWASP Second Top Vulnerability ... 184
   7.2 Comparison of Contributions with Existing Work ............. 191
      7.2.1 Security Requirements Modeling Component ............... 191
      7.2.2 Risk Assessment Component ................................ 194
      7.2.3 Decision Analysis Component .............................. 195

8 Tool Support for Decision Analysis ............................ 200
   8.1 Features and Functionalities ................................ 200
      8.1.1 Semi-Automated Even Swaps Module ...................... 201
      8.1.2 Comparison-Based Decision Analysis Module .............. 203
   8.2 Tool Design and Architecture ............................... 205
   8.3 Tool Benefits and Limitations .............................. 208

9 Conclusions and Future Work .................................... 211
   9.1 Conclusions and Limitations ................................. 212
      9.1.1 Modeling Component ...................................... 212
      9.1.2 Risk Assessment Component ............................... 215
      9.1.3 Decision Analysis Component .............................. 217
   9.2 Future Work ................................................ 220
List of Abbreviations

AC: Access Complexity
AHP: Analytic Hierarchy Process
ANP: Analytic Network Process
ATAM: Architecture Trade-off Analysis Method
AV: Access Vector
Au: Authentication
BBN: Bayesian Belief Nets
CAPEC: Common Attack Pattern Enumeration and Classification
CDP: Collateral Damage Potential
DoS: Denial of Service
CVSS: Common Vulnerability Scoring System
CWE: Common Weakness Enumeration
DRM: Digital Rights Managements
eSAP: electronic Single Assessment Process
E: Exploitability
FTA: Fault Tree Analysis
GA: Guardian Angel
GORE: Goal-oriented requirements engineering
GUI: Graphical User Interface
ITS: Intelligent Transportation System
KB: Knowledge Base
MACBETH: Measuring Attractiveness by a Categorical Based Evaluation Technique
MAUT: Multi Attribute Utility Theory
MCDA: Multi-criteria decision analysis
MTO: Ministry of Transportation, Ontario
MODA: Multiple Objective Decision Analysis
Morda: Mission Oriented Risk and Design Analysis
MRS: Marginal Rate Substitution
NFRs: Non-Functional Requirements
NVD: National Vulnerability Database
OCTAVE: Operationally Critical Threat, Asset, and Vulnerability Evaluation
OWASP: Open Web Application Security Project
QR: Qualitative Reasoning
RBAC: Role-Based Access Control
RC: Report Confidence
RE: Requirements Engineering
RL: Remediation Level
RoSI: Return on Security Investment
SAEM: Security Attributes Evaluations Method
SD: Strategic Dependency
SDL: Security Development Lifecycle
SDLC: systems development life-cycle
SR: Strategic Rationale
SVDT: Security Verification and security solution Design Trade-off analysis
TD: Target Distribution
VMS: Variable Message Sign
XSS: Cross Site Scripting
Chapter 1

Introduction and Motivation

*Security is about trade-offs, not absolutes.*

Ravi Sandhu [133]

Software developers and project managers make key decisions such as which technologies, architectural patterns, design solutions, or products to use [44]. These choices can have a significant impact on satisfaction of functional and non-functional requirements (NFRs) and achievement of design objectives and stakeholders’ goals. Often, adopting a solution to satisfy a group of goals can aid or detract from the satisfaction of some NFRs or obstruct some functionality [143]. Developing a software system that satisfies all the requirements and objectives of multiple stakeholders is practically impossible, and ultimately, stakeholders must make trade-offs among different competing requirements and goals.

In such decisions, security is typically only one design objective among many. Eliciting security requirements and selecting proper security solutions involves making trade-offs, because security usually competes with other objectives, e.g., reduces usability or performance, obstructs some functionality, or poses significant financial costs. Security goals can have their own contradictions because confidentiality, integrity, privacy, accountabil-
ity, availability, and recovery from security attack often conflict fundamentally.

Security is more of a cat and mouse game between the defenders and attackers [25]. Ultimately defenders attempt to mitigate the damage of attacks or increase the effort and time required for successful exploitations. One can only aim for “good enough” security, because firstly, absolute security is impossible (both in theory and practice). Secondly, security requirements can result in trade-offs with other design objectives, given the competing demands of many stakeholders. To select “good enough” security mechanisms requirements analysts and project leaders need to evaluate security risks objectively, which boils down to evaluating consequences of countermeasures on the probability and potential damage of attacks. Security countermeasures most likely have side-effects on other goals, and thus, selecting a security solution involves making trade-offs between security and other goals and requirements. This is ultimately a decision problem.

In a requirements trade-off decision problem, satisfaction of system requirements and stakeholders’ goals are the criteria for making the decision. In order to accommodate security concerns into this decision, security-related factors such as risk of attacks and vulnerabilities, security countermeasures and their remediation impacts must be evaluated and analyzed as well. Preferences of stakeholders over the decision criteria must be formulated into the decision problem as well. In the following section, a case scenario illustrates different aspects of the requirements trade-off problems that are addressed in this thesis.

1.1 Illustrative Example: Digital Rights Management

Consider a simple Digital Rights Managements (DRM) media player [25]. The player gets an encrypted media and given a valid user key, decrypts the media, and decodes the digital content to an analogue audio. The user must purchase an activation code for the player, and the player works only if the activation code is valid at the time of
using the player. The first concern is to identify and analyze potential threats in this domain: *What are the vulnerabilities within this system? How critical is the risk of these threats? What are the consequences of exploiting these vulnerabilities? Which risks can be neglected and which ones are critical?*

The DRM player contains two hard-coded credentials: *Valid activation code* and *Player key*. A software cracker can use the DRM player without buying an activation code, either through static code analysis to extract the valid code or by tampering with the binary code to bypass the license checks. The main protection strategies against *Tampering the binary code* attack are [25]:

1. **Obfuscation**: By obfuscating a program, the code is transformed into a form that is more difficult for an adversary to understand or change than the original code. Obfuscation adds an overhead to the code which causes performance drop downs. In a large code base, obfuscation needs to be done by automated tools that guarantee the obfuscated code behaves the same as the original program.

2. **Tamper Proofing**: Tamper proofing algorithms detect that the program has been modified. Once tampering has been detected, a tamper response is executed which usually causes the program to fail.

3. **Distribution with Physical Token**: Physical tokens are hardware-based protections that try to provide a safe environment for data, code and execution. By employing a physical token, the user needs to show possession of a token to use the software.

The second concern is analyzing alternative security countermeasures: *What is the impact of each solution on the risks? What are the side-effects of each solution on other requirements and goals?*

These alternative security solutions against tampering have side-effects on other goals such as portability, delay, and cost. Table 1.1 shows the impact of each solution on these factors and the tampering risk. For example, without applying any security solution, the
risk (of tampering) is High, and with the presence of different security solutions, this risk is reduced to different levels.

Consequence tables in general can contain heterogeneous data, i.e. different goals can be evaluated in different scales and by different techniques. For example, in Table 1.1, player’s delay ($G_3$) is estimated in milliseconds based on the properties and specification of alternatives. On the other hand, not enough information is available to quantitatively measure the risk of the tampering attack, so this risk (in the presence of different security solutions) is evaluated in the ordinal scale of Low, Medium, and High.

Table 1.1: Consequences of alternative security solutions on the DRM player requirements and the risk of tampering

<table>
<thead>
<tr>
<th>Alternative</th>
<th>Risk</th>
<th>Minimize</th>
<th>Maximize</th>
<th>Minimize</th>
</tr>
</thead>
<tbody>
<tr>
<td>No security solution</td>
<td>High</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code Obfuscation</td>
<td>Medium</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Tamper Proofing</td>
<td>Low</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Distribution with Physical Token</td>
<td>Low</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The decision over the alternative solutions must take the preferences of stakeholders into consideration. For example, the delay and portability affect the final decision over these solutions, but security and cost could be significantly more important for stakeholders. The third objective of this thesis is analyzing such decision problems: Which of the alternative solutions offer the best trade-off between security gains and negative impacts on other goals? How do the stakeholders’ preferences affect the final decision? Which solution is the optimum one with respect to risks, side-effects of solutions on stakeholders’ goals, and their preferences (when some of the criteria are not measured quantitatively)?

This thesis addresses the raised concerns and issues in the above example: risk identification and assessment, analyzing security solutions, and systematic decisions analysis.
1.2 Challenges

We have grouped security decision analysis approaches into two categories: quantitative and qualitative. We explain the underlying idea of each category and discuss the challenges of applying them to make security trade-offs.

Quantitative Approaches: Quantitative methods take (financial) costs and benefits of security solutions into account, and usually calculate a utility for each solution. These methods (e.g., [22, 68]) take a risk and security degree for each solution and assess how well the requirements are satisfied, compare alternative security solutions, and select an optimum solution based on purely numerical evidence.

The risk \( r \) of a security attack is described by the level of damage \( d \) that the successful attack poses to the system and the probability \( p \) that the attack occurs: \( r = p \times d \) \[90\]. A security solution may mitigate the damage to a lower level \( d' \) or reduce the probability of an attack to a lower chance \( p' \), so the risk is reduced to \( r' = p' \times d' \). If the damage is expressed in terms of financial costs, then the risk represents the financial damage of the attack. In this way, managers can easily decide whether to apply a security solution, by examining if the solution’s cost \( c \) is lower than the financial damage that they can save \( c < r - r' \). When managers need to decide over a set of alternative solutions, they can select a solution which minimizes the costs while mitigating the risks \( \min(c + r' - r) \).

However, measuring risks, software qualities, and design objectives is hard, because these factors are often intangible and soft in nature, especially in the early phases of the project. Numerical measures of quality goals and requirements may have no physical interpretation in the application domain [95]. Formal and rigorous ways to effectively calculate the strength of a design solution, in terms of requirements satisfaction levels, are not yet accepted and practical.

Measurement scales for software qualities are not usually available or agreed. Stakeholders may choose (or may be able) to evaluate some requirements numerically, and
some requirements qualitatively. Thus, ultimately, different requirements are evaluated in different scales, e.g., absolute values, percentages, ordinal numbers, or qualitative or nominal labels. For example, cost could be expressed by absolute measures, while privacy may be evaluated only by ordinal values. When different requirements are measured (or estimated) on different scales, normalizing inconsistent types of measures into a single scale is troublesome and may not result in meaningful utility values.

Using ordinal scales of measurement poses another problem: When stakeholders and decision analysts evaluate the security level of a solution in an ordinal scale (e.g., 1 to 9), they may incorrectly interpret the level of 8 as double the security of level 4. However, these numerical values are rough estimations in an ordinal scale, and assumptions about their ratio or intervals are mathematically incorrect [48]; thus, the use of utility theory or mathematical operations (e.g., average) over these numbers is not meaningful.

**Qualitative Approaches:** Making trade-offs among requirements and selecting a solution among multiple security alternatives is challenging. As pointed out in the studies on the psychology of security [136, 137, 49], security trade-offs are naturally subjective and hard to make: security decisions are based on the personal judgement and feeling of stakeholders and analysts. Different stakeholders have different expectations from the software system, as well as different levels of risk tolerance and personal privacy expectations. Such differences would be reflected on stakeholders’ preferences and their judgement about different solutions.

Most often, quantifying security evaluations and preferences is challenging. Even if some requirements can be refined into measurable variables, time and budget limitations preclude elaborate methods for obtaining quantitative data in the early stages of software development. Many non-functional requirements have a soft nature, which makes them hard, if not impossible, to measure. Such requirements are treated as soft goals in some Requirements Engineering (RE) techniques, e.g., $i^*$ [150], Tropos [53], and GRL [96]. Goal model evaluation techniques such as [24, 67, 54, 6], enable reasoning about the
partial satisfaction of soft goals by using qualitative labels such as partially satisfied (✓),
sufficiently satisfied (✓), partially denied (✗), and fully denied (✗).

Goal model evaluation techniques (such as [24, 67, 54]) enable analyzing requirements
satisfaction in the absence of numerical data. However, goal modeling methods that rely
on goal refinement for achieving finer-grained differentiation among alternatives typically
employ a minimal set of labels to indicate the satisfaction level of goals (e.g., ✓, ✓, ✗, ✗). These labels are meant to be used in conjunction with iterative goal refinement to arrive
at satisfactory solutions. However, such labels do not have precise meaning for decision
analysis purposes. For example, one cannot tell whether solution $A_1$, which satisfies
$G_1$ but denies $G_2$, should be preferred over solution $A_2$ which partially satisfies $G_1$ and
partially denies $G_2$, ($✓ + ✗ = ✓ + ✗$). Stakeholders are often able to make finer
distinctions among a given a set of alternatives with respect to impacts of alternatives
on the goals.

**Common Challenges:** Making trade-offs among requirements involves some other
challenges regardless of using quantitative or qualitative values for analysis:

1. **Manual requirements prioritization:** In general, eliciting relative attractiveness
(also known as preferences, relative importance, priorities) of goals and requirements
is a challenging process and requires stakeholders to answer cognitively-hard queries
about their preferences. Preference elicitation methods such as AHP may require up to
$m^2/2$ number of queries from stakeholders to compare preferences of $m$ goals. Extract-
ing stakeholders’ preferences over multiple criteria into numerical importance weights is
typically labor-intensive and error-prone.

2. **Incomparable scales:** Decision makers may choose to evaluate the satisfaction
level of different NFRs by using a mixture of techniques, and in different scales such
as absolute, ordinal, and ratio. Aggregating requirements measures in different scales is
usually error-prone or not possible.

3. **Scalability and extensive data collection:** Eliciting required information to
make an objective decision usually involves collecting an extensive amount of data from stakeholders. This process can become complicated and almost impossible when dealing with a large number of requirements and/or alternatives.

4. Subjective risk assessment by non-experts: In practice, evaluating the damage of attacks, estimating attack probability and mitigating the impact of countermeasures is challenging and error-prone. Risk assessment involves collecting the subjective (and often inaccurate) opinions of security experts, mining previous attack repositories (which may not be available), and collecting information from sparse security knowledge sources. Non-expert software analysts and project leaders, which are the majority of software professionals, sometimes have not had security-specific training, and yet are directly or indirectly responsible for satisfying the security goals. In most cases, they may not even know what relevant risks threaten their system or product.

1.3 Research Objectives

The objective of this work is to develop a framework that assists requirements analysts in identifying security trade-offs and making a decision over alternative security solutions objectively and systematically. By a systematic decision analysis method, we mean supporting analysis with a comprehensive process for modeling, risk assessment, and decision analysis. By an objective decision, we mean making decisions based on an informed judgement about the risks and trade-offs, even though the judgement about solutions is not entirely objective. To help analysts make a decision with the presence of qualitative, subjective, or incomplete data, we use modeling as a means for collecting more information, structuring the problem, and identifying the trade-offs. We use heuristics as a means to deal with the subjectivity of problem decision input data.

To this end, we need a way for expressing and evaluating costs and benefits of alternative security solutions. These costs and benefits are not limited to financial gains
or losses, and can be expressed as positive or negative effects of solutions on reducing security risks, achieving design objectives, and satisfying stakeholders’ goals.

In sum, the objective of this research is to seek for a systematic way that helps in identifying, analyzing, and making security trade-offs objectively under these conditions:

1. Some of the system requirements and stakeholders’ goals are competing.
2. The system is under the risk of some security vulnerabilities/attacks, and different vulnerabilities are associated with different degrees of risk.
3. Alternative design solutions have different costs and benefits (i.e., negative and positive impacts on requirements satisfaction and risks).
4. Stakeholders have different preferences over system requirements and design objectives.
5. Accurate and complete quantitative data to support the decision and rationalize about the costs and benefits of each solution is not available.
6. The system needs to simultaneously satisfy multiple requirements of different stakeholders, so the conflicting requirements are balanced out.

1.4 Contributions

The contribution of this thesis is developing a security requirements trade-off decision framework that addresses some of the raised challenges in security requirements trade-off analysis. Although this work focuses on security requirements, the decision analysis methods are applicable to any trade-off decision problem that deals with the above challenges. This framework has three main components:

1. **Security requirements modeling**: A modeling notation for expressing security vulnerabilities (i.e., weaknesses and flaws) and exploitation scenarios within goal-oriented requirements models of software systems.
2. Risk Assessment: A vulnerability-centric and qualitative risk assessment based on security requirements models of the first component.

3. Decision analysis: A decision analysis method that takes risk assessment results, and enables a systematic decision making in the absence of complete numerical data.

Figure 1.1 lists the contributions of each component. Although these three components are needed for a comprehensive security trade-off analysis process, each individual component can be seen as an independent contribution and can be used individually. Table 1.2 summarizes the structure of the thesis chapters based on the main three components of the framework and relevant publications. The components are further discussed in what follows.

1.4.1 Component One: Security Requirements Modeling

The first component of this thesis focuses on enhancing the $i^*$ modeling notation [150] with security specific elements. The $i^*$ agent- and goal-oriented framework enables analyzing software systems from the point of view of multiple interacting agents that have their own goals. The agents provide some services and rely on other agents for some services. This provides a suitable basis to analyze security risks that can threaten achievement of agents’ goals. In addition, the network of dependencies among actors helps analyze how the impacts of security risks propagate from one agent (or sub-system) to
Table 1.2: Contributions of three main components of the framework. (Publications that are based on these contributions are listed in Table 1.3)

<table>
<thead>
<tr>
<th>Component</th>
<th>Chapter</th>
<th>Publications</th>
<th>Contributions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security Requirements Modeling</td>
<td>4</td>
<td>ER07 [33]</td>
<td>1.a Explicitly expressing vulnerabilities and exploitations within models</td>
</tr>
<tr>
<td></td>
<td></td>
<td>DKE [34]</td>
<td>1.b Identifying trade-offs between security and other requirements and goals</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ER09 [38]</td>
<td>within goal-oriented models</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REJ [39]</td>
<td>1.c Tracing the impacts of low-level vulnerabilities to high-level goals and</td>
</tr>
<tr>
<td></td>
<td></td>
<td>STPSA [32]</td>
<td>requirements models</td>
</tr>
<tr>
<td>Risk Assessment</td>
<td>5</td>
<td>ER09 [38]</td>
<td>2.a Driving security risks based on vulnerabilities, criticality of</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>vulnerabilities, their impacts, and their exploitation scenarios</td>
</tr>
<tr>
<td></td>
<td></td>
<td>REJ [39]</td>
<td>2.b Intertwining CVSS with security requirements analysis for assessing the</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>risks associated with vulnerabilities</td>
</tr>
<tr>
<td>Decision Analysis</td>
<td>6</td>
<td>SAC [36]</td>
<td>3.a Systematic decision analysis</td>
</tr>
<tr>
<td></td>
<td></td>
<td>RE09 [35]</td>
<td>3.a.1 In the absence of direct evaluation, measurement, or estimation of</td>
</tr>
<tr>
<td></td>
<td></td>
<td>COMPSAC [37]</td>
<td>requirements and goals satisfaction</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3.a.2 In the presence of a mixture of quantitative and qualitative data in</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>varieties of scales</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3.b. Automating the Even Swaps decision analysis process and preferences</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>elicitation and intertwining it with security requirements trade-off analysis</td>
</tr>
</tbody>
</table>

another agent (or sub-system). These models enable analyzing attackers and their goals in terms of autonomous agents that interfere with the usual dependencies and goals of innocent agents.

The extended $i^*$ notation enables the expression of vulnerabilities and exploitation scenarios within the $i^*$ requirements model of the system. Thus the model can be used as the basis of evaluating the impacts of exploitations on the system requirements and
### Table 1.3: Related publications on the contributions of this thesis

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Title</th>
<th>Venue</th>
</tr>
</thead>
<tbody>
<tr>
<td>DKE [34]</td>
<td>Modeling and analysis of security trade-offs - A goal oriented approach</td>
<td>Data and knowledge Engineering Journal, 2009</td>
</tr>
<tr>
<td>ER09 [38]</td>
<td>A Modeling Ontology for Integrating Vulnerabilities into Security Requirements Conceptual Foundations</td>
<td>Conceptual Modeling Conference (ER’09)</td>
</tr>
<tr>
<td>SURV [31]</td>
<td>A Semi-Automated Decision Support Tool for Requirements Trade-off Analysis</td>
<td>COMPSAC’11 Conference</td>
</tr>
</tbody>
</table>

stakeholders’ goals. Vulnerabilities are usually flaws, bugs, and errors at the code level. By adding vulnerabilities to goal-oriented i* requirements models, the immediate impacts of such low-level weaknesses on the running system can be traced back to high level business goals. This is a pro-active approach, and before flaws are introduced into the final system during the implementation, they are anticipated and analyzed beforehand. The models can accommodate security mechanisms and the remediation impacts of solutions on security goals. The resulting models provide a basis to reason about mitigating impacts of solutions on the damage and/or probability of risks.
Contributions of this component are summarized in Table 1.2. The focus of this system is software security, and case studies and examples of the modeling approach address application security context (chapter 4 and 7). However, in general, \( i^* \) modeling and specifically security extensions on \( i^* \) can be used for system, network, infrastructure, or enterprise security as well.

Publications: We introduced the use of security concepts such as vulnerabilities and attacks along with the \( i^* \) framework in [33]. Several case studies were conducted in [34] which illustrate how the extended notation is more expressive than some existing security requirements modeling approaches. We formalized the meta-model of the notation in [39]. In [38], we identified basic concepts for modeling and analyzing vulnerabilities and generalized them into an ontology. We used the modeling notation for analyzing an Identity Management system in [32] and trust trade-off analysis in [35].

1.4.2 Component Two: Risk Assessment

The second component of this thesis is a vulnerability-centric risk assessment method. In the proposed method, we identify security risks by analyzing the criticality and exploitability of vulnerabilities within the system. We intertwine the Common Vulnerability Scoring System (CVSS) [109] framework within our risk assessment approach, so the probability and damage of exploitations on stakeholders’ goals are estimated by the CVSS vulnerability criticality metrics. Then, the mitigating impacts of security solutions on the damage and/or probability of attacks are analyzed. Security solutions may have side-effects on other requirements, for instance, may reduce performance and/or threaten privacy. Both financial costs and negative impacts on other requirements are considered in the final decision over alternative solutions.

Contributions of this component are summarized in Table 1.2. The focus of CVSS is IT vulnerabilities; thus although in case studies and examples of the risk assessment approach (chapter 5 and 7), we focus only on application security, the proposed risk assessment component can be used for general IT systems, network, infrastructure, or enterprise risk assessment.

Publications: In [39], we adapted goal model evaluation techniques (e.g., [24, 67]) for
analyzing the impacts of vulnerability exploitations on stakeholders’ goals and system requirements. In [35], we analyzed the risk associated with the abuse of trust dependencies. We intertwined the requirements analysis with vulnerability assessment using CVSS in [40].

1.4.3 Component Three: Decision Analysis

The third component of this thesis is a set of decision analysis methods that help requirements analysts select security solutions with respects to costs and benefits of alternative solutions and preferences of stakeholders over different requirements. Although the focus of this thesis is on analysis of security requirements, the decision analysis contribution can be applied to any general decision problem with multiple competing criteria.

We propose two main decision analysis methods for two different conditions: 1) when some factors are numerically measurable and the rest (or all factors) are evaluated qualitatively and in different scales, 2) when the decision criteria are not measurable quantitatively or even qualitatively. In the second condition, consequences of security solutions are not available. Thus, we seek out other types of information about the alternatives. We propose a decision analysis method that requires domain experts to compare consequences of solutions on decision criteria, instead of evaluating the strength of solutions’ impacts.

In both conditions, we ultimately adopt the Even Swaps multi-criteria decision analysis approach [64] to decide over alternative solutions. The Even Swaps process helps make trade-offs between decision criteria by trading one requirement for another, thus, it does not require eliciting importance weights of requirements and risks. Requirements and risks can be evaluated in a mixture of scales and by different measurement methods. We introduce a set of rules and a tool for tailoring and enhancing the Even Swaps process for security risk analysis to decide over alternative security solutions. The tool semi-automates the Even Swaps process to minimize the reasoning burden on human analysts. Although we use the tool to analyze security requirements trade-offs, the decision aid tool is not specifically designed for security decision problems, and it can solve any general decision problem which involves a family of criteria and alternative actions.

Publications: The qualitative decision analysis approach based on comparing alternatives
is introduced in [36]. A hybrid approach that enables analyzing both qualitative and quantitative evaluations and measurements is presented in [37].

1.5 Organization

This thesis is structured as follows. Chapter 2 presents an overview of existing security requirements analysis approaches, including modeling notations, risk assessment methods, security requirements analysis frameworks, and (security) decision analysis methods. In Chapter 3, a framework and process for modeling security requirements, vulnerabilities, and exploitations scenarios is presented. The framework also covers the of use the resulting models in risk assessment and decision analysis. Chapter 4 presents the extended $i^*$ notation, with security concepts for modeling vulnerabilities, exploitation scenarios, and countermeasures. In Chapter 5, the resulting models are used as the basis of a qualitative and vulnerability-centric risk assessment. The result of the risk assessment process is fed into a decision analysis method. In Chapter 6, a set of decision analysis methods are presented for making security requirements trade-offs and deciding over alternative solutions. Each component is illustrated and evaluated by a set of case studies at the end of each chapter. In Chapter 7, all three components of the framework are applied to a case study scenario as a proof of concept and an evaluation of the utility of the proposed methods. Chapter 7 compares the contributions of this work with existing similar approaches based on the results of case studies and analytical evaluations of proposals. Chapter 8 discusses the tool support for the decision analysis methods. Finally, we conclude in Chapter 9 with a summary of this thesis and an outline of future research directions.
Chapter 2

Background and Related Work

Security is a property of a system which remains dependable in the face of malice, error, or mischance [7]. As defined in ISO/IEC 13335 standard [75], in the scope of information systems, security consists of seven states: confidentiality, integrity, availability, authenticity, accountability, non-repudiation, and reliability. Such security goals are critical due to potential adversaries that attempt to compromise the system.

Security requirements engineering frameworks derive security requirements based on security-specific concepts, such as vulnerability, attack, risk, exploitation, countermeasure, etc. In security terminology, an attacker performs intentional unwarranted actions to break the security of a system by exploiting a vulnerability [136]. In addition to threat analysis, security requirements are derived from analysis of interactions and dependencies of social actors in the organizational contexts [98, 97]. Such considerations to extract security requirements bring other related issues such as trust among actors, ownership, permission, and delegation to security requirements modeling and analysis [51].

This chapter overviews security modeling notations and security requirements engineering frameworks. We survey existing risk assessment methods, and decision analysis techniques for selecting security countermeasures. The purpose of this chapter is to highlight existing and relevant work as well as challenges and gaps that are not addressed in current methods.
2.1 Modeling for Security Requirements Analysis

Models capture an abstraction of certain types of information or descriptions, communicate the information, make implicit information explicit, and act as a repository for knowledge and rationale. Conceptual modeling notations are a major area of study in software engineering for facilitating requirements extraction, management, analysis, and visualization. In addition, models can be used for system and architecture design, analysis, verification, and visualization.

Conceptual modeling notations for software engineering activities express different concepts to serve different purposes. Security issues demand special reasoning and analysis that traditional software engineering modeling approaches do not consider. For example, a general behavior modeling notation expresses interactions of entities in the system without considering the harmful behavior of an external adversary. Thus, the resulting models would not provide a basis for reasoning about the impacts of the malicious behavior of the adversary on requirements, design, and architecture.

Several security modeling notations have been developed for modeling and analyzing security-specific concepts such as attackers and their malicious behavior (attacks), vulnerabilities, assets, and countermeasures. We categorize these notations into two groups: the ones that focus on modeling attacks and vulnerabilities, and the ones that focus on specifying security protection and security requirements. We call the former groups black side notations and the latter white side modeling approaches. It should be mentioned that the categories can have overlaps, i.e. some of the modeling notations may provide conceptual foundations to model both white- and black-side concepts.

2.1.1 Black Side: Vulnerability and Attack Modeling

In some of the notations, attacks and security mechanisms are modeled in a similar way. In some methods, the goals and means of the attacker and defender are differentiated. Sindre and Opdahl [91] discuss that graphical models become much clearer if the distinction between malicious and non-malicious elements is made explicit and the malicious actions are visually distinguished from the legitimate ones. In particular, Sindre and Opdahl show that the use of
inverted elements strongly draws the attention to dependability aspects early on for those who
discuss the models. The notion of Misuse Cases [140], Abuse Cases [108], Anti-goals [142], and
malicious actors in the i* Framework [98, 97, 33] are examples of approaches that invert icons
or concepts to indicate the functionality not wanted in the system.

A misuse case is a use case, from the point of view of a hostile actor, which poses threats
to the system. In this approach, countermeasures are expressed as security use cases which
mitigate misuse cases. To express such impacts of countermeasures, use case relationships
are extended by adding a "prevent" relationship to the misuse case notations. Sindre and
Opdahl [140] claim that looking at systems from a misuser perspective increases the chance of
discovering threats that would otherwise be ignored.

Rostad [130] suggests extending the misuse case notation for modeling vulnerabilities and
insider attackers. The extended notation introduces a new type of actor called an insider that
has capabilities and permissions of an authorized actor who misuses the given permissions. The
concept of vulnerability is defined as a weakness that may be exploited by the misuse. The
other similar notation based on conventional UML use case modeling is the notion of an abuse
case [108]. While use cases are abstract episodes of useful and desired interactions between
a system and its environment, abuse cases are specifications of interactions whose results are
harmful to the system. An abuse case describes the abuse of privilege used in normal use case.

Liu et al. [98] analyze attackers, countermeasures, access control, and vulnerabilities in
actors dependency networks, of i* models [150]. In this approach, all actors are assumed to
be potential attackers, and inherit the capabilities, intentions, and social relationships of the
corresponding legitimate actors. The actors in the basic dependency model are substituted with
their corresponding attackers, and then the impact of the attack on the dependency relationship
is traced to other actors. In another use of i* modeling for security analysis in [33], inverted i*
elements are employed for distinguishing malicious actors, goals, and tasks of attackers.

Attack modeling notations mostly focus on decomposing the attacker’s goal into a set of
actions [135] or required steps and conditions to mount an attack [107, 124]. Schneier was first
to associate the term “Attack Tree” with the use of tree style models for expressing different
ways in which a system can be attacked [135]. The Attack Tree is suggested as a formal and
methodical way for describing the security of the system based on varying attacks. The root node of an attack tree is the goal of the attack and a node can be decomposed by AND/OR relations. Leaf nodes represent different ways to achieve the top goals. Schneier claims that to select proper countermeasures against attacks, designers benefit from understanding all the different ways in which a system can be attacked, who the attackers are, and what their abilities, motivations, and goals are.

Fault Tree Analysis (FTA) [144] is one of the most commonly-used techniques in reliability engineering. It determines various combinations of hardware and software failures and human errors. To develop Fault Tree models, system failures are identified and each failure is analyzed on its own Fault Tree. Then the Fault Tree is refined into more detailed events. The leaves of Fault Trees are undevelop/external events. The logical relationship is represented by several types of logical gates such as AND, OR, XOR, PRIORITY, etc. Fault Trees have been used for analysis of failure conditions of complex software systems. For example, Helmer et al. [65] employs Fault Trees for identifying and analyzing requirements of Intrusion Detection Systems.

McDermott [107] proposes the Attack Nets method for modeling attacks as a Petri Nets with a set of places representing interesting states or modes of the security relevant entities of the system. The Attack Net modeling approach enables expressing the steps of an attack, and attacker actions. Attack Nets has a set of transitions that represent input events, commands, or data that cause one or more security relevant entities to change their states. The Attack Nets has a set of tokens, which move from one place to another, along the directed arcs, to indicate the progress or concurrency of the attack.

Phillips et al. [124] introduced Attack Graphs to analyze vulnerabilities in computer networks. Attack Graphs provide a method for modeling attacks and relating them to the machines in a network and to the attackers. The Attack Graph model is developed based on attack templates, attack profiles, and network configurations. Attack templates describe generic steps in known attacks and conditions which must be held. These conditions are matched with the network configuration and attacker profile to generate the Attack Graph. An Attack Graph is an attack template instantiated with particular attackers/users and machines, and to analyze the Attack Graph one needs to identify the attack paths that are most likely to succeed.
Chapter 2. Background and Related Work

Sheyner et al. [138] argue that constructing Attack Graphs is a crucial part of vulnerability analysis for a network. Since constructing Attack Graphs by hand is tedious and error-prone, they suggest an automated method for generating and analyzing Attack Graphs. Gupta et al. [58] found the informal attack graph modeling approach useful for iterative system design. In the modeling approach in [138], the system’s purpose, potential attacker goals, the path to successful attack, and trust boundaries of the system are modeled and documented.

CORAS framework [29] provides a modeling approach, inspired by UML, for expressing assets, risks that target the assets, vulnerabilities, and security solutions. The modeling notation has been improved over time, and [17] introduces new elements for expressing unwanted incidents, risks, accidental and deliberate threats. The threat models relate a threat to vulnerabilities and risks. Risks are related to unwanted incidents, and incidents are connected to the target assets. Finally, treatments can be related to vulnerabilities and risks.

In a separate section, we extensively survey modeling notation for expressing the concept of vulnerabilities, since vulnerability analysis is central to the framework in this thesis.

2.1.2 White Side: Security Requirements and Countermeasures

Modeling

Another major group of conceptual modeling notations for security focuses on expressing protection mechanisms against malicious behavior. Although attacks are considered in many of these conceptual modeling languages, these types of contributions mainly focus on modeling security relevant information, security countermeasures, and security requirements. Security requirements modeling and analysis frameworks are separately studied later, and this section focuses on conceptual modeling notations only.

UMLsec [81] is an extension to UML that allows expressing security relevant information within UML diagrams. The main uses of such approaches are first, to encapsulate secure design knowledge and make it available to developers in the form of a widely-used design notation, and secondly, to provide formal evaluation to check if the constraints associated with the UMLsec stereotypes are fulfilled in a given specification. The extensions are suggested in the form of
UML stereotypes, tags, and constraints that can be used in various UML diagrams such as activity diagrams, statecharts, and sequence diagrams. The stereotypes and tags encapsulate the knowledge of recurring security requirements of distributed object-oriented systems, such as secrecy, fair exchange, and secure communication link. By assigning a stereotype or tag to part of a model and retrieving the threats, the behavior of the subsystem can be analyzed to check the impact of the threat and the security of the system with the presence of the threat. UMLsec focuses on the need for joint execution of the system with the presence of attacks and countermeasures to check if the security mechanisms successfully protect the system against the attacks.

SecureUML [99] is another UML-based modeling language for the model-driven development of secure distributed systems. SecureUML takes advantage of Role-Based Access Control (RBAC) for specifying authorization constraints by defining a vocabulary for annotating UML-based models with information relevant to access control.

Mouratidis et al. [114] introduce extensions to Tropos methodology [53] for incorporating security concerns into the agent-oriented development process. In this approach, security constraints, secure dependencies, threats, and security goals, tasks, and resources are introduced and added to the Tropos modeling notation. To model these concepts, the Tropos modeling notation [19] is extended. In this approach, secure entities are tagged with an “s” to indicate those tasks, goals, and softgoals are security-related. Security requirements are dealt with as constraints on the functionalities. The assignment of a security constraint to a goal is indicated using a constraint link with a restricts tag. Threats represent circumstances that have the potential to cause loss or problems that put the security features in danger. Secure Tropos provides an additional modeling construct to the i* notation for representing threats. In a similar work in [113], the Secure Tropos framework is extended for modeling security capabilities.

Matulevicius et al. [105] adapt the Secure Tropos proposal for security risk management. In this enhanced framework, the i* beliefs are used for modeling vulnerabilities. In this way, vulnerabilities are treated as attackers’ assumptions and beliefs about system weaknesses. Vulnerabilities are located inside the boundary of actors and the positive impacts of vulnerabilities are related to the attack model.
2.1.3 Limitations of Existing Modeling Approaches

In this section, limitations and capabilities of the methods are studied and compared. Several conceptual modeling approaches such as [135, 144, 107, 124] employ tree style or graph-based constructs for modeling attacks. Graphs are useful for expressing the attack steps as nodes and events as edges. Trees are useful for decomposing the attack progress, causes, and events. However, Attack Trees do not provide constructs to express required resources, access level, skills required to perform an attack, or consequences of successful attacks.

Although Fault Trees enable modeling faults and the ability to trace them to leaf events or errors, it does not provide the means to express which vulnerabilities of the system lead to faults and failures. Although “minimum path-sets” are proposed for modeling ways to prevent a failure, the impact of the countermeasures on other events, faults, vulnerabilities, and attacks, cannot be expressed explicitly. The Attack Nets approach does not support countermeasure modeling and analysis. In addition, Attack Nets does not consider modeling and analyzing the concept of vulnerability, and the relationship between attacks and vulnerabilities. Although Attack Graphs are able to model the steps of an attack, post and pre conditions, required configurations, and capabilities, they do not provide a way to express the impact of the attacks on system functionalities.

Abuse case and misuse case modeling are two examples of security modeling from the point of view of system functionalities. However, the abuse case model is not semantically related to the use case model; therefore, the abuse case model does not provide the means to analyze the impact of an abuse case on other use cases. Actors and malicious actors could be the same entities or different, but the abuse case approach does not differentiate them.

Use case modeling in general is very limited. The decomposition mechanisms and causality relations cannot be expressed as part of the use cases’ conceptual models. This limits reasoning on the attack description and attacks consequences, when use case-based models are used for security analysis. For example, misuse case models do not express vulnerable elements and misuse cases as an exploitation of vulnerabilities, or why a misuser attacks the system, what is the impact of a security use case and a misuse case on other use cases.
Using the inverted strategic actors in [98], one cannot explicitly model the impacts of countermeasures on attackers’ goals or other goals, because the countermeasures are placed inside the boundary of the victim actor and the countermeasure analysis does not consider which player needs to employ the countermeasures.

UMLsec defines a set of stereotypes and tags that encapsulate security design knowledge to be used as part of UML diagrams. However, UML is not a requirements engineering notation, and the only diagram that focuses on the expected functionalities from the users’ point of view is the use case diagram. The formal semantic defined for use case models in UMLsec is limited to a few stereotypes. The resulting models do not express attackers’ behavior, and threat description is limited to using the notion of Delete, Read, Insert stereotypes to change a state of the subsystem. Therefore, the usefulness of the modeling constructs is based on the expressiveness and comprehensiveness of the stereotypes. SecureUML can be used in the context of a model-driven software development process to generate access control infrastructures. However, the meta model of the SecureUML notation is limited to the access control concepts.

In Secure Tropos [114], security related entities are tagged with an “s”. However, distinguishing security entities from other system entities does not affect the result of analysis on the models. The notion of threat introduced in Secure Tropos is limited to a visual representation. Threats are not assigned to an attacker, and the models do not express goals and refinements of the attacks. In the enhancement of Secure Tropos notation in [105], threats are linked to the source actor but vulnerabilities are not related to the elements that bring them to the system. While impacts of vulnerabilities on attacks are expressed in the models, impacts of countermeasures on vulnerabilities are not captured.

## 2.2 Vulnerability Modeling and Analysis Approaches

This section surveys and compares different approaches to modeling, organizing, and analyzing security vulnerabilities in requirements engineering frameworks. We also discuss the types of reasoning about vulnerabilities that existing conceptual frameworks support.
2.2.1 Vulnerability Catalogs

The most primitive way of modeling and organizing vulnerabilities is grouping flaws and weaknesses into catalogs such as [1, 27, 117, 121]. Although catalogs are not conceptual models, they are not entirely structure-less. Various web-based software vulnerability knowledge bases provide searchable lists of vulnerabilities. Catalogs of vulnerabilities contain different types of information with different information granularity, which are useful for specific stages of the development.

These web portals provide a shared and standard way for identifying, specifying, and measuring software weaknesses and vulnerabilities. Their aim is to increase the level of awareness about vulnerable products and the severity of vulnerabilities. For example, the National Vulnerability Database [1], SANS top-20 annual security risks [1], and Common Weakness Enumeration (CWE) [27] provide updated lists of vulnerabilities and weaknesses. CVE contains vendor-, platform- and product-specific vulnerabilities. Such technology- and system-oriented vulnerabilities are not useful to decide on security requirements in the early stages of the development where target platform and technology is not yet decided. The SANS list and CWE catalog include more abstract weaknesses, errors, and vulnerabilities. Some entries in these lists are technology and platform independent, while some of the vulnerabilities are described for specific product, platform, and programming language.

2.2.2 Modeling Vulnerabilities for Security Requirements Engineering

In secure software engineering frameworks, vulnerabilities usually refer to the general openness to attacks and risks. For example, Liu et al. [98] propose a vulnerability analysis approach for eliciting security requirements, where vulnerabilities are the weak dependencies that may jeopardize the goals of depender actors in the network of social and organizational actors.

Only a few security software engineering approaches analyze vulnerabilities, as flaws and weaknesses of the system, for eliciting security requirements. For example, Matulevicius et al. [105] treat vulnerabilities as beliefs in the knowledge base of attackers which may contribute
to the success of an attack. In [106], the $i^*$ framework is extended to represent vulnerabilities and their relationship with threats and other elements of the $i^*$ models.

In the CORAS UML profile [29], UML stereotypes and rules are defined to express assets, risks that target the assets, vulnerabilities, accidental and deliberate threats, and security solutions. CORAS provides a way for expressing how one vulnerability leads to another vulnerability and how a vulnerability or combination of vulnerabilities lead to a threat. CORAS also provides the means to relate treatments to threats and vulnerabilities.

In [142], threat trees are built systematically through anti-goal refinement until leaf nodes are derived that can be software vulnerabilities observable by the attacker. These vulnerabilities are assigned to the attackee. Solutions are anticipated by protecting against the vulnerabilities or avoiding them.

Rostad [130] suggests extending the misuse case notation for including vulnerabilities into requirements models. Vulnerabilities are defined as a weakness that may be exploited by misuse cases. Vulnerabilities are expressed as a type of use case, with an exploit relationship from the misuse case to the vulnerability and an include relation with the use case that introduces the vulnerability.

2.2.3 Comparison of the Vulnerability Modeling Notations

The missing point in the surveyed approaches is the lack of modeling constructs that express how vulnerabilities enter into the system and how they spread out. The link between attacks and vulnerabilities are implicitly and explicitly modeled in all of the surveyed approaches. However, among the modeling notations that provide explicit constructs for modeling vulnerability, only a few frameworks such as CORAS [29], $i^*$ security extensions [39, 33], and extensions of misuse case models [130] relate the countermeasures to vulnerabilities. The semantics of the countermeasure impact in [29, 130] is not well defined, and the model cannot be used to evaluate the impact of countermeasures on the overall security of the system. Although modeling and analyzing the steps and the order of actions to accomplish an attack may affect the countermeasure selection and development, the existing frameworks for security requirements engineering do not incorporate the concept of sequence (temporal order) into their meta-model.
Chapter 2. Background and Related Work

2.3 Security Requirements Frameworks

This section surveys existing frameworks for eliciting, modeling, and analyzing security requirements. The goal of this survey is to investigate the main ways for identifying security requirements. We study the role of conceptual modeling notations in elicitation and analysis of security requirements. We also study how different approaches for deriving and expressing security requirements result in different ways to express security requirements.

2.3.1 Agent and Goal-Oriented Requirements Frameworks

Several security requirements frameworks tackle security by goal-orientated analysis of systems and requirements. Lamsweerde [142] proposes a goal-oriented framework for generating and resolving obstacles to goal satisfaction. In this framework, two models are developed iteratively and concurrently: first, the model of the system-to-be, and secondly, the anti-model. Anti-model includes anti-goals, which are the attackers’ goals and malicious obstacles to security goals, set up by the attackers to threaten security goals. Anti-goals are refined to form a threat tree, in which the leaf nodes are either software vulnerabilities or anti-requirements. New security requirements are then obtained as countermeasures against vulnerabilities.

Liu et al. [97] explicitly model relationships among strategic actors in order to elicit and analyze security requirements. In this approach, first generic role dependency patterns between actors in the domain are identified. This model can be elaborated to express whether the roles in the dependency relationship have trust, security, or privacy expectations from other roles. Then the roles that attackers can potentially play are considered and attacks to the system are modeled as negative contributions to dependency link. The agent-oriented analysis continues with elaborating the role-agent hierarchy and modeling the dependency derivation. This helps find out what dependencies are inherited to the attacker. In this approach, security requirements are originated from specifying security goals that actors expect from each other.

In a similar approach, Liu et al. [98] analyze attackers, countermeasures, access control, and vulnerabilities in actors dependency networks. In this work, ordinary actors are assumed to be potential attackers as well, and thus, they inherit the capabilities, intentions, and social
relationships of the corresponding legitimate actor. The underlying idea in this work is that dependency relationships bring vulnerabilities to the system and the depending actor. The dependency vulnerability analysis aims to find which dependency relationship is vulnerable to the attacks. In this regard, the actors in the basic dependency model are substituted with their corresponding attacker, and then the impact of the attack to the dependency relationship is traced to the network of actors.

The contribution in [51] introduces Secure Tropos\(^1\), a formal framework for modeling and analyzing security requirements based on the concept of trust, ownership, permission, and delegation. In this framework, ownership is defined as the relationship between an actor and a service if an agent is the legitimate owner of the service. Trust among two actors and a service means that actor A trusts the actor B to fulfil a service S or achieve goal G. Delegation among two actors and a service means that actor A explicitly delegates a goal, execution of a task, or access to a resource to actor B. These concepts are dealt with in the normal functional requirements model to check whether certain security requirements would be satisfied with the presence of such properties (more details are discussed in Section 2.3.2).

We described the Secure Tropos modeling notation [114] in the previous section. In [113], security concerns are integrated into all phases of Tropos agent-oriented methodology: from early and late requirements, to architecture and detailed design. In the early requirements phase, a Security Diagram is constructed and security constraints are imposed on the stakeholders. During the late requirements stage, security constraints are imposed on the system-to-be in the Security Diagram. The system is presented as one or more new actors, who have a number of dependencies with the other actors of the organization. In the architectural design stage, the security capabilities, security constraints, and secure entities that the new actors introduce are identified and assigned to each agent of the system. At the detailed design stage, agent capabilities and interactions are specified.

In a related work to the Secure Tropos requirements framework, Bresciani et al. [20] suggest a quantitative security requirements analysis on the security constraints models. The

\(^1\)In security requirements literature, two different frameworks developed by different researchers are called Secure Tropos [114, 51].
analysis considers measures of security criticality and complexity. The goal of analysis is to make sure that security bottlenecks are identified and an actor is not overloaded with security responsibilities.

### 2.3.2 Trust-Based Requirements Frameworks

Viega et al. [145] argue that trust and trustworthiness are foundations of security, and the basis of trust relationships and trust formation can dramatically affect the underlying security of any system. They assert a trust relationship between the entities must be formalized and mapped into the system requirements for implementation. Without recognizing all the entities and their trust relationships in a software system during the requirements phase of a project, that project is doomed from the start. They conclude that it is very important to minimize or eliminate the trust assumption between various components in a multi-party system. Therefore, a recent shift in security requirements engineering is toward analyzing trust relationships and trust assumptions. Approaches such as [63, 62, 61, 60, 51] analyze effects of trust relationships and assumptions about trust on security requirements.

The work in [62, 61] focus on analyzing the trust assumptions in the context of problem frames [77] for deriving security requirements. The focal point of these contributions is the argument that trust assumptions can have fundamental impacts on how the system is perceived [62, 61]. In this regard, trust is defined as the quantified belief by a trustor with respect to the competence, honesty, security and dependability of a trustee within a specified context. By considering trust assumptions, a requirements engineer believes and accepts that the domain holds certain properties and specifications. To incorporate the trust assumption into the problem frames model, which is used as the requirements elaboration modeling technique, the problem frames notation is extended with an arc from the domain to a dotted oval describing the properties assumed to be true [61]. By adding trust assumptions, the model works as a documentation of ways that the requirements engineer trusts the behavior of the domain.

Based on the notion of trust assumptions, Haley et al. [60] propose a security requirement engineering framework for expressing and analyzing security requirements as constraints on functional requirements. The security requirements process in this work starts with identifying
Chapter 2. Background and Related Work

functional requirements. Then, security goals and security requirements are identified. Finally, satisfaction arguments are constructed. The result of this framework is a set of requirements core artifacts [111] that mainly document application and security goals, assets, management control principles, functionality, quality, security requirements, and system architecture. Security goals are operationalized into security requirements, aiming to protect the assets from harm. Security requirements are treated as constraints on the systems’ functional requirements.

Giorgini et al. [51] also consider trust, ownership, permission, and delegation. However, this approach mainly focuses on eliciting access control requirements based on analyzing ownership, permission, and delegation relationships in a trust chain of actors.

2.3.3 UML-Based Requirements Frameworks

UML is a widely used notation in the analysis and design of software systems. Developers already know UML modeling notation; therefore, UML and its extensions have also been adapted for security requirements engineering [140, 108, 81].

For example, UMLsec [81] is an extension to UML that allows expressing security relevant information within UML diagrams. In UMLsec, security requirements are defined by assigning security stereotypes elements of design models. Since UML is not a requirements engineering notation and the only diagram that focuses on the expected functionalities of the system from the users’ point of view is the use case diagram, several security requirements engineering methods based on use cases are proposed.

Jurjens [82] combines the use of UMLsec modeling, use-case driven process, and goal trees to design the system, along with modeling functional and non-functional requirements respectively. In this method, the goal tree is developed to record the result or reasons for design actions in the UMLsec diagrams. Security goals can be refined by adding more system details, such as UMLsec stereotypes or tag-values.

We described the misuse case and abuse case modeling notations in previous sections. The process of eliciting security requirements by misuse cases [140] starts with identifying critical assets. Then security requirements for each asset are defined. In the third step, threats to each security requirement are defined and expressed as misuse cases. In the fourth step, risks
are identified and analyzed. Finally, security requirements are defined as either security use cases or in the mitigation field of misuse case description. Sindre and Opdahl [140] assert that the visualization of links between use cases and misuse cases helps organize the requirements specification and trace requirements to the threats that motivated them. They also propose templates for specifying misuse case details to support requirements extraction [139].

Firesmith [47] also employs the notion of use cases and misuse cases for analyzing security requirements. Firesmith claims use cases are typically misused to unnecessarily specify security architectural mechanisms instead of security requirements. Therefore, he suggests a template for describing reusable security use cases for specifying security requirements [47].

Another example of UML profiles for security analysis is the CORAS framework [30]. CORAS, inspired by UML diagrams, use use-case similar models for expressing assets, risks, vulnerabilities and security treatment. General UML diagrams are also used in the CORAS methodology for modeling the target system context [17].

2.4 Security Risk Assessment Methods

Risk management methods are frequently used as part of the secure software development life cycle [57, 59]. Several methods such as Security Attributes Evaluations Method (SAEM) [22], CORAS UML profile and methodology [69, 29, 17], extensions to Tropos for risk modeling [9], improvements on Secure Tropos for risk assessment by Matulevicius et al. [105], and the risk-based security requirements engineering framework [106] offer modeling or analysis approaches for security risks assessment.

SAEM [22] is a cost-benefit analysis method for comparing alternative security designs. SAEM relies on a quantitative risk and benefit assessment based on assessment of security technologies benefits, analyzing the threat index before and after applying security technologies, and comparison of alternative technologies. The benefit of a security technology is that it assesses how well the technology mitigates a risk. Analysts capture a rough estimation about the quality and effectiveness of countermeasures. The results of the technologies’ effectiveness are a percentage indicating how a threat is reduced by a security technology. The benefit
assessment indicates how the overall threat index is affected. A threat index indicates the frequency of an attack and its probable outcomes. In addition to considering the total changes in the threat index, alternative technologies are compared based on the coverage of prevention, detection, recovery, coverage of threats, and cost of security technologies.

Asnar et al. [9] also introduce a method for modeling and analyzing risks at the organizational level by extending Tropos methodology to a Goal-Risk Model. The Goal-Risk Model provides the basis to analyze if the requirements are satisfied and the risk level is acceptable for every actor. In this approach, risk assessment involves analyzing evidence of satisfaction or denial of tasks and goals, the costs of events and tasks, utility values of higher goals, the likelihood of local and global events, and the risk tolerance for each actor.

The risk-based security requirements engineering framework proposed by Mayer et al. [106] is concerned with integrating requirements engineering practices and security engineering, as well as intertwining requirements and architecture design. The main idea is to align IT security with business goals. To this end, impacts of risks on business assets are analyzed, risks are related to the threats and vulnerabilities in the architecture, and security requirements are identified to mitigate the risks. IT risks are further decomposed into three components: \( \text{Risk} = \text{Threat} \times \text{Vulnerability} \times \text{Impact} \). In this way, risk is characterized by the opportunity of exploiting one or multiple vulnerabilities, from one or many entities, by a threatening element using an attack, causing an impact on business assets.

We described the CORAS modeling language in the previous section. In addition to the UML-inspired modeling notation, CORAS provides a methodology based on the unified process for conducting security analysis [29]. The CORAS risk assessment method is adapted to address requirement elicitation. In [17], a seven-step risk analysis method based on the CORAS modeling approach is presented. The analysis method consists of analyzing the target context by developing UML and asset models. Then, potential attackers, vulnerabilities, and target assets are identified. The risks that have severe consequences and high likelihoods are selected for further analysis. To perform detailed risk analysis, threat scenario diagrams are developed and unwanted incidents are identified. Finally, treatments are selected for the risks that are not acceptable.
Evans et al. [43] developed Mission Oriented Risk and Design Analysis (Morda) as a methodology for analyzing security risks for designing functional and secure networks. Morda is based on Multiple Objective Decision Analysis (MODA), and combines threats, attacks, and mission impact concepts for deriving an unbiased risk metric. In this method, relevant missions and the impact of an adversary on them are identified. To analyze the adversary, the objectives of attacks are identified and an attack tree is developed. Morda suggests calculating an attack score to assess the risks. The combination of attack scores represents the adversary’s preferences for a portfolio of attacks against the system’s missions. This parameterized attack portfolio defines the overall system risk in terms of calculated utility scores. Finally, security engineers derive countermeasure alternatives, first by focusing on the highest-scoring attacks and characterizing potential countermeasures according to their costs and benefits. In [42], Evans and Wallner argue that the attack score provides the metric necessary for making trade-off decisions, and design decisions can be made by comparing the changes in risk, cost, and performance parameters.

The Operationally Critical Threat, Asset, and Vulnerability Evaluation (OCTAVE) [4] is a risk management methodology that was developed at Carnegie Mellon University’s Software Engineering Institute (SEI) in collaboration with CERT. OCTAVE focuses more on organizational risk, and only slightly on technical risks. The OCTAVE framework provides guidelines for systematically evaluating information security risks. In the OCTAVE analysis process, the asset-based threat profiles are created first. Then infrastructure vulnerabilities are identified, and finally, security strategy and plans are developed. OCTAVE may be viewed as a heavyweight method because OCTAVE guidelines consist of 18 volumes of large and complex worksheets and practices to implement.

The Microsoft Security Development Lifecycle (SDL) [57] is a software development process used and proposed by Microsoft to bake security into the development process. Threat Modeling is a core element of the SDL. The Microsoft Threat Modeling Process has five steps: 1) Identify Security Objectives, 2) Survey the Application, 3) Decompose it, 4) Identify Threats, and 5) Identify Vulnerabilities. STRIDE is a classification scheme for characterizing known threats, and stands for Spoofing, Tampering, Reputation, Information Disclosure, Denial of Service, and
Elevation of Escalation. Microsoft supports threat modeling at the design phase, with a tool for modeling data stores, processes, and data flows, which also helps organize threats against the data assets being specified in the model. The tool enables analysts to document consequences of threats and solutions, and the threats are ultimately translated to a bug report in a bug tracking system.

2.4.1 Limitations of Current Risk Assessment Methods

A major limitation of traditional risk analysis methods is disregarding the quality requirements as trade-off parameters. These methods focus on financial costs and security gains and losses for the cost benefit analysis. These methods may result in selecting a solution that only minimizes the costs, or maximizes the security, or in the best case, balances out security and financial costs, while leaving other factors out.

Security requirements are tightly coupled with other functional and non-functional requirements, and making security trade-offs needs consideration of not only costs and security risks, but also the competing goals of multiple stakeholders. Many current risk assessment methods are independent of security requirements and design analysis. Methods that provide guidelines and framework to intertwine risk assessment with the rest of the development life cycle tend to have heavy-weight processes. For example, due to the complexity of the OCTAVE processes, OWASP does not anticipate that OCTAVE will be used at large by application designers or developers, especially because it fails to take threat modeling into consideration. SDL is a light weight process that also provides threat modeling tools, but the risk analysis is based on data flow analysis. Analyzing threats based on the flow of data limits the analysts’ point of view to the types of risks they consider. For example, Denial of Service (DoS) and reverse engineering attacks are example of threats that data flow analysis would be able to address.

The other problematic side of risk assessment is that security risk analysis methods most often rely on availability of risk parameters’ values i.e. expected loss (product of loss with its respective probability), expected severe loss, the standard deviation of loss [15], probability and motivations behind attacks. However, the Microsoft Risk Management Guideline [110] concludes that there is no formal and rigorous way to effectively calculate values for risks
parameters and countermeasure effects. In other words, while they may appear to provide more detail, the financial and numerical values actually obscure the fact that the numbers are based on estimates. For example, how can one precisely and accurately calculate the impact that a highly public security incident might have on the different goals of stakeholders? If it is available, historical data can be examined, but quite often such data is not available for organization-specific cases.

The same guideline ([110]) asserts that “organizations that have tried to meticulously apply all aspects of quantitative risk management have found the process to be extremely costly, and projects usually take a very long time to complete the first cycle, and they usually involve a lot of staff members arguing over the details of how specific fiscal values were calculated”.

2.5 Multi-Criteria Decision Analysis Methods

Multi-criteria decision analysis (MCDA) is defined as an umbrella term to describe a collection of formal approaches for taking account of multiple criteria and helping decision makers explore decisions when intuitive gut-feel decision making is not satisfactory [14]. Three concepts usually play a fundamental role for analyzing and structuring the decision aiding process: 1) alternative, or more generally potential action, 2) family of criteria, and 3) a problem formulation. Preference modeling and problem structuring are also main concerns in MDCA. In this section, we briefly overview some of MDCA methods for evaluating alternatives and modeling the preferences.

Utility Theory

Multi Attribute Utility Theory (MAUT) by Keeney and Raiffa [89] emphasizes the use of multi attribute preference models. Preference theory studies the fundamental aspects of individual choice behavior, such as how to identify and quantify an individual’s preferences over a set of alternatives, and how to construct appropriate preference representation functions for decision making [45]. Preference representation function under certainty is known as a value function, and preference representation function under risk is referred to as a utility function. For example, utility values are used for representing the quality value of non-functional
requirements in [148]. In Multi-Attribute Value Theory, the weight of a criteria \(w(i)\) and the
value that criteria \(i\) provides, \(v(i)\), conjointly calculate the total utility value:

\[
u = \sum_i w(i)v(i)\]

In MAUT, evaluating an alternative like \(j\), mainly involves assigning a utility weight, \(v(j)\),
using real-valued functions in the criteria of comparison and calculating an expected value for
each alternative, considering the probability, \(p(j)\) of having \(v(j)\):

\[
u = \sum_j p(j)v(j)\]

**Analytic Hierarchy Process**

The Analytic Hierarchy Process (AHP) [131] and its generalization, the Analytic Network
Process (ANP), are theories of relative measurement of intangible criteria. In this approach to
relative measurement, a scale of priorities is derived from pair wise comparison measurements
of elements. In the AHP, paired comparisons are made with judgments using numerical values
taken from the AHP absolute fundamental scale of 1-9, where 1 indicates that two elements
are of equal value and 9 indicates favoring one element over another with the highest order of
affirmation.

The choice of AHP fundamental scale is explained by Weber’s law about the change in
sensation and stimulus [132]. In a less mathematical way, Satty justifies the scale of 1 to 9 of
AHP based on humans’ ability to distinguish between high, medium, and low at one level, and
for each of them in a second level below. An absolute number of 1 is assigned to \((low, low)\) and
9 is assigned to \((high, high)\).

AHP has been applied to requirements trade-off problems: Karlsson et al. [86] apply AHP
for prioritizing software requirements and a cost-value analysis. A case study of applying AHP
for software requirements prioritizing is given by Karlsson in [85].

**Outranking Methods**

Outranking methods model preferences by using a binary outranking relation, \(S\), which
indicates if action \(a\) is at least as good as action \(b\), then \(aSb\). The construction of an outranking
relation is based on two major concepts: 1) Concordance: for an outranking, \(aSb\) to be validated,
a sufficient majority of criteria should be in favor of this assertion, 2) Non-discordance: when the concordance condition holds, none of the criteria in the minority should oppose too strongly to the assertion \( a \succ b \).

In outranking methods, the decision maker deals with a set of alternatives and a set of criteria to assess the alternatives. A real number indicates how good each alternative is with respect to the criteria. An importance coefficients weight is also determined to reflect the voting power of the criterion when it contributes to the majority that is in favor of an outranking.

The family of ELECTRE methods [45] are examples of outranking methods that calculate a *concordance* and *discordance index* for all pairs of alternatives to exploit the outranking relations and identify a small subset of actions, from which the best compromise action could be selected.

**MACBETH**

Measuring Attractiveness by a Categorical Based Evaluation Technique (MACBETH) [12] is a multicriteria decision analysis approach that requires only qualitative judgments about differences of value to help quantify the relative attractiveness of options. Using MACBETH, decision analysts are asked to provide preferential information by pair-wise comparison of elements, which results in an ordinal scale of the elements preferences. The difference of attractiveness between the elements are asked in form of “very weak”, “weak”, “moderate”, “strong”, “very strong” or “extreme” labels. To build an interval (value) scale, the qualitative judgements of difference of attractiveness need to be represented by non-overlapping (disjoint) intervals of real numbers. The basic idea underlying MACBETH is that the limits of these intervals should not be arbitrarily fixed, but determined simultaneously with numerical value scores for the elements. The basic MACBETH scale can be obtained from solving a linear program.

**Even Swaps**

Even Swaps [64] is a preference elicitation method based on value trade-offs and eliminating criteria. The decision analyst collaborating with stakeholders changes the consequence of an alternative on one attribute, and compensates this change with a preferentially equal change in the consequence of another attribute. This creates a new *virtual alternative* with revised
consequences [116]. The virtual alternative is as preferred as the initial one, and it can be used as a surrogate. Now, one of the attributes is irrelevant, i.e., the consequences of all the alternatives on this attribute are indifferent. This irrelevant attribute can be eliminated from the decision analysis process to reduce the problem size.

**Fuzzy Set Theory**

The classic multi attribute decision making methods generally assume that all criteria and their respective weights are expressed in crisp values. In a real-world decision situation, the criteria may contain imprecision or vagueness inherent in the information, which causes a serious practical constraint. In general, the term “fuzzy” commonly refers to a situation in which the attribute or goal cannot be defined crisply, because of the absence of well-defined boundaries of the set of observations to which the description applies [83].

The general first phase of fuzzy MCDA deals with the measurement of performance ratings or the degree of satisfaction with respect to all attributes of each alternative. In the second phase, alternatives are ranked and compared according to the resulting aggregated performance ratings obtained from the first phase.

**Qualitative Reasoning:** Qualitative Reasoning (QR) is an area of research, mainly in artificial intelligence that automates reasoning and problem solving about the physical world, when precise mathematical representation of the physical phenomena and behavior is not available [18]. The ultimate goal of QR is to enable the prediction of the physical world behavior without formal formulation and the numerical simulation of variables that describe the domain of interest [76]. The underlying justification for this area of study is observing that experts in most engineering fields routinely reason with much less information than would be required to do detailed quantitative analysis [76].

QR is concerned with representing the imprecise knowledge and inference techniques using such knowledge to predict the physical system behavior. The main knowledge items needed to do QR are causal relations which are essential elements of human understanding of all phenomena [76]. Thus representing imprecise knowledge is concerned with model formulation and abstraction. The model of the physical system needs to include relevant information at the
right level of abstraction, enforced with relevant qualitative arithmetic.

This area of research is indirectly related to the requirements decision analysis, in the sense that in analyzing trade-offs between requirements, when dealing with multiple alternative solutions, our ultimate goal is to predict the results of applying each solution on requirements. These predictions would be the basis of decision making, and thus, QR can help reason about the system’s behavior when numerical data is not available.

2.5.1 Requirements Decision Analysis

Requirements trade-off analysis is the systematic examination of advantages and disadvantages of requirements as well as the design choices for a system to achieve the right balance among several competing goals [5]. Some approaches analyze trade-offs explicitly. For example, the notion of misuse cases [5] has been used to visualize the structure of the trade-off situation in a way that emphasizes the essential points of conflicts and trade-offs.

Architecture Trade-off Analysis Method (ATAM) [13] explicitly addresses design decision trade-offs. ATAM is used to evaluate whether an architecture decision satisfies particular quality goals. ATAM helps designers prioritize scenarios and evaluate alternative tactics. However, ATAM is a labor-intensive analysis method which relies on the subjective opinion of domain experts. CBAM [13] takes the results produced by ATAM as input and adds cost factors into the trade-off analysis.

Security Verification and security solution Design Trade-off analysis (SVDT) [68] specifically addresses security solution design trade-off and risk assessment, by using Bayesian Belief Nets to structure the trade-off problem. SVDT uses a fixed set of trade-off parameters (instead of user-defined) in the topology of the Bayesian Network, and thus, the fixed topology limits the decision criteria that can be analyzed.

Requirements decision methods, such as [44, 73, 101, 86, 21, 9, 84, 95] rely on the availability, accuracy, and meaningfulness of quantitative costs and benefits measures. Feather et al. [44] propose a quantitative model for strategic decision analysis and trade-off analysis, by the “coarse quantification” of relevant risk factors and their interactions. Bresciani et al. [21] suggest a quantitative security requirements analysis on the security constraints models developed using
Secure Tropos method. In this method, to calculate the security criticality of the system, a value within the range of 1-5 is assigned to each security constraint in a dependency relationship. The goal of this method is to reassign security constraints to different actors to ultimately reduce the complexity and criticality of the overloaded actors. In [84], attribute values, such as contribution values and preference matrices, are added to goal graphs to choose and adopt a goal from the alternatives and to recognize the conflicts among the goals.

The preference model-driven service selection method in [101] enhances the i* Framework with quantitative preference information based on input from domain experts and users into the AHP preference elicitation process. The contributions of alternative actions are calculated by using Weighted Net-Value Statistics [148]. The main idea of Weighted Net-Value Statistics is that the contribution of solutions is a qualitative indicator that cannot be computed, and can only be estimated by people who are familiar with the system design. This method collects estimations of multiple experts in the form of a range. The method also gathers the confidence of experts in their estimations. It calculates the variance and standard deviation of all evaluations and also unifies the evaluations into one single number for each alternative solution.

The multi-criteria preference analysis method in [73] is designed for requirements negotiation, based on the win-win model [16]. In this method, stakeholders express their win conditions, and the negotiation helps balance win conditions of multiple stakeholders. In this method, each individual stakeholder assigns a score to alternatives according to some decision criteria. The relative criteria weights are extracted by using the SMART method [146], AHP, and geometric progression. Both methods in [73, 101] ultimately calculate a utility value for each solution to select the best option.

Letier and Lamsweerde [95] argue that due to the lack of accuracy and measurability of goal formulations, and the lack of impact propagation rules through goal models, domain-specific quality variables are needed to reason on partial goal satisfaction. In the proposed work in [95], goals are specified in a precise probabilistic way, and the impacts of alternative decisions on the degree of goal satisfaction are analyzed numerically.

Techne [80] is a requirements modeling language that supports representation and reasoning about requirements and alternative solutions. This language enables expressing desired condi-
tions as goals, softgoals, and constraints, as well as domain assumptions, such as optionality and conflicts of goals and preferences of stakeholders over the goals. A solution in Techne is a combination of tasks and domain assumptions that satisfy all compulsory goals and quality, also aiming to satisfy as many preferred optional goals as possible. Techne reasoning is based on comparing solutions identified via simulation using propositional logic. In this reasoning method, candidate solutions are generated and compared to finally select one solution.

Aguilar et al. [3] propose a goal-oriented approach for optimizing non-functional requirements based on finding Pareto front configurations that maximize softgoals satisfactions. In this approach, the value of $i^*$ Help contribution is $+1$, Some+ is equal to $+2$, and Make is $+4$, and the negative impacts of Hurt, Some-, and Break are $-1$, $-2$, and $-4$ respectively. Based on these values, the algorithm finds a set of configurations that dominate at least one other configuration (they are called Pareto front configurations). One configuration dominates another if all of its contributions are at least equal or better than the other. Having the Pareto front configurations, then stakeholders prioritize softgoals in a list to help find a configuration among Pareto fronts that maximize the softgoals. However, the proposed method in [3] does not provide an explicit and systematic method to actually aid the decision according to the priorities.

2.5.2 Challenges in (Security) Requirements Decisions

Making trade-offs among requirements and selecting a solution among multiple security alternatives is challenging. As pointed out in the studies on psychology of security [136, 137, 49], security trade-offs are naturally subjective and hard to make: different stakeholders have different expectations from the software system, as well as different levels of risk tolerance and personal privacy expectations; therefore, they impose different security requirements which can be conflicting with security and other goals of other stakeholders. Ultimately, security decisions are based on personal judgement and feelings of stakeholders and analysts.

We have identified two distinct groups of approaches to (security) requirements trade-off decision problems: quantitative and qualitative methods. In many existing decision analysis methods, measuring the quality of alternative solutions, when the solution is still in the form of
Chapter 2. Background and Related Work

a requirements specification and not a running system, is a challenge. Thus, in early RE phases, analysts estimate many factors; however, these quantitative estimates elicited from stakeholders about their preferences and the utility of solutions are imprecise, uncertain, or ill-defined, and there is the risk of relying on them despite the high levels of estimation involved [45, 119]. In this section, we elaborate the main challenges of these approaches.

Quantitative Approaches

In Chapter 1, we discussed that quantitative approaches can help determine the optimum solution by applying mathematical operations. These mathematical operations are seen as important tools to support objective decision analysis; therefore, many of the requirements decision methods [44, 73, 101, 87, 22] extensively rely on the availability, accuracy, and meaningfulness of numerical estimations of risks, costs, benefits, and the satisfaction level of requirements. For example, Feather et al. [44] propose a quantitative model for strategic decision analysis based on “coarse quantification” of relevant risk factors and their interactions.

However, quantitative approaches require software analysts and stakeholders to have cognitive abilities and the empirical knowledge required to provide accurate and complex quantitative requirements models as well as requirements satisfaction evaluations in the form of absolute measures. The method in [3] works based on assigning numerical values such +1, +2, +4 to Help, Some+, and Make contributions. These values are arbitrary, and later in the decision analysis process, stakeholders can misjudge that a Make contribution is four times stronger than Help, or Some+ is twice as strong as a Help contribution.

Quantitative approaches have other limitations and weaknesses in the context of requirements trade-off analysis:

1. Lack of measurement methods: Measuring software qualities and design objectives is hard, because these factors are often intangible and soft in nature, especially in the early phases of the project. Numerical measures of quality goals and requirements may have no physical interpretation in the application domain [95]. Formal and rigorous ways to effectively calculate the strength of a design solution, in terms of requirements satisfaction levels, are not yet accepted and practical. The fundamental problem is that while financial or other
numerical measures do appear to provide more details, they can actually obscure the fact that the results are based on estimates [110]. Even if some requirements can be refined into measurable variables, time and budget limitations preclude elaborate methods for obtaining quantitative data in the early stages of software development.

2. **Lack of unified measurement scales**: In general, scales of measurements for software qualities are not available or agreed. Stakeholders may choose to evaluate some requirements numerically, and some requirements qualitatively, on different scales, e.g., absolute values, percentages, ordinal numbers, or qualitative labels. For example, cost can be expressed by absolute measures, while privacy may be evaluated only by ordinal values. When different requirements are measured (or estimated) on different scales, normalizing inconsistent types of measures into a single scale is troublesome and may not result in a meaningful utility value.

3. **Non-linearity of value functions**: Utility functions calculate a value that indicates the utility of an alternative over a set of goals, by summing up the product of value functions and weights (preference) of the goals. When assessing requirements satisfaction, the value function is not necessarily a linear function of the satisfaction level of requirements. For example, assume the security level of a system is represented by percentages, where 80% indicates the system’s availability and data confidentiality is guaranteed for 80% of the time. By ensuring the security for 80% of time, we cannot necessarily conclude that the utility of the system on security is 0.8 in the scale of 0-1. For example, stakeholders may believe that if security is not guaranteed for more than 95% of the time, the security utility is zero. The security value of 95% could be 0.5, the value for 99% could be 0.9, etc. Hence, a simple linear function cannot necessarily map the security measures to utility values.

4. **Implicit and unintended rankings**: To extract alternatives rankings, stakeholders can be interviewed to elicit numbers that indicate the priority of alternatives, e.g., on the scale of 1 to 10, $A_1$’s rank is 2, $A_2$’s rank is 8, etc. These numbers are not obtained by objective measurements, but instead, decision stakeholders judge about the strength of alternatives by choosing a value between a predetermined lower limit for the worst possible alternative and a predetermined upper limited for the best alternative [100]. Although stakeholders implicitly compare each alternative to a minimum and maximum contribution strength, alternatives are
not explicitly ranked against each other.

5. Incorrect mathematical judgement: A numerical relationship between two quantities is quantitatively meaningful if the relation also holds for the (chosen) quantitative scale and the relation defines a meaningful arithmetical operation over the numbers; for example, it is meaningless to compare the mean of ordinal values [48]. When stakeholders and decision analysts are asked to evaluate the security level of a solution in an ordinal scale (e.g., 1 to 9); they may assume the level of 8 is twice as secure as the level of 4. However, these numerical values are rough estimations in an ordinal scale, and assumptions about their ratio or intervals are mathematically incorrect [48]; thus, the use of utility theory or mathematical operations (e.g., average) over these numbers is not meaningful.

6. Cognitive demands for requirements prioritization: To build a scale of preferences, several numerical and non-numerical techniques (e.g., Outranking methods, MAUT) require decision analysts and stakeholders to produce numerical representations of their strengths of preferences, either directly or indirectly, which is not a natural cognitive task [12]. Requirements prioritization needs complex decision-making, domain knowledge, and estimation skills [94]. Lehtola et al. [94] provide empirical evidence showing that users find it difficult to prioritize requirements and estimate the requirements values. They conclude that prioritization results should be taken more as being indicative than as an ultimate truth.

Qualitative Approaches

Some requirements can be refined into measurable variables. However, many non-functional requirements have a soft nature, which makes them hard, if not impossible, to measure. Such requirements are usually treated qualitatively, for example, as soft goals in i* [150], Tropos [53], and GRL [96]. Goal model evaluation techniques such as [24, 67, 54, 6], enable reasoning about the partial satisfaction of soft goals by propagating qualitative labels such as partially satisfied (\(\vee\)), sufficiently satisfied (\(\surd\)), partially denied (\(\varnothing\)), and fully denied (\(\varnothing\)) through the goal graph. In Techne [80], softgoals’ satisfaction is analysed using propositional logic and thus alternatives cannot be compared according to different degree of softgoals’ satisfaction.

Goal model evaluation techniques such as the ones in [24, 67, 54] enable analyzing require-
ments satisfaction in the absence of numerical data. Goal models can be used to structure the requirements trade-off problems. However, qualitative goal model evaluation techniques have varieties of limitation for making decisions objectively:

1. **Low granularity of evaluation labels:** Goal modeling methods that rely on goal refinement for achieving finer-grained differentiation among alternatives typically employ a minimal set of labels to indicate the satisfaction level of goals. For example, in [24, 67, 54], possible satisfaction degrees are partially satisfied ($\mathcal{A}$), fully satisfied ($\checkmark$), partially denied ($\mathcal{K}$), fully denied ($\mathcal{X}$), conflict (□), and unknown (?). These labels are meant to be used in conjunction with iterative goal refinement to arrive at satisfactory solutions. However, those degrees of satisfaction do not have a precise meaning for decision analysis purposes:

   - Stakeholders may not be able to provide a meaningful and absolute value that measures the contribution of alternatives. The $i^*$ contribution links are limited to make and help (or some help), and break and hurt (or some hurt), while stakeholders may be able to differentiate the strength of such contributions in a more granular scale.

   - Partially satisfied and partially denied evaluation labels [24] bear conceptual redundancies. For example, if confidentiality is partially satisfied, under some circumstances, the confidentiality could be denied, and therefore, it is also partially denied. A fully denied softgoal indicates that the level of satisfaction for the softgoal is zero. In this fashion, currents methods evaluate the goal satisfaction in three levels: zero, partial, or full, which may not be enough for many scenarios.

2. **Unreliability of judgement about conflicts:** The propagation rules in goal model evaluation techniques often result in undetermined satisfaction for higher level goals due to conflicting contributions. In some qualitative evaluation methods [24, 67], when a number of elements have conflicting contributions to a higher goal, human judgement is required to combine the contributions and decide about the evaluation value of the higher goal. This judgement can be cognitively demanding, since it relies on domain knowledge rather than agreed rules, guidelines, and mathematical or logical operations. Hence, the qualitative labels resulting from the goal model evaluation may not be assumed reliable for decision making.
3. **Lack of operations on qualitative labels:** The set of partial and sufficient satisfaction (denial) labels is not helpful for evaluating and comparing alternatives with respect to multiple goals. For example, one cannot tell whether solution $A_1$, which satisfies $G_1$ but denies $G_2$, should be preferred over solution $A_2$ which partially satisfies $G_1$ and partially denies $G_2$. ($\checkmark + \mathcal{X} = \checkmark + \mathcal{X}$)
Chapter 3

A Framework for Security Requirements Trade-off Analysis

In recent years, software companies and government agencies have become particularly aware of security risks that vulnerabilities impose on systems security, and many projects target analyzing and reporting detected vulnerabilities of products and services. For example, various web portals and on-line databases of vulnerabilities are made available to security administrators. Examples of this are the National Vulnerability Database (NVD) [117], SANS top-20 annual security [1], and Common Weakness Enumeration (CWE) [27] all provide updated lists of vulnerabilities and weaknesses. The Common Vulnerability Scoring System (CVSS) [26] also provides a method for evaluating the criticality of vulnerabilities. The Open Web Application Security Project (OWASP) collects the top vulnerabilities annually, assesses the criticality of vulnerabilities, and analyzes possible attack vectors.

In this context, a major problem in achieving security goals in software systems is the overwhelming amount of security-related information and guidelines, variety of tools, and numerous security risks and vulnerabilities that need to be prevented. Software analysts are not security experts and lack the required security training, and thus are often not able to identify relevant security risks. Commonly, security tools focus only on detection of vulnerabilities and security testing, instead of providing systematic ways for prevention of vulnerabilities.
Security experts at Deloitte [79] state that the root-cause behind vast number of similar vulnerabilities in every system is that while many of the organizations have formal enterprise-wide systems development life-cycle (SDLC) processes, none have baked security into those processes. Thus a systematic security requirements engineering framework that intertwines security with the earliest development activities is immediately needed for analyzing vulnerabilities and their risks. A step-by-step process for incorporating security requirements and risk assessment into general requirements engineering activities helps requirements analysts leverage the available empirical security knowledge.

This chapter overviews the proposed framework for incorporating security into the system requirements analysis from early phases of the development. The focus of this framework is on security trade-offs analysis for deciding over alternative security solutions. Figure 3.1 shows an overview of the proposed security requirements engineering framework. This chapter overviews the steps of the process. Detailed description of methods, notations, and algorithms are provided in the later chapters. This section also clarifies the scope of this thesis by also listing problems that we do not address in our framework.

### 3.1 The Modeling Process

The first three steps in the proposed framework focus on modeling system requirements, stakeholders’ goals, and non-functional requirements. The main purpose of the first three steps is to incorporate vulnerabilities into the goal-oriented models of system requirements.

- **Step 1**: Understand stakeholders’ goals and system requirements by developing an $i^*$ model of the system and its main actors.

- **Step 2**: Identify relevant vulnerabilities in empirical knowledge portals and add them to the system model.

- **Step 3**: Identify potential attacks that can exploit the vulnerabilities and analyze the impacts of these attacks on stakeholders’ goals and system requirements.
Chapter 3. A Framework for Security Requirements Trade-off Analysis

Step 1: To identify and assess risks within a software system and its environment, first, goals of stakeholders, system’s functionalities and expected qualities in the target system need to be understood. Thus, the proposed risk assessment and security decision analysis framework starts with a modeling process, using the $i^*$ notation. The $i^*$ Framework was first introduced in [150]. The $i^*$ agent- and goal-oriented models help analysts understand the main actors in the system as well as actors’ dependencies, goals, tasks, and required resources. In Chapter 4, we extensively justify the choice of using $i^*$ as the basis of the modeling activity.

While developing $i^*$ models enables analysts to do powerful types of reasoning, $i^*$ modeling involves some difficulties, that we acknowledge, but we do not tackle them in this thesis directly:

1) $i^*$ models do not scale easily and effectively: the graphical models become large and hard to comprehend quickly. Although adding details to the model helps understand the target system and domain better, at the same time, detailed models are less readable and useful for communication and comprehension.
For example, consider a simple web application that provides users with customized content. The main actors of this scenario are **End User**, **Web Server**, and **Browser**. One can break the **Web Server** to finer-grained interacting actors. For example, the web server can be seen as a set of **Web Applications**, an **Application Server**, a **Web Server** such as IIS or Apache, **Back-end Data Base**, an **Operating System**, and the physical **Server Hardware**. These details can be useful if the security risk assessment includes hardware security and the web server’s vulnerability analysis (e.g., assessing security risks of IIS within the scope of the target web application). However, if the target assessment is purely about the web applications being developed to run on a server, then analyzing the system with the presence of extra actors only complicates the process.

2) Related to this issue, analysts need to capture a good-enough level of abstraction in the models (based on their own judgement), because common guidelines about the necessary and required level of abstraction for developing $i^*$ models are not broadly available.

3) Another challenge in developing $i^*$ models is the lack of an accepted modeling process, that provides the steps and activities needed to create agent- and goal-oriented models of the target system. These gaps are partially addressed by a set of guidelines in [71] and surveys of good and bad $i^*$ modeling practices in [147].

**Step 2:** The $i^*$ model of a software system provides a suitable basis for identifying vulnerabilities within the target system and its surrounding environment. Every resource or task of a particular actor can be vulnerable to certain threats. At this step, vulnerabilities and their potential effects on stakeholders’ goals and system requirements are added to the requirements models. The resulting model provides the high-level directions of risk assessment for the target system. For example, in the simple web application case we discussed in Step 1, the **Browser** renders and creates HTML pages and to do so, it sometimes runs required client-side **script**. This bears the vulnerability of running **Malicious Scripts**. To structure this information within requirements models, the original $i^*$ meta-model needs to be expanded, and additional language syntax and semantics are needed to express the concept of vulnerability and its relation to the tasks being performed within the system. These details are explained in the next chapter (Chapter 4).
Although in Chapter 4 we will discuss how to use $i^*$ requirements models as the basis of the search for vulnerabilities in security knowledge portals, this thesis is not concerned with the problem of discovering relevant vulnerabilities in the target domain, and we assume analysts have the minimum security background to judge whether a vulnerability is relevant in the system being analyzed.

**Step 3:** Once vulnerabilities are identified, analysts can brainstorm possible attack scenarios that exploit the vulnerabilities. The attack scenarios can be modeled using the extended $i^*$ notation, which is syntactically similar to the $i^*$ agent- and goal-oriented modeling, but captures the goals and actions of the potential adversary. For example, mounting Cross Site Scripting (XSS) [125] is an attack scenario that involves exploiting Malicious Scripts to achieve some higher level goals, e.g., stealing end users’ cookies.

Note that the entire modeling process (Steps 1-3) can be iterative, because by adding an attack scenario, analysts may need to add more details to the $i^*$ model, which can result in identifying new vulnerabilities. The iterative nature of our modeling approach is extensively discussed in Chapter 4.

### 3.2 The Risk Assessment Process

The next three steps in the proposed framework focus on assessing the risks of vulnerabilities and exploitation scenarios. In Chapter 5, details of the risk assessment by evaluating vulnerability are explained. This process involves three main steps as follows.

- **Step 4:** Evaluate the probability and damage of vulnerability exploitations by evaluating vulnerability metrics.

- **Step 5:** Identify critical risks that threaten important goals, and suggest alternative security solutions for those risks.

- **Step 6:** Identify security solutions’ impacts and analyze consequences of solutions on risks and stakeholders’ goals.
Step 4: Models developed in earlier steps of the process are the basis of the risk assessment. In Step 4, each exploitation scenario is analyzed by using the Common Vulnerability Scoring System (CVSS), and a probability and damage factor is assigned to the attack scenario. For example, the result of this step indicates that the risk of the Cross Site Scripting exploitation scenario has a Medium probability and a High damage.

Step 5: In Step 5, with respect to the damage and probability of exploitation scenarios, a group of critical risks are selected. Then, analysts brainstorm and identify alternative security countermeasures to prevent the exploitation scenarios, patch the vulnerabilities, or alleviate the impacts of exploitations. For example, to avoid the Cross Site Scripting attack, one can disable JavaScript on the browser, which patches the vulnerability. To avoid drawbacks of having JavaScript disabled, in the web application, developers can consider varieties of alternative solutions that can also be used together: Black List Input Validation, White List Input Validation, and HTTP Input Encoding.

Vulnerability analysis is the basis of identifying relevant security countermeasures. Note that this thesis does not address the process or methods for identifying relevant security solutions, and we assume analysts have the minimum security knowledge needed to search for security mechanisms for given vulnerabilities.

Step 6: In the final step of the risk assessment process, consequences of security countermeasures are evaluated. Security is about trade-offs, and each solution, while reducing the risk of vulnerabilities, can impose costs, performance degradations, usability conflicts, and so forth. Risk assessment is about evaluating the probability and damage of vulnerability exploitations with the presence of possible countermeasure solutions to compare the “before” and “after” of applying each solution considering risk levels and side-effects on other goals.

For example, one solution against Cross Site Scripting is to Disable JavaScript at the Browser. This is an easy and financially low cost solution. The probability of the Cross Site Scripting attack will be zero by applying this solution. However, the usability costs are high: the user will not be able to leverage the benefits of client side scripts that deliver certain functionalities, and may ultimately lose the whole benefit of the target web application.
Adding the countermeasures to the security requirements model involves accommodating new intentional elements within the $i^*$ models of the system. The additional elements that model goals, tasks, and resources relevant to the implementation of a security countermeasure can introduce new vulnerabilities to the system. This requires a re-evaluation of vulnerabilities, risk assessment, and the introduction of additional security solutions. Thus, in general, the process of Step 1 to 6 is iterative, and analysts may revise the security requirements model until all critical vulnerabilities are addressed with a security solution.

### 3.3 The Decision Analysis Process

The decision analysis algorithm (Steps 8 to 10) is iterative: it selects a pair of alternatives to compare and a new cycle starts. At the end of each cycle, one alternative is preferred to the other, and the defeated one is removed from the list of alternatives. These cycles continue until one alternative remains, which is the best solution overall:

- **Step 7**: Aggregate consequences of security solutions into a table, called *consequence table*.

- **Step 8**: Select a pair of alternative solutions for each cycle of the decision analysis process.

- **Step 9**: Suggest a chain of goal tradings to stakeholders to find the preferred alternative in the pair.

- **Step 10**: Find the defeated alternative (the weaker in the comparison), and remove it from the list of solutions.

**Step 7**: The starting point of the decision analysis process is aggregating consequences of alternatives on decision criteria into a *consequence table*. Important goals of stakeholders in the $i^*$ models as well as the damage and probability of risks (which need to be minimized) are the decision criteria. Some of these criteria could be numerically measurable, and some can only be evaluated by using qualitative or nominal labels. Finally, some goals and requirements may not have any scale of measurement, and analysts may not be able to even assign them a qualitative
value. In Chapter 6, we present two decision analysis methods for making security decision with the presence or absence of quantitative and/or qualitative data. The decision analysis process is conducted by requirements analysts that are not security specialists or decision making experts.

**Step 8:** Once consequences of alternative countermeasures are aggregated into a table, a pair of alternative solutions is selected and compared. At this step, the decision aid method helps analyst decide which solution in the pair is overall a better choice with respect to the decision criteria (goals, risks, requirements). This solution is kept in the decision problem and the other one of the pair is removed from the problem and will not be considered in the rest of the process.

**Step 9:** To identify which solution is more preferred in a pair of security solutions, we take advantage of the Even Swaps [64] multi-criteria decision analysis method. This method and its enhancements are introduced in Chapter 6, Section 6.1.2 and Section 6.2.4.

**Step 10:** In the final step of each cycle, the dominated solution in the pair is removed from the list of potential solutions. This reduces the size of the problem and a new cycle of analysis starts. Finally, the remaining solution is recognized as the best overall solution among available options.
Chapter 4

Modeling Security Requirements to Support Trade-off Analysis

“Complex software designs difficult for you to describe textually can readily be conveyed through diagrams. Modeling provides three key benefits: visualization, complexity management, clear communication.” IBM’s Web site

Models capture an abstraction of certain types of information and descriptions, communicate the information, make implicit information explicit, and act as a repository for knowledge and rationale. Modeling requirements facilitates requirements extraction, management, analysis, and visualization, and can also aid system and architecture design and analysis. Security-specific models help in analyzing attacks, risks, countermeasures, and their impacts. By intertwining security-related models with requirements and design models, security concerns can be crossed cut with other software development activities and decisions.

The quality and usefulness of reasoning results critically depend on the appropriateness of the model [76]. There is no “true” or “correct” model, since any model is necessarily an abstraction and the appropriateness of a model depends on the reasoning purpose, i.e. what questions the analyst is trying to answer by constructing and analyzing the model [76]. In security requirements analysis, the choice of modeling approach also depends on the purpose
Chapter 4. Modeling Security Requirements to Support Trade-off Analysis

of reasoning. The purpose of modeling in this thesis is establishing a basis for selecting proper
security solutions, by considering security vulnerabilities and attacks, their risks, and trade-offs
of security with other requirements.

What distinguishes security from other non-functional requirements are the threats posed
from external or internal attackers. These attackers can be external adversaries or be among
stakeholders. Attackers in general try to access or obtain assets. Assets are anything valuable in
an organization [75] and the subject of the attacks [136]. In application security, a vulnerability
is a property of the system or its environment that in conjunction with an attack can lead to
security failures [7]. By analyzing vulnerabilities and attacks from early phases of requirements
analysis and design, we can elicit security requirements that protect the system against the risk
of vulnerabilities and attack scenarios, and we can choose security countermeasures with an
informed judgement about the risks.

However, security is not only a technical matter. Security is both a system and a so-
cial/organizational problem. To capture both aspects of security in requirements modeling
and analysis, in this chapter, we take advantage of the $i^*$ agent- and goal-oriented modeling
framework [150]. The ability of the $i^*$ framework to model agents, goals, and their dependen-
cies makes it suitable for understanding security issues that arise among multiple malicious or
non-malicious agents with competing goals.

The $i^*$ notation offers a way to model actors’ dependencies, goals, assets, and actions. The
$i^*$ modeling notation can express refinement of goals into the actions and assets, and action
decomposition. The resulting requirements model can be a foundation for expressing entities
or actions that introduce vulnerabilities to the system. $i^*$ enables modeling contributions of
actions on stakeholders’ goals. This can be tailored to capture the effects of vulnerabilities on
the satisfaction of system requirements and stakeholders’ goals. By adding vulnerabilities and
exploitation scenarios to the $i^*$ model analysts can trace where a vulnerability is introduced to
the system, how the vulnerability is propagated to other elements and agents, what tasks can
exploit the vulnerability, and what the consequences of exploitations on higher goals are. In
addition, the main benefit of $i^*$ compared to other similar notations like KAOS [28] is that $i^*$
provides required elements for linking one agent or stakeholder to another, using dependency
Chapter 4. Modeling Security Requirements to Support Trade-off Analysis

links. We also adopt and adapt $i^*$ contribution links to relate the attackers (and attacks) to victims (and vulnerabilities). Section 4.6.1 provides some empirical evidence about the potential benefits of the $i^*$ notation for security requirements analysis.

In this chapter, first, we discuss essential concepts in a security requirements modeling notation. We extend the $i^*$ modeling notation with security-specific concepts, mainly vulnerability and attack. We extend the $i^*$ meta-model with security concepts. The meta-model relates the concepts of vulnerability, attack, and security countermeasure to the $i^*$ modeling constructs. Then, we present the security requirements modeling process along with the syntax and graphical representation of the modeling notation. Finally, we discuss security reasoning based on the resulting models.

4.1 The Meta-Model of Modeling Notation

This section investigates the conceptual foundation for the security requirements engineering framework proposed in this thesis. We identify and discuss the basic security conceptual modeling constructs that we adopt in the meta-model of the modeling notation. This discussion is grounded in the security engineering literature.

4.1.1 Relevant Concepts

An asset is anything that has a value to the organization [75]. Assets can be people, information, software, and hardware [29]. They can be the target of attackers and, consequently, need to be protected. Assets such as software products, services, and data, may have vulnerabilities. In software systems, a vulnerability is a weakness or a backdoor in the system which allows an attacker to compromise the correct behavior of the system [7, 134, 118]. Identifying vulnerabilities helps analysts understand how vulnerabilities spread through the system’s components and, consequently, helps evaluate the impacts of exploitations.

Potential ways that an attacker can violate the security of (a component of) a system are called threats (or attacks) [136]. Essentially, an attack is a set of intentional unwarranted actions which attempt to compromise confidentiality, integrity, availability, or any other desired feature.
of the system. Though the general idea of attack is clear, there is no consensus on a precise definition. For instance, Schneider [134] points out that an attack can occur only in presence of a vulnerability. Conversely, Schneier [136] broadens this vision, considering also attacks that can be performed without exploiting vulnerabilities.

Analyzing attacks and vulnerabilities allows analysts to understand how system security can be compromised. Another critical aspect in risk analysis is attackers’ motivations (malicious goals). Examples of malicious goals are disrupt or halt services, access confidential information, and improperly modify the system [11]. Schneier [135] argues that understanding who the attackers are along with their motivations, goals, and targets, aids designers in adopting proper countermeasures to deal with the real threats. Analyzing the source of attacks helps better predict the actions that attackers take.

Threat analysis aims to identify the types of risks that an organization might be exposed to and the harm they could cause to the organization (i.e. the severity of threats). Threat analysis starts with the identification of possible attackers, evaluates their goals, and ways to achieve them. Through threat assessment, analysts can assess the risk and cost of attacks and understand their impact on system security.

Such knowledge helps analysts identify appropriate countermeasures to protect the system. A countermeasure is a protection mechanism employed to secure the system [136]. Countermeasures can be actions, processes, devices, solutions, or systems intended to prevent a threat from compromising the system. For instance, they are used to patch vulnerabilities or prevent their exploitation.

Several frameworks for security analysis take advantage of temporally-ordered models for analyzing attacks [107, 124]. Incorporating the concept of time into the attack modeling helps understand the steps of actions and vulnerability exploitations that can lead to a successful attack. However, temporal aspects add complexity to requirements engineering models which may not be suitable for the early requirements analysis or trade-off security decision making. Especially, in the early stages, risk assessment is about analyzing consequences, costs, and threats, regardless of the ordering of actions that can lead to a risk.

Besides the concepts described above, there are other concepts relevant to security require-
ments. For instance, Massacci et al. [103] integrate concepts from trust management, such as permission, trust and delegation, into a requirements engineering framework to address authorization issues in the early phases of the software development process. Risk analysis frameworks (e.g., [10]) employ the concept of events to model uncertain circumstances that affect the correct behavior of the system. However, events do not allow the analysis of (malicious) intentional behavior and, therefore, they are more appropriate to assess risks and elicit safety requirements in critical systems.

Security is not only limited to the identification of protection mechanisms to address vulnerabilities. Security originates from human concerns and intents [98]; the social issues of organizations where different actors can collaborate or compete to achieve their goals should be considered as part of security requirements analysis [52, 98]. In addition, security is a subjective and personal feeling [137]; therefore, security requirements analysis and security-related decision makings require analyzing the personal and organizational goals of stakeholders as well. For this purpose, we take advantage of agent- and goal-oriented concepts such as intentional actors, goals, and social dependencies. In the following sections, we integrate security concepts into the meta-model of the $i^*$ agent- and goal-oriented framework. There is some evidence in the security requirements engineering literature (e.g., [52, 98, 104, 142]) that these concepts provide the means for analysis of organizational and social contexts in which the system-to-be operates.

### 4.1.2 The Extended $i^*$ Meta-Model

The ability of the $i^*$ framework to model agents, goals, and their dependencies makes it suitable for understanding security issues that arise among multiple malicious or non-malicious agents with competing goals. In particular, $i^*$ enables understanding security from both a technical and social/organizational point of view.

The structure of actors’ dependencies and goals’ hierarchies in $i^*$ models enable propagating vulnerabilities through the decomposition and dependency links to other elements of the model and other actors. Moreover, $i^*$ enables modeling contribution of goals, actions, and assets on other goals. Such relations can be used to capture the effects of vulnerabilities on the
satisfaction of system requirements and stakeholders’ goals.

In this section, we present the meta-model of the security requirements engineering notation, which extends the i* meta-model with security concepts (Figure 4.1). The meta-model includes both the i* Strategic Dependency (SD) diagram, which captures the actors and their dependencies and the i* Strategic Rationale (SR) diagram, which expresses internal goals and the behavior of actors. The extended meta-model includes the concepts of vulnerability, attack, security countermeasures, and their relationships with i* elements.

The i* Meta-Model

In the i* meta-model (Figure 4.1), an actor is an active entity that has strategic goals and intentionality within the system or the organizational setting. Actors carry out activities and produce entities to achieve their goals by exercising their knowhow [150]. Actors can be roles or agents. A role captures an abstract characterization of the behavior of a social actor within some specialized context or domain of endeavor. An agent is an actor with concrete and physical manifestations that can play some role.

Intentional elements in the i* framework are goals, softgoals, tasks, and resources. A goal represents the intentional desire of an actor, without specifying how the goal is satisfied. Goals are also called hard goals in contrast to softgoals which do not have clear criteria for deciding whether they are satisfied or not. A task is a set of actions which the actor needs to perform to achieve a goal. A resource is a physical or an informational entity that can represent assets.

The relations between actors are captured by the notion of dependency. Actors can depend on each other to achieve a goal, perform a task, or furnish a resource. For example, in a goal dependency, an actor (the depender) depends on another actor (the dependee) to satisfy a goal (the dependum). In addition to the dependum, two other intentional elements are involved in a dependency. One element represents why a depender needs the dependum, and the other element specifies how the dependee satisfies the dependum.

The meta-model in Figure 4.1 also describes relationships between intentional elements inside the boundary of actors. Actors have (soft)goals and rely on other (soft)goals, tasks, and resources to achieve them. Softgoals can be decomposed into more softgoals using AND/OR
Chapter 4. Modeling Security Requirements to Support Trade-off Analysis

Figure 4.1: The i* meta-model extended with the concept of vulnerability and attack decomposition relations. Means-end links are relations between goals and tasks, and indicate that a goal (the end) can be achieved by performing alternative tasks (the means). Tasks can be decomposed into any other intentional elements through task decomposition links. Decomposing a task into sub-elements means that sub-elements need to be satisfied or available to have the root task successfully performed.

Softgoals and other intentional elements can contribute either positively or negatively to other softgoals. This is expressed by contribution links. A contribution relation is characterized by an attribute type which can be Help (+), Make (++), Hurt (−), Break (−−), and Unknown (?) values. By linking an intentional element to a softgoal by a Make (Break) contribution, one can express that satisfaction of the intentional element is enough to fully satisfy (fully deny) the softgoal, while a Help (Hurt) contribution indicates that the intentional element has positive (negative) impact, but the impact is not enough to fully satisfy (deny) the softgoal. This qualitative approach reflects the nature of softgoals that do not have clear-cut satisfaction criteria.
Security Extensions to the $i^*$ Meta-Model

This section extends the $i^*$ meta-model with concepts of vulnerability, attack, countermeasure, and their impacts. Figure 4.1 highlights security-specific elements that are added to the $i^*$ meta-model. In this meta-model, by adopting a task or employing a resource, a vulnerability can be brought to the system. The concept of vulnerability is not limited to specific reported vulnerabilities or to general classes of vulnerabilities. For example, one can model the famous worm called 2000 ILOVEYOU\(^1\) or general class of argument injection or modification.

For the sake of simplicity, we call an intentional element that bears or introduce a vulnerability a vulnerable element. Vulnerabilities are concrete weaknesses or flaws that exist in a component of the system like a process, asset, algorithm, or platform, whereas goals and softgoals represent actors’ intentions and quality attributes. In the $i^*$ conceptual framework, adopting a task or employing a resource describes a concrete way of achieving a (soft)goal; therefore, (soft)goals which are abstract, and independent of operationalization, do not introduce a flaw or vulnerability.

Exploitation of vulnerabilities can have an effect on the same element that bears the vulnerabilities or on other tasks, goals, or resources. The effect is characterized by an attribute, type, which can be Hurt (−), Break (−−), or Unknown (?).

An attack is a set of actions that an attacker performs to exploit one or more vulnerabilities to compromise the system or part of it. In Figure 4.1, we use an aggregation relation to indicate the tasks, actors, and vulnerabilities, and their effects are assembled and configured together to mount an attack. This definition of attack is based on the definition proposed by Schneider [134] in which vulnerabilities are a key aspect of any attack. This choice is due to the fact that we are mainly interested in analyzing the effects of vulnerabilities on the system. Attacks that are performed without exploiting vulnerabilities can be modeled by introducing a new class of attacks in which their target is a task or a resource instead of a set of vulnerabilities.

Resources and tasks can have an impact on attacks. Such tasks and resources can be interpreted as security countermeasures; however, we do not distinguish them from non-security

\(^1\)http://www.cert.org/advisories/CA-2000-04.html
mechanisms in the meta-model as these distinctions do not affect the requirements analysis. The impact can have different types: Hurt (−), Break (−−), and Unknown (?). Security countermeasures can patch vulnerabilities, alleviate the effect of vulnerabilities, or prevent the malicious tasks that exploit vulnerabilities.

By patching a vulnerability, the countermeasure fixes the weakness in the system. An example of such a countermeasure is a new update that the software vendor provides for released products. A countermeasure that alleviates the vulnerability effects does not address the source of the problem, but it reduces the effects of the exploitation. For example, a backup system mitigates the effect of security failures that cause data loss. Countermeasures can prevent the actions that the attacker performs, which can consequently prevent exploitation of the vulnerability. For example, an authentication solution prevents unauthorized access to assets.

Countermeasures can prevent the execution of vulnerable tasks or the usage of vulnerable resources. This results in removing the vulnerability brought to the system by these elements. For example, one can disable JavaScript option in the browser to prevent exploitation of malware run by the browser.

Countermeasures can also have side-effects on (soft)goals. However, such impacts are different from security impacts and are captured through relationships such as means-end, decomposition, and contribution relations, which are part of the $i^*$ general meta-model.

Figure 4.2 depicts the extended meta-model, derived from the meta-model in Figure 4.1. Figure 4.2 includes a new type of actor called attacker. An attacker is a specialization of the $i^*$ actor element; thus, the same modeling rules and properties of $i^*$ actors can be applied for modeling attackers.

Within their actor’s boundary, attackers have malicious intentional elements such as malicious goals and malicious softgoals. The concept of a boundary is added to link the malicious elements to the attacker. An attack involves an attacker, malicious tasks that he performs to exploit a set of vulnerabilities, and the effect of exploited vulnerabilities on other actors’ intentional elements.
Discussion

The definition of attack and security countermeasure is fundamentally a matter of perspective: a task or a goal counted as malicious can be perceived as non-malicious from a different point of view. Sequences of actions for mounting an attack are naturally similar to sequences of actions performed by a legitimate actor to mount a counter-attack. Therefore, the line to differentiate malicious actions from non-malicious ones is arbitrary, and distinguishing malicious goals from non-malicious goals depends on the perspective adopted by the system designer.

Malicious elements have the same semantics of ordinary intentional elements: they can be similar or identical to non-malicious elements. For example, the desire to have a high profit is not a malicious goal, but an actor can achieve such a goal either by working legally and honestly or by attacking another organization. A task can be interpreted as malicious in one context,
while being counted as non-malicious in a different context. For example, one can install a camera for spying, intruding on privacy, and so on, whereas a surveillance camera can be used for security purposes. In this example, the goal for performing tasks indicates if the task is malicious or not.

Since malice is a matter of perspective, distinguishing malicious and non-malicious behavior does not affect the security requirements reasoning method. Therefore, the meta-model presented in Figure 4.1 is a neutral meta-model that does not distinguish malicious and non-malicious elements. However, as shown by Sindre and Opdahl [91], graphical models become much clearer if the distinction between malicious and non-malicious elements is made explicit and the malicious actions are visually distinguished from the legitimate ones. Sindre and Opdahl show that the use of inverted elements strongly draws the attention to dependability aspects early on for those who discuss the models. Therefore, the meta-model in Figure 4.2 separate attackers and malicious goals from actors that employ countermeasures for protecting their goals.

4.2 Modeling Vulnerabilities

Vulnerabilities are not independent concepts. They are weaknesses in the actions and resources of the system. Thus, we embed them into the requirements model and attach them to the entities that introduce them to the system. As discussed earlier, we use the $i^*$ framework to model system requirements and stakeholders’ goals. The requirements model captures stakeholders and system actors together with their (soft)goals, the tasks to achieve those goals, required resources, and the dependencies among them. The vulnerabilities layer extends the requirements model by adding the vulnerabilities that tasks and resources brings to the system and the impact that exploitation of vulnerabilities has on the system.

4.2.1 Eliciting and Modeling Initial Requirements

Requirements modeling intends to identify and model stakeholders’ needs and system requirements. As an illustrative example, we analyze requirements and vulnerabilities in a simple web
application scenario. Figure 4.3 shows the requirements “view” of a browser which requests the content from a web server to build HTML pages. We call this a “view” to separate it from attack and solution views of the model and help better modularize complex models. The User depends on a software agent, the Firefox Browser, to Browse web sites. The browser depends on the User to Enter inputs and depends on the Web server for Web page dynamic content and JavaScript.

The requirements view in Figure 4.3 also describes high level goals and tasks of the Browser. For instance, one of the Firefox browser’s tasks is to Show the web pages, and to perform that, the browser needs to Run the JavaScript with user inputs. This makes the final customized HTML page, and to this end, the browser Requests and gets pages from the server and Gets users’ input.
4.2.2 Modeling Vulnerabilities

Vulnerability modeling shifts the focus of analysis to the elements that bear the vulnerabilities and bring risks to the system. To incorporate specific vulnerabilities or classes of vulnerabilities into the requirements model, we identify vulnerable tasks and resources and then propagate vulnerabilities to other elements within the system.

To represent vulnerabilities, the $i^*$ modeling notation is enhanced with a new graphical element (a black circle). The black circle is chosen to resemble a hole or weakness in the system which leaves a back door for attacks. Vulnerabilities are graphically attached to tasks and resources. This implies that the execution of the task or availability of the resource brings the vulnerability to the system. To represent the possible effect(s) of an exploited vulnerability on goals, tasks, and resources, $i^*$ contribution links are reused. The vulnerability effect is visually represented by a dotted line with a label, $l$, where $l \in \{-, -, ?, \}$, similar to the $i^*$ negative contribution links. Vulnerabilities are added as a “layer” on the requirements view. This is called a “layer” to separate it from the rest of the model and help better modularize a complex requirements model.

Figure 4.3 shows vulnerabilities of a browser. One of the browser’s tasks is to Show the web pages, and to perform that, the browser needs to Run the JavaScript with user inputs. The browser Requests and gets pages from the server and Gets users’ input. Each of these tasks bring a vulnerability to the system. By downloading a JavaScript code from the web server, a Malicious script can be downloaded as well. The user inputs can also contain Malicious input. As a result, when the browser runs the JavaScript with the user inputs, the browser is exposed to the combination of the Malicious script and Malicious user input vulnerabilities.

Remarks: $i^*$ Notation Deviation: In some cases, the $i^*$ models in this thesis deviate from the original $i^*$ syntax [150] and the most recent $i^*$ notation guide [71]. Figure 4.4 illustrates a common deviation from $i^*$ dependency modeling, which is a short cut to avoid modeling repeated dependums. Figure 4.5 shows another common deviation of $i^*$ syntax in this thesis, where intentional elements of one actor contribute to goals of other actors directly.
Figure 4.6 shows a deviation from $i^*$ decompositions, where a goal can also be decomposed. Deviations from $i^*$ syntax in the models in this thesis solely aim to reduce the model complexity and excessive number of elements inside the actors’ boundaries. An extensive analysis of $i^*$ syntax deviations are discussed in [66], and the analysis of their advantages and disadvantages is out of the scope of this thesis.

Figure 4.4: Syntactical deviation from $i^*$ dependencies

Figure 4.5: Syntactical deviation from $i^*$ contributions

Figure 4.6: Syntactical deviation from $i^*$ decompositions
4.2.3 Vulnerability Propagation

The vulnerability layer can be expanded on top of the requirements view. When an actor depends on another actor for a vulnerable task or resource, the vulnerability is carried to the depender actor by the vulnerable dependum. Figure 4.7(a) depicts the propagation of vulnerabilities in the reverse direction of dependencies. This figure shows that for dependency relations, the vulnerability $V$ in the dependee’s resource $R_{(how)}$ is propagated to the dependum, $R_D$, and the dependee’s element, $R_{(why)}$. For example, in Figure 4.3, the Web server depends on the users for User’s inputs and stores the users’ provided content. Examples of such applications are Wiki pages, discussion forums, and e-mail servers which store users’ provided content. Later, other users depend on the Web server for Web page dynamic content and JavaScript. However, the User’s input vulnerability gets propagated to other users as the Malicious Script vulnerability. The Malicious Script is brought to the Firefox agent because of the dependency link between the Browser and Web server.

Vulnerabilities can also be propagated through decomposition links. Decomposition links
refine tasks into more detailed elements with higher-resolution information. Since vulnerabilities are usually low-level flaws at the code or system level, it is easier to identify vulnerabilities for concrete sub-elements rather than for high-level abstract ones. Therefore, vulnerabilities are propagated bottom-up from sub-elements to the high level task.

Figure 4.7(b) depicts the vulnerability propagation rule through decomposition links. If a task, $T_{\text{root}}$, is decomposed into a task, $T_{\text{child}}$, and a resource, $R_{\text{child}}$, where $T_{\text{child}}$ and $R_{\text{child}}$ bear vulnerabilities $V_1$ and $V_2$, the root task would receive both vulnerabilities $V_1$ and $V_2$. As shown in Figure 4.7(b), the analyst can either assign the vulnerability effect to the child ($\text{effect}_2$ for $V_2$) or to the root ($\text{effect}_1$ for $V_1$) element based on the context. With respect to the context, one may determine that propagated vulnerabilities have a combined effect as well. Propagating the effects of vulnerabilities cannot be automatically deducted from the structure of the model and requires human judgement and security experiences.

A concrete example of vulnerability propagation through decomposition links is shown in Figure 4.3 where the Run the JavaScript with user inputs task is decomposed into vulnerable tasks. Accordingly, the root task receives both vulnerabilities. These vulnerabilities or their combination can have various effects on the goals and tasks of the browsers when running the JavaScript.

Figure 4.8 shows how vulnerabilities are combined. The effect of exploiting the combination of Malicious script and the malicious user input is expressed using the vulnerability effect link with a $\rightarrow$ (break) contribution from the combination of the vulnerabilities to Protect users’ cookies and Build the correct HTML page.

### 4.3 Modeling Exploitation Scenarios (Attacks)

The aim of attack modeling is to define a view of the security requirements model that represents the possible ways in which attackers can exploit vulnerabilities. To build the attack model, we can take advantage of existing approaches (e.g., Attack Tree [135] and Anti-goals [142]) to develop a tree-like malicious goal model. In addition, catalogues of malicious goals [11] might be useful for driving attackers’ goals.
As discussed earlier, the proposed modeling notation graphically distinguishes malicious and non-malicious elements using a black shadow in the background of malicious elements as proposed in [98, 33]. The exploitation of a vulnerability by a malicious actor is graphically represented by a link, labelled \textit{exploit}, from the malicious task to the vulnerability.

A vulnerability may have different effects on other goals and mechanisms. Different attacks that exploit a vulnerability may have different effects on other elements. Therefore, to graphically relate an attack to the effects of the vulnerability exploitations, the corresponding vulnerability effect links for each attack are labelled with the same tag number that the exploit
link is tagged. In this way, an attack is a quadruple consisting of an attacker, malicious tasks that the attacker performs, a set of vulnerabilities, and their effects (see the meta-model in Figure 4.2). Exploitation scenarios are a separate “view” of the requirements model that link the requirements model with the vulnerability layer. The attack view can be dealt as a separate view, and thus once developed; it can be reused for other requirements views.

For example, Figure 4.8 extends a fragment of the requirements model in Figure 4.3 by introducing two possible attackers: Random hacker and Fake Web Site. Fake Web Site, is a Web server who intends to steal user’s passwords from cookies. The Fake Web Site is modeled in Figure 4.3 as an (inverted) i* role, since we refer to a generic web site rather than to a specific web site.

The Fake Web Site uses the Phishing attack by exploiting the Malicious script. The Random hacker role is a (malicious) specialization of the User role, and thus inherits the user’s capabilities. For instance, the Random hacker can browse a website and enter inputs. The hacker can use these capabilities for his malicious intents such as obtaining other users cookies. One possible way to obtain cookies of other users is Cross-site scripting which consists of injecting a malicious URL into the JavaScript and extracting the cookies from malicious URL logs. To inject a malicious URL into the JavaScript, the hacker injects malicious URL as user input by playing the role of an ordinary user. As discussed earlier, to specify the malicious task that exploits the vulnerability and causes the effect both the exploit and vulnerability effect links are labelled with a tag (number one).

4.4 Modeling Countermeasures

By developing requirements, vulnerabilities, and attack template models, analysts have the machinery necessary to evaluate the risks threatening the system (See Chapter 5). Based on the risk assessment results, analysts select proper security countermeasures to protect the system. To model countermeasures, we have not added a new modeling element to the i* framework, because countermeasures share the same nature with other tasks and resources.

Different countermeasures can have different impacts on attacks. A countermeasure can
alleviate the effect of a vulnerability, patch it, or prevent malicious tasks or system’s functionalities that introduce the vulnerabilities, or a countermeasure can make certain tasks of the attacker more difficult. These impacts are modeled through alleviate, patch, and prevent (make difficult) links respectively. Countermeasures are added to the requirements view and their consequences are linked to the vulnerability layer or attack view. We deal with countermeasures as a “layer” on top of the previous views and layers. This helps to separate different views and layers of complex security requirements models.

Figure 4.9: The countermeasure view for the web server and browser example. (The elements with highlighted color are countermeasures)

The model in Figure 4.9 depicts the countermeasures for the vulnerabilities and attacks in Figure 4.8. The countermeasure elements are highlighted using a darker color. (The highlighted
color in the models does not bear any semantic significance and only intends to highlight the countermeasures in the figures)

In Figure 4.9, the web server employs two security mechanisms: Validate user input and Remove HTML tags from user input. By removing the HTML tags, the malicious code is removed from the user input. This impact is modeled through prevent relations between the countermeasures and the malicious task Inject malicious URL as user input with a negative impact (−). By validating user input, the Malicious user input is partially patched. At the browser side, one can Disable JavaScript and use Anti Phishing toolbar. Disabling JavaScript prevents performing Run the JavaScript with user inputs, hence, the vulnerable task is not performed anymore. As a result, the vulnerabilities that are brought by running JavaScript no longer exist.

4.5 The Iterative Nature of the Modeling Process

We discussed in Chapter 3 that the process of developing goal-oriented security requirements model is iterative, mainly because by adding a countermeasure, new intentional elements need to be accommodated into the requirements view to specify goals of the countermeasures, tasks and resources needed to implement the countermeasures. Countermeasures can also be in the form of entirely new actors that the system components depend on. These additional elements can introduce new vulnerabilities into the system. The introduction of new vulnerabilities prompts a new modeling iteration, and ultimately requires a new process of risk assessment and identifying proper security countermeasures (Chapter 5).

For example, consider the scenario in Figure 4.10. In this general scenario about software development, the ultimate goal of the Application Developer is to make profit by selling copies or licenses of application. This ultimately boils down to make the program piracy difficult. This requires that the Application Program protects the application from being cracked and pirated.

To run the application, the binary image (on disk) is needed. However, having the binary image on the disk brings the vulnerability of static analysis of plain binary on
Figure 4.10: The requirements and attack view, and vulnerability layer of a general application under reverse engineering risk

disk. The attacker accesses binary image (on disk) to create control flow graph from the binary image, which is a reverse engineering technique that helps the attacker understand the code. The attacker, which in this example is an Application Cracker, tries to produce modified versions of the program or DRM content and redistribute it. This ultimately results in Illegal Users being able to install and run the copied application, which has negative impacts on the goals of Application Developer to ensure that only legitimately licensed customers use the software.

Figure 4.11 shows the first countermeasure layer suggested to patch the vulnerability of
The underlying idea behind this solution is to **encrypt the binary code stored on the disk**, so instead of having the plain binary stored on the disk, the application would be in the form of an **encrypted binary (stored on the disk)**. This makes accessing the binary on the disk almost impossible, because to access the plain binary, the attacker actually needs to break a cryptography mechanism. The proposed security solution would **mimic the OS loading behavior**, and remove the encryption on the protected module, and ultimately pass the control to the original program module.

![Figure 4.11: The countermeasure layer of the general application under reverse engineering risk](image)

Although the **Binary Encryption Solution** provides the discussed security benefits, its implementation can introduce new vulnerabilities. At some point, the solution **removes the encryption on the protected module** and this introduces another vulnerability: **Binary obtained from memory image**. Now the attacker can try to **copy the code from memory** and the same attack that was possible previously still threatens the application. Exploiting the new vulnerability by copying the image in the memory is not challenging; thus, another security countermeasure is needed. For example, one solution is to break the binary image to basic code blocks and encrypt the code blocks. Then, at the load time, only individual blocks...
are loaded and run in the memory. These blocks get encrypted and stored on the disk after being run. In this way, the application avoids revealing the entire binary in the memory at once, which makes copying the binary image from the memory very difficult.

Figure 4.12: The countermeasure layer of the general application under reverse engineering risk

In theory, these iterations may result in a never-ending loop. For example, an attacker may find an easy way to put together the basic code blocks (which are loaded into the memory) to create the entire binary image. This threat requires another countermeasure. In practice, however, we aim for good enough security and thus, the iterative modeling process continues until analysts address all critical vulnerabilities by using cost-effective solutions. The decision analysis method to select a set of solutions is discussed in Chapter 6.
4.6 Case Studies and Evaluation

This section focuses on evaluating the proposed modeling notation through modeling case studies and an exploratory study. We model case study scenarios by using our notation as well as other existing modeling notations. We argue how and in what direction, the proposed modeling technique is more expressive and enables risk analysis.

4.6.1 Exploratory Study

Goal-oriented requirements engineering (GORE) techniques are widely recommended for security requirements engineering and risk analysis; however, it is not empirically shown that such methods are analytically and cognitively a fit for security analysis tasks. Empirical evidence has rarely been gathered for assessing the effectiveness, usability, and understandability of GORE for security analysis. Evaluation of GORE methods is often limited to case studies and action research such as [102]. GORE methods have been analytically analyzed based on their expressiveness, or by ontological analysis, analysis of syntax (such as analysis of $i^*$ visual syntax [112]), reflective analysis of $i^*$ syntax in [66], and analysis of $i^*$ models developed by practitioners [41, 147].

We conducted an exploratory study followed by interviews to investigate the impacts of adopting $i^*$ on security analysis tasks. Subjects (individually) analyzed a system in terms of intentional actors, goals, and dependencies in order to identify risks and to choose security countermeasures. Another group of subjects were given an informationally equivalent text scenario. Two representations are informationally equivalent if all information in one is also inferable from the other and vice versa [93].

The effectiveness of the notation can be evaluated by measuring the understanding of analysts through problem solving tasks that require participants to reason about the domain being represented in the model [50]. In this study, subjects were asked to identify security risks, and accordingly suggest security protections to reduce or remove the identified risks. This study showed that subjects who used an $i^*$ model identify different types of risks and protection mechanisms than the ones who used scenarios. The follow-up interviews uncovered
the cognitive process involved in comprehending an $i^*$ model to identify risks and security countermeasures.

**Methodology**

We distributed a scenario (in the online banking domain) among 11 graduate students who were familiar with the $i^*$ modeling notation. Five students were given the scenario in English natural language and the rest were given a corresponding $i^*$ model of the same scenario. Subjects were asked to individually act as the requirements engineer in the project of developing the online banking website. We asked them to identify and list security threats in the login process, and explain what actions may bring threats to the system. Then, we asked them to suggest security protection mechanisms to the designers of the application in order to prevent the threats. Finally, the subjects were given a short questionnaire to survey their level of confidence in their answers, their $i^*$ modeling skills, the ease of understanding the model, etc.

The independent variable is the input material (scenario text or model). The dependant variable is the subjects performance in the requirements engineering task. The outcome of such tasks is not objectively measurable. Therefore, in this study, we focus on qualitative analysis of the output produced by the subjects. We assigned the risks and countermeasures identified by the subjects to different risk patterns.

The patterns of risks and protection mechanisms were developed through a pilot study. In the pilot study, three computer science graduate students participated as subjects in Group 1 and two participated in Group 2. The results of the pilot tests were analyzed using Grounded Theory [55]. From the data collected, the key points were marked, such as web server, data transfer, encryption, etc. These codes are grouped into similar concepts, and from these concepts, categories of risks and protections are formed. For example, to identify the attack types, we detected three types of attack targets or source: 1) attacks against the Web Server and Browser directly, 2) attacks that exploit weaknesses in Users’ actions, and 3) attacks that target the weaknesses during communications among these parties.
Chapter 4. Modeling Security Requirements to Support Trade-off Analysis

Results

The risks identified by the respondents in the first group (who were given the scenario in natural language) focus on vulnerabilities brought to the system through the actions that the User performs. Consequently, subjects in this group typically identify attacks in which the legitimate User is impersonated. For example, the risks of a simple and guessable password, stolen cookies, and password reset by unauthorized users are typical risks identified by the respondents in this group. The Web Server and Browser as attack targets were unnoticed by most subjects. Attacks that can occur during the communication between the Web Server and Browser are mostly unnoticed by subjects in Group 1 as well.

In addition to the attacks that User’s actions bring to the system, respondents in the second subject set focused on risks against the Web Server and Browser, and especially communication between the Web Server and Browser. This can be due to the explicit and visually expressed roles of Web Server and Browser, and dependency links expressed in the $i^*$ model, provided to participants in this group. Dependencies draw the attention to actors that the User depends on and potential threats that they can impose.

Both groups of respondents suggested technical security mechanisms such as encryption, but most of the respondents in the second group suggested protection mechanisms that rely less on technical solutions and more on social interactions or behavior of the users and the bank. For example, the subjects in this group recommended the use of a strong password and secure social interactions between the bank and users, such as going to the bank in person or verifying an e-mail address by the client for the password reset process.

Such recommendations show that an $i^*$ model would draw the attention to the social interactions of the actors and consequently, the analysts would be guided to protect the system through social engineering, such as requiring the user to log-off at the end of a transaction, automatic log-off, enforcing strong passwords, etc. Respondents in Group 2 also identified more protection mechanisms that focus on securing the communication between the Web Server and Browser. For example, the use of secure Internet communication protocols and avoiding the storage and communication of cookies are typical mechanisms suggested by the respondents in the second subjects set.
Follow-up Interviews

The interviewees were asked to reason about elements of the $i^*$ model that they refer to for finding the security risks. They were asked to rationalize why the risks that they identified may threaten the system and how they detected the risks. They were also asked to point out the elements within the model which helped them quickly spot the risks within the system. In what follows, the methods and $i^*$ modeling elements that the participants used are reported.

1) **A dependency is a risk alarm.** Two of the participants expressed that dependencies are one of the main points of vulnerabilities. Dependencies may indicate a communication and information transmission which could be a point of attack against the system. The subjects then brainstormed all possible ways that one could threaten the log-in process during the communication steps.

2) **A means-end link is a security decision point.** Two subjects stated that means-end links are a main point of focus for finding security risks. One participant brainstormed all possible ways that each alternative action through means-end links could be attacked to decide which option is more dangerous. Similarly, the second subject expressed the means-ends are decision points and the option that imposes less risks should be selected.

3) **Softgoals are security trade-off points.** One of the subjects expressed that she does not need to take the softgoals into the consideration when identifying security risks. Two other participants used the softgoals to identify more security risks. For example, one of the subjects pointed at the *Easy to remember password* softgoal within the model, and reasoned that he would examine the risks brought to the system if such a softgoal is satisfied; for example, an easy password is a guessable password or easy to generate by trying combinations. The other participant argued that having a risk in mind, softgoals would help to decide on proper countermeasures that satisfy the softgoals.

4) **Task and goal refinements explain the process, and the process could be flawed.** All three subjects stated that one source for identifying risks is analyzing the *User* goals and actions’ hierarchy. They examined each action, seeking errors or unsafe behavior that may lead to security exploitations.
Exploratory Study Biases

Several biases and limitations threaten the validity of results in this study or similar empirical experiments:

1) The most critical threat to the validity of this experiment is the subjective evaluation of participants’ performance in problem solving tasks. Commonly agreed metrics that reflect the understanding of subjects from the scenario and measure their problem solving performance do not exist. Qualitative evaluation of the output may not accurately measure the actual quality of risks and protection mechanisms that subjects identified. Due to these critical issues, we avoided analysing the results quantitatively, and the conclusions we draw are not based on statistically significant data. Instead, we qualitatively identified differences in patterns and types of output. In such evaluations, human errors and misjudgement cannot be entirely avoided. However, blind evaluation of the output by the researcher prevented biased judgement in favour of a group of subjects.

2) The subjects who participated in the main experiment were familiar with the basics of $i^*$ modeling and analysis. Therefore, the subjects who were not given the $i^*$ model could have analyzed the risks by implicitly thinking in terms of goals, agents, and dependencies. As a result, our study may actually compare the implicit $i^*$ analysis and explicit reasoning on $i^*$ models. While this limits the conclusions we can draw, we avoided recruiting subjects who were not familiar with $i^*$ modeling to participate in Group 1, because they would have less (or different) cognitive skills than the subjects in Group 2, which may cause a significant difference in their performance.

3) Another source of bias in this study is inconsistent domain and security knowledge among subjects. The subjects in Group 2 (who had a superior performance) might have simply been more familiar with Internet security than others. To reduce the chance of such accidental results, this experiment needs to be replicated with a larger number of subjects.

4) One major threat in this study is that the $i^*$ model used in the experiments is not absolutely equivalent with the scenario text description. In practical terms, it is impossible to guarantee that two different representations transmit the same meaning to a human reader.
Keeping this threat in mind, the corresponding model of the scenario was developed by an expert in $i^*$ notation and proof-reviewed by another notation expert.

5) An extraneous variable that may affect the subjects’ performance and results is the scale of the problem. In this study, the scenario specifies a simple and small-size online baking scenario. However, the $i^*$ models may not be suitable for a large-scale scenario and consequently may confuse the analyst. On the other hand, it is possible that browsing a large but visual model results in better performance than reviewing a long and tedious text-based description of the scenario.

In sum, the result of the current study is not generalizable to other modeling notations, problem size, or scenario domains. Conducting empirical studies with minimum biases is extremely challenging, and empirical evidence has rarely been gathered for assessing the effectiveness, usability, and understandability of modeling and analysis methods for security analysis. Thus, in what follows, we focus on case studies and comparison of our methods with existing work to evaluate the framework proposed in this thesis.

### 4.6.2 Modeling Case Studies

Case studies in this section aim to illustrate the expressiveness of the proposed modeling notation in Chapter 4 and compare the modeling approach with other similar and comparable notations. In this section, we apply the notation to two example cases originally used by other authors to illustrate their similar techniques for security requirements analysis. In the first example case, we modeled and analyzed the Guardian Angel (GA) system [122], a patient and physician supporting system, which was modeled and analyzed in [98]. The second case study is the eSAP system, an agent-based health and social care system, which was used as the case study scenario in [20, 114, 113].

The main objective of these case studies is to evaluate and compare existing goal- and agent-oriented approaches with the proposed method in terms of the expressiveness of security models and the ability to express security trade-offs. The case studies demonstrate how different approaches to modeling and analyzing security aspects would enable different types of analysis and design decisions. In addition, the contribution of the current method is distinguished from
the previous work. To illustrate the differences, we have borrowed fragments of the original models developed in [20, 98].

In the third case study, we compare applying two entirely different notations for modeling vulnerabilities with the proposed notation in this thesis: First, integrating the concept of vulnerability into misuse case models, as an example of a requirements modeling approach. Second, revising the CORAS framework with the concept of vulnerability as well, as an example of a risk analysis framework that models and analyzes vulnerabilities.

**Case Study One: Guardian Angel Project**

Guardian Angel (GA) is a patient and physician supporting system using software agents [122]. This information system is centered on the individual patient instead of the provider, in which, a set of software agents integrates all health-related concerns about an individual. This personal system will help track, manage, and interpret the subject’s health history, and offer advice to both patient and provider. Patients provide the medical information to a PDA which is connected to other systems such as the GA-Home, GA-Hospital, and peer GA-PDA.

**Original GA case study:** The methodological framework for dealing with security requirements in [98] is illustrated with the examples of designing GA software agents. The requirements elicitation process in this approach starts with identifying and modeling actors and their goals and tasks (using the i* modeling notation). In parallel, vulnerabilities, malicious actors and intents are identified and integrated into the goal models. Finally, countermeasures against attacks are chosen and added to the models, and the process ends with countermeasure analysis.

This approach takes advantage of the network of Strategic Dependency (SD) [150] to model dependencies among actors and their goals. The resulting models are the basis of analyzing opportunities and vulnerabilities. In vulnerability analysis, each dependency is analyzed as a potential threat against the system. The basic premise for attacker analysis is that all the actors are assumed guilty until proven innocent. In this way, each actor in the goal model can be a potential attacker and is studied in two roles: its regular role and its potential malicious role. One of the actors in the dependency relation is substituted by its corresponding attacker, and
vulnerable dependencies are highlighted with a black shadow rectangle. Figure 4.13 depicts the original model for vulnerability analysis of dependency links in [98]. In this model, GA Hospital Module depends on Insurer Agent. In the vulnerability analysis, each dependency is examined as a potential threat against the system. Each actor plays the role of a potential attacker as well as its regular role.

Figure 4.13: Dependency vulnerability analysis of the GA system in [98].

Figure 4.14 illustrates the original attack model of the GA system presented in [98]. For each vulnerable dependency, alternative attacks are identified and modeled as tasks that contribute to the malicious goal. In the model in Figure 4.14, malicious tasks and their contributions on patients’ goals are located inside the boundary of the malicious actor. Malicious goals and tasks are not distinguished from regular goals. For example, Theft of Permission is an attack (highlighted in Figure 4.14); however, the notation does not syntactically distinguish it from the usual tasks of an actor. Thus, it is not clear whether Theft of Permission is an unwanted and malicious task, or it is a desirable and necessary service that poses negative side-effects on patients’ goals.

In the model in Figure 4.15, hypothetical threats are presented as beliefs, since their existence is based on the designers’ assumptions. In this model, countermeasures are added to the attack model, and are related to attacks with a “break” or “hurt” contribution link (Figure 4.15). The countermeasures affect other goals such as performance and usability. However, the goal model evaluation in the original model is limited to evaluating threats and their cor-
The model does not specify which actor employs the security countermeasures and what side-effects of security mechanisms are on other actor’s goals. For example, User Authentication Mechanism is a countermeasure in Figure 4.15, and it breaks the Theft of Permission attack. This countermeasure is located inside the Peer GA-PDA, which is the defender.

Repeating the GA scenario using the proposed notation: Figure 4.16 depicts the security trade-off model developed using the proposed $i^*$ extensions. In this model, the same security mechanisms and goals in Figure 4.15 are used to structure the GA-PDA goal model. In this view, potential attacks, malicious goals, and sub-goals are taken into account. This model expresses impacts of malicious tasks on attacker’s goals and GA-PDA’s goals.

The model in Figure 4.16 expresses potential malicious behavior, alternative protection mechanisms, their trade-offs and consequences. If potential malicious behavior is refined into a deep-enough hierarchy of malicious tasks and required resources, the leaf nodes can be well-defined steps required for an attack, which provides analysts with metrics to measure the effort and resources needed to launch the attack successfully. This ultimately helps assess risks more objectively.

The goal model captures the effects of each alternative attack on malicious and non-malicious
actors’ (soft)goals. The model explicitly expresses trade-offs that alternative mechanisms and potential attacks impose. For example, two alternative solutions for the Authentication and authorization process are Biometrics and Password-based authentication. While they “prevent” certain malicious tasks, they have side-effects on goals of GA-PDA.

Comparison of Approaches: The proposed modeling technique in this chapter is significantly similar to the modeling approach in [98]. However, the security analysis approach in these two proposals are different in various directions. The underlying idea to identify risks in [98] is based on dependency vulnerabilities. In our method, the starting point of risk analysis is understanding attackers’ intentions. Understanding why attackers are interested in a system is vital. For example, analysts usually need to differentiate between the goals of a professional hacker and the intentions of a curious kid to select proper security mechanisms [135].

The process of risk analysis continues with developing attackers’ goal hierarchy and identifying alternative actions that help attackers to achieve their goals, as well as resources that they need to acquire. The emphasis of our method is refining the goal model into a deep-enough hierarchy, so the leaf nodes provide the analyst rough metrics for assessing the probability and damage of attacks.
Figure 4.16: Security requirements and attack model of the GA system, using the proposed $i^*$ extensions

We apply the same approach in countermeasure analysis. Unlike the approach in [98] which finds counter attacks for each malicious task, we focus on modeling security goals of the defender. Ultimately, we identify security countermeasures that achieve the goals of defender and tackle malicious tasks of attacker as well.

Case Study Two: eSAP System

The second case scenario is the electronic Single Assessment Process (eSAP), which is used as the case study system in [20, 114, 113]. The eSAP system is an agent-based health and social care system for the effective care of older people. Taking into account a substantial part of eSAP from the original case study, the following stakeholders are defined: The Older Person actor is the old person (patient) that wishes to receive appropriate health and social care. The Professional actor represents health and/or social care professionals involved in the care of the Older Person. The DoH actor represents the English Department of Health.
which is responsible for the effective care of the Older Person. The Benefits Agency actor is an agency that helps the Older Person financially, and the R&D Agency actor represents a research and development agency that is interested in obtaining medical information.

Original eSAP case study: The approach proposed in [114, 113] integrates security concerns into the Tropos framework. Figure 4.17 shows part of the original security diagram of eSAP, which contains the security goals, threats, and safeguards. Threats are modeled explicitly in the “security diagram” using a pentagon. The impact of threats on softgoals such as privacy and availability is modeled using contribution links.

![Figure 4.17: Part of the “security diagram” for the eSAP system, original model in [113]](image)

Security constraints are modeled in a diagram called the actor security requirements diagram. In this model, goals are expressed by “constraints”, which are represented using a cloud placed between the dependencies, at the side of the actor who has to satisfy the constraint (Figure 4.18). This approach is useful, since it enables the designer to relate the security requirements to goals and functionalities. The approach in [114, 113] introduces “security diagram”, which uses $i^*$ notation to model security requirements and potential threats.

In [114, 113], once the stakeholders’ goals and security constraints are identified, each actor is studied in more depth and alternative security solutions are examined considering the constraints imposed on the actor. Then, secure entities are introduced to help satisfy the imposed security constraints. These entities are distinguished using an $(s)$; however, it is not discussed
why security requirements and constraints need to be distinguished by other softgoals.

Repeating the eSAP scenario using the proposed notation: Figure 4.19 illustrates the goal model of the professional actor, developed using the proposed extensions to $i^*$. This model includes threats, security mechanisms and contributions of goals and attacks. It also includes malicious tasks and goals and their impact on the goals of the Professional actor. Figure 4.19 illustrates a model for security trade-off analysis of the eSAP system using the proposed approach.

Comparison of Approaches We now consider the limitations in the approach taken in [113] and compare it with the proposed approach in the current work. The actor analysis model (Figure 4.18) does not capture the impact of security goals on attacks and threats. Therefore, alternative security solutions are modeled through means-end relationships without the trade-offs that they impose on other goals and softgoals.

The model in Figure 4.17 does not trace the threats to the source actors, and the relation between countermeasures and threats are not elaborated. The main lack of the security diagram in the original case study is that the model does not include actors; therefore, it is not clear who
Figure 4.19: Interactions of Attacker, Professional and eSAP actors, and security goals and trade-offs, modeled using the proposed notation

the attackers are, why they would attack the system, and what tasks and resources are needed to successfully mount the attack. For example, Cryptography attack remains as a high level and abstract threat and thus assessing the probability and damage of such a threat depends on the subjective opinion of analysts and their understanding of what entitles a Cryptography attack.

Besides neglecting the attack decomposition, in the models in [113], the actor that employs
the mechanisms is not captured, side-effects and security impacts of the mechanisms on other goals of actors are not expressed, and this makes cost-benefit analysis of countermeasures subjective because analysts have not refined the solutions into measurable security metrics and actions. All these missing elements and relationships are explicitly expressed in the model in Figure 4.19, in exchange for adding to the size and considerable complexity of the resulting model.

Case Study Three: Vulnerability Modeling

In the previous sections, we compared our modeling approach with two similar modeling methods that are both centred on applying i* for security requirements analysis. To evaluate the expressiveness of i* modeling for security analysis, compared to other conceptual non-goal-oriented modeling notation, we compare our method for modeling security vulnerabilities with misuse case models [140] and CORAS UML profile models [29]. We analyze and model a simple web application scenario under the risk of Cross-Site Scripting.

Modeling vulnerabilities by Use Cases: The left hand side of Figure 4.20 shows a misuse case model [140] for a web application scenario, and the right hand side of the model shows adding a vulnerability and linking it to (mis)use cases. An attack (misuse case) exploits the vulnerability, and the effect of the exploitation on other uses cases is expressed as a “threaten” link. For example, Cross-Site Scripting exploits Malicious script and user input, and this can threaten Building the correct HTML page.

Modeling vulnerabilities by CORAS: The right hand side of Figure 4.21 depicts the Cross-Site Scripting vulnerability in graphical CORAS risk models. In this model, threats (the threatening actors) are directly connected to vulnerabilities. With a slight modification in the model at the right hand side of Figure 4.21, the relationship between threat scenarios, unwanted incidents, and vulnerabilities is explicitly expressed. The exploitation effects and countermeasure impacts are modeled using the existing CORAS relationships with additional tags. Treatments patch the vulnerabilities, prevent threat scenarios, or alleviate the effect of vulnerabilities.
Repeating the web application scenario using the proposed notation: Figure 4.22 shows the vulnerabilities and related security constructs in the web application scenario by using an i* model. Vulnerabilities are attached to tasks and resources, which implies that the execution of the task or the availability of the resource brings the vulnerability to the system. This model graphically distinguishes malicious and non-malicious elements using a black shadow in the background of malicious elements. The exploitation of a vulnerability by an attacker is represented by a link labelled \textit{exploit} from the malicious task to the vulnerability. The exploitation of vulnerabilities can affect goals, tasks, and availability of resources. In this model, countermeasures are modeled using task elements, and their impacts are expressed as
Figure 4.22: Graphical representation of vulnerabilities in $i^*$ models

contribution links with *alleviate, prevent, or patch* tags.

**Comparison and Discussion:** Each conceptual modeling notation suits a specific purpose and certain types of analysis. For example, use case models in general, and misuse case models specifically, are simple ways for highlighting the main services that different actors receive from the system and potential ways that these services are being misused in a malicious way. CORAS models can highlight main assets, main vulnerabilities and attackers that threaten the assets, and unwanted incidents that are the result of vulnerability exploitations. However, misuse cases and CORAS do not provide constructs to represent delegations of assets and dependencies between actors. Therefore, they cannot model and analyze the propagation of vulnerabilities to system components. Misuse cases and CORAS models do not express why a misuser attacks the system and they do not link the misuser’s actions to his/her goals. As discussed by Schneier [135], understanding why an adversary attacks the system is critical to understanding the probability and damage of attacks.

A common limitation of $i^*$, misuse cases, and CORAS is that they are not expressive enough to model temporally-ordered actions and vulnerability exploitations that lead to an attack. Temporally-ordered models can be crucial tools for analyzing the steps of an attack. On the other hand, CORAS models enable the expression of consequences as unwanted incidents. These unwanted incidents are explicit elements within CORAS models. These explicit incident elements are more expressive than $i^*$, where consequence links need to be traced to the affected elements, and the chain of contributions on higher elements need to be evaluated. While an $i^*$
model is more complex, it provides stronger reasoning power compared to CORAS models; for example, analysis of consequences and effects propagation. (Mis)Use case models are suitable for understanding the scope of security problems, main actors, their expectations, and main risks. Use case models are usually high level and suitable for communicating with project managers or customers, thus, refinement and elaboration mechanisms over use cases and misuse cases are not explored.
Chapter 5

Risk Assessment

“Measuring risk means walking a thin line. Balancing what is highly unlikely from what it totally impossible.”  Jim Campy

The Seven Deadly Sins of Measurement

The risk \( r \) of a security attack is described by the level of damage \( d \) that the successful attack poses to the system and the probability \( p \) that the attack occurs: \( r = p \times d \) [90]. In an ideal situation, security risks are known; their damage is measurable in terms of financial costs, and the probability of different attacks can be estimated based on the history of previous attacks. Ideally, the mitigating impacts of security solutions are quantifiable with respect to probability \( p \) and damage \( d \). However, in practice, quantifying the damage of attacks, estimating attacks probability, and quantifying the mitigating impacts of countermeasures is challenging, error-prone, and involves collecting the subjective (and possibly inaccurate) opinions of security experts, and mining previous attack repositories (which may not be available).

Some requirements or risk factors can be refined into measurable variables, but many non-functional requirements and security metrics have a soft nature, which makes them hard, if not impossible, to measure. Stakeholders may choose to evaluate some factors and requirements numerically, and some requirements qualitatively, on different scales; e.g., absolute values, percentages, ordinal numbers, or qualitative labels. When different requirements are measured (or
estimated) on different scales, normalizing inconsistent types of measures into a single scale is troublesome and may not result in a meaningful utility value.

The other problematic side of risk assessment is that attackers usually try to get one step ahead of defenders. Therefore, software developers and managers always need to anticipate potential threats from the point of view of external adversaries. In this condition, one source of deriving potential threats is identifying and analyzing vulnerabilities within the system. A vulnerability is a weakness of an asset [90], flaw, or error in a specification or implementation which can be exploited to break into the system [127].

Some vulnerabilities, such as buffer over-flow, are common and well-known to the security community. However, the damage and probability of exploitations, even for commonly-known vulnerabilities, are not always quantifiable or available. Vulnerability scoring frameworks such as Common Vulnerability Scoring System (CVSS) [109] assist in evaluating the criticality of vulnerabilities by providing fine-grained metrics that are easier to measure. However, vulnerability analysis in general and CVSS specifically, are not widely adopted in, and intertwined with, security requirements engineering and risk assessment. This could be due to the fact that vulnerabilities are usually flaws, bugs, and errors at the code level, and associating such low-level weaknesses and their risks to requirements satisfaction and design objectives is not straightforward. There is a gap between relating immediate effects of a vulnerability exploitation on the running system to ultimate impacts of this exploitation on requirements and high-level business goals.

**Summary of problems:** One part of the risk assessment challenge stems from the lack of knowledge about vulnerabilities and exploitation scenarios. Other problems stem from the lack of analysis methods:

1. An objective way to measure the probability of exploitations.
2. An objective way to measure the degree of damage that exploitations can cause, and relate it to high-level requirements and goals.
3. An objective way to accurately estimate mitigating consequences (and side-effects) of security solutions.
Chapter 5. Risk Assessment

This chapter tackles these problems. The main contribution of this component is a vulnerability-centric risk modeling and assessment method, which in Chapter 6 is supported by a qualitative decision aid algorithm for selecting proper countermeasures for critical risks. The risk assessment method in this chapter brings together the use of an existing vulnerability scoring system (CVSS) and the security requirements modeling approach introduced in Chapter 4.

The modeling process helps understand how vulnerabilities are introduced to the system, and how they are exploited. The risk assessment method helps evaluate the criticality of exploitations, based on the information expressed in the models. The $i^*$ models provide part of the rationale for evaluating the probability and damage of vulnerability exploitations on system requirements and stakeholders’ goals. We intertwine the CVSS framework with risk assessment, in the sense that the probability and damage of exploitation scenarios on stakeholders’ goals are estimated by CVSS vulnerability criticality metrics. The risk level is expressed in terms of qualitative probability and damage values. We tailor some of the CVSS metrics for assessing the risks of attacks in un-trusted environments.

5.1 From Vulnerabilities to Risks

Identifying relevant risks in the target domain and the technology being used requires understanding the domain and its valuable assets. $i^*$ modeling is a way to understand main players, dependencies, resources, services, and assets within a business domain. The extended $i^*$ modeling notation (with security concepts), which was introduced in Chapter 4, helps express vulnerabilities and their potential effects on stakeholders’ goals or system requirements. The $i^*$ model makes it possible to relate identified vulnerabilities to requirements. For example, the vulnerabilities in a DRM player ($i^*$ model in Figure 5.1) are illustrated in Figure 5.2 and the impacts of potential attacks are related to the elements of the $i^*$ model.

Having an $i^*$ model as the starting point, the main concern is identifying relevant vulnerabilities. Every action (tasks in the $i^*$ notation) or resource within the system can potentially introduce a new vulnerability to the actor that owns the elements, or depends on a vulnerable actor (directly or indirectly).
Common Weakness Enumeration (CWE) [27] and Open Web Application Security Project (OWASP) [121] provide a categorization of vulnerabilities, such as authentication issues, buffer errors, code injection, etc. CWE provides a set of software weaknesses in source code and operational systems as well as weaknesses related to architecture and design. The CWE and OWASP include abstract weaknesses, errors, and vulnerabilities that are independent of specific platforms or technologies. This makes CWE and OWASP suitable starting points to search for relevant vulnerabilities in a given goal model. To detect relevant vulnerabilities, analysts can search the CWE catalogue for related terms explicitly or implicitly expressed in the model.

For example, CWE does not explicitly discuss any vulnerability related to (Valid) activation code or Player key. However, these keys are actually hard-coded in the DRM player. By searching the term hard-coded in CWE, the weakness ID-798, the “Use of Hard-coded Credentials”, is found. CWE states that software that contains hard-coded credentials, such as a password or cryptographic key, leads to a significant authentication failure. In the context of the DRM player, two instances of such hard-coded credentials are Valid activation code and Player key.
Figure 5.2 shows three different attacks that exploit the vulnerability of hard-coded credentials. For example, a software cracker can use DRM player without buying activation code either by extracting the valid code by static code analysis or tampering the binary code to bypass the license check. Common Attack Pattern Enumeration and Classification (CAPEC) and OWASP provide a list of possible attacks that exploit CWE vulnerabilities. The next concern in risk assessment is whether these exploitations are serious, i.e. are these attacks likely to occur and is the damage considerable?

Figure 5.2: Attack scenarios and vulnerabilities against the DRM player
Chapter 5. Risk Assessment

5.2 Adapting CVSS for Vulnerability-Centric Risks Assessment

CVSS is an open framework developed by the National Institute of Standards and Technology for assessing vulnerability criticality across many disparate hardware and software platforms. CVSS in general helps IT managers prioritize vulnerabilities and suggest solutions for remediating those that pose high risks [109]. Adaptation of CVSS in industry is growing, for example, it was incorporated into a web security product by a major security vendor.

The final CVSS score indicates the criticality of vulnerabilities. The score indicates both the level of damage in the case of exploitation and the likelihood of exploitation. CVSS is composed of three metric groups: Base, Temporal, and Environmental, each consisting of a set of metrics. In the original CVSS, these metrics measure the criticality of vulnerabilities. We use the Base and Temporal metrics to assess the difficulty of exploiting vulnerabilities. The difficulty of exploitation ultimately indicates how probable the exploitations are. Environmental metrics as well as negative impacts of exploitations on intentional elements within the i* models indicates the damage of exploitations.

We adopt and adapt the CVSS metrics to evaluate security risks posed by vulnerabilities. Table 5.1 summarizes the vulnerability assessment metrics that are adopted in this work for evaluating the damage and probability of exploitation scenarios. CVSS provides guidelines to evaluate these metrics objectively as well as equations for evaluating the Base, Temporal, and Environmental metric groups to calculate a score between 0 and 10 for each of the metric groups [109]. In this work, we evaluate and aggregate the metrics on the scale of Low, Medium, and High to simplify the evaluation process in the early requirements stage; especially because more detailed numerical data are not available. Ordinal labels of Low, Medium, and High are also suggested in risk assessment best practices such as NIST recommendations [74] and Microsoft SDL [57].

On the other hand, the choice of three ordinal levels is fundamentally arbitrary. The

---

Table 5.1: Adopting and adapting CVSS Metrics for assessing risk of vulnerabilities

<table>
<thead>
<tr>
<th>CVSS Metrics</th>
<th>Sub-Metrics</th>
<th>Possible Values</th>
<th>Implication</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Base Metrics</strong></td>
<td><strong>Access Vector (AV)</strong></td>
<td>Locally (High)</td>
<td>General network (Low)</td>
</tr>
<tr>
<td></td>
<td><strong>Access Complexity (AC)</strong></td>
<td>Low</td>
<td>Adjacent Network (Medium)</td>
</tr>
<tr>
<td></td>
<td><strong>Authentication (Au)</strong></td>
<td>No authentication (Low)</td>
<td>Single authentication (Medium)</td>
</tr>
<tr>
<td></td>
<td><strong>Exploitability (E)</strong></td>
<td>Unproven method (High)</td>
<td>Proof-of-Concept (Medium)</td>
</tr>
<tr>
<td></td>
<td><strong>Remediation Level (RL)</strong></td>
<td>Unavailable fix (Low)</td>
<td>Temporary Fix or Workaround approach (Medium)</td>
</tr>
<tr>
<td></td>
<td><strong>Report Confidence (RC)</strong></td>
<td>Low</td>
<td>Official Fix (High)</td>
</tr>
<tr>
<td></td>
<td><strong>Collateral Damage Potential (CDP)</strong></td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td><strong>Target Distribution (TD)</strong></td>
<td>Low percentage (Low)</td>
<td>Medium percentage (Medium)</td>
</tr>
<tr>
<td><strong>Environmental Metrics</strong></td>
<td><strong>Impact of exploitation on goal g:</strong> Impact(EXP,g)</td>
<td>Low</td>
<td>High percentage (High)</td>
</tr>
<tr>
<td></td>
<td><strong>Damage on Goals</strong></td>
<td>Medium</td>
<td>High</td>
</tr>
</tbody>
</table>

The proposed risk assessment method is independent of the scale of evaluation being used and the aggregation method we use to combine the risk metrics into a probability and damage value can be tailored for other scales. The proposed decision analysis method in Chapter 6 enables analyzing risk factors regardless of the evaluation scale being used.

5.2.1 Adaptation of CVSS for Risk Assessment

The CVSS score indicates the effort and capabilities needed for a successful attack. These factors ultimately show how likely the exploitation of the vulnerabilities: the more effort needed to mount an attack, the less likely the attack is. Another risk indicator is the attackers’ expected gain from the attack or damage caused to the defender side. The higher the gain would be, the higher the risk is.

We adopt and adapt the CVSS metrics from Base and Temporal metrics groups to measure the probability of successful exploitations. Environmental group metrics and impacts of exploitations on stakeholders’ goals are used to assess the damage of exploitations. Table 5.1 shows the mapping of these metrics to ordinal measures of Low, Medium, and High.
Base Metrics

Base metrics represent the intrinsic and fundamental characteristics of a vulnerability that are constant over time and environments. The Base metrics are Access Vector, Access Complexity, and Authentication (See Table 5.1). The Access Vector (AV) metric reflects how the vulnerability is exploited: Locally, through an Adjacent Network, or General Network access. A local access is demanding (High effort), while exploitation through a general network is easier and more probable (Low effort). The Access Complexity (AC) metric measures the complexity of the attack to exploit the vulnerability. The Authentication (Au) metric measures the number of times an attacker must authenticate to a target in order to exploit a vulnerability: Multiple, Single, None (High, Medium, and Low effort respectively).

Model-Based CVSS Evaluation: The main practical challenge to evaluate vulnerabilities using CVSS is assigning an objective value to these metrics. By combining CVSS evaluation and goal-oriented security models, evaluating the CVSS metrics can be supported by referring to the model. The Access Vector (AV) metric can be evaluated by exploring the dependencies between the attacker and the system actors. The analyst can argue about the AV value based on the interactions between the attacker(s) and the system. For example, consider the vulnerability of Malicious user input in Figure 4.8 in Chapter 4. The attacker in that scenario tries to inject a malicious URL into the input. The analyst can reason this step of the attack is performed over the Internet (a general network), and thus, AV= Low (low effort).

The Access Complexity (AC) metric depends on the complexity of the attack scenario that exploits a vulnerability. Two factors in the attack view influence the AC metric: the number of tasks and resources needed to accomplish the attack, and the complexity or effort required to perform each task or furnish the resources. Therefore, evaluating AC strongly depends on the completeness and accuracy of the attack view. For example, in Figure 5.2, Tamper the binary code to bypass the license check is not further decomposed to finer-grained steps. Thus, the AC metric depends on analysts’ judgement about the complexity of the tampering task. Now consider the attack in Figure 4.10, where the attacker tries to Create control flow graph (static analysis) from the binary image. This attack is decomposed into Accessing the binary image and Understanding the branch instruction targets. Now the AC
metric depends on how complex each step is. Accessing the Binary image (on disk) is fairly simple, while Understanding the branch instruction targets can be more complex. Thus, overall the access is neither complex nor simple and AC=Medium for this attack.

Evaluating the Authentication (Au) metric based on the attack view of the security requirements model is straightforward. The presence or absence of tasks and resources related to authentication in the attack model provides the rationale to evaluate the value of Au. For example, in Figure 4.10, authentication is not part of the attack decomposition, and thus Au is None (Au= Low).

Temporal Metrics

Temporal metrics represent the characteristics of a vulnerability that change over time but not among environments. Temporal metrics include Exploitability, Remediation Level, and Report Confidence. Exploitability (E) measures the current state of exploit techniques or code availability: Unproven, Proof-of-Concept, and Functional code or method. With an unproven exploitation method, the effort needed to successfully exploit the vulnerability would be high, while a functional and available exploitation method reduces the required effort (Table 5.1).

Remediation Level (RL) can be Official Fix, Temporary Fix/Workaround approach, and Unavailable Fix to patch the vulnerability. When there exists an official fix for the vulnerability, the required effort to exploit the vulnerability is high. The higher the remediation level is, the harder it is to exploit the vulnerability; hence, these values are respectively mapped to the ordinal values of Low, Medium, and High based on the analyst’s judgement.

Finally, Report Confidence (RC) measures the degree of confidence in the existence of the vulnerability and the credibility of the technical details collected. The higher the confidence is, the higher the aggregated Temporal score would be. Table 5.1 shows the mapping of the metrics to ordinal measures of Low, Medium, and High.

Model-Based CVSS Evaluation: Temporal metrics cannot be evaluated entirely based on the security requirements model. The vulnerability-related information that analysts and modelers gather to develop the attack view of the model provides the rationale to evaluate the metrics. For example, evaluating Exploitability (E) and Remediation Level (RL) metrics
depends on whether modelers and analysts have been able to find exploit techniques and reme-
diation solutions in the knowledge portals such as CWE and OWASP. The Report Confidence
(RC) depends on the confidence of the analysts and the completeness and precision of the
security models.

For example, in the scenario in Figure 5.2, we found several static analysis tools that can
help mount $ATT_1$ easily, and thus, the metric E is Highly available and functional (Low effort).
We found Workaround approaches against static analysis, thus RL is Medium. The model and
search are not comprehensive, thus RC is Medium.

**Environmental Metrics**

Environmental metrics represent the characteristics of a vulnerability that are relevant and
unique to a particular environment. Among environmental metrics, Collateral Damage Poten-
tial (CDP) measures the potential for loss of life, physical assets or revenue through damage or
theft of property or equipment. Target Distribution (TD) is an indicator to approximate the
percentage of systems that could be affected by the vulnerability.

**Model-Based CVSS Evaluation:** Evaluation of the Collateral Damage Potential (CDP)
metric can be supported by the security requirements model. The attack view and vulnerability
layer capture the impacts of vulnerability exploitations on services of system actors and stake-
holders’ goals. Goal model evaluation methods such as [67, 39] can help trace the consequences
of exploitations on higher level goals such as reputation damage or revenue loss. For exam-
ple, consider the vulnerability of **Static analysis of plain binary on disk** in Figure 4.10.
Having the binary on the disk has a negative impact on **making static and dynamic code
analysis hard**. This ultimately has a negative impact on the goal of **Application Program to
Protect application from being cracked and pirated**. This impacts the **Application
Developer’s goals**, and ultimately hurts **Making profit by selling copies or licenses of application**. Thus, for the exploitation scenario in Figure 4.10, CDP= Medium.

Target Distribution (TD) can also be evaluated by analyzing the security requirements
model. In the same way that the CDP is evaluated, analysts can trace the propagation of
the vulnerability exploitation effects throughout the goal graph and evaluate what parts of
the system and what stakeholders are affected by the vulnerability. For example, the impacts of Static analysis of plain binary can be propagated to Application Developer actors and one of their top goals are affected, thus, TD= Medium.

**Vulnerability Impacts Metric**

The Base group also covers impact vectors that measure the damage of the risk. These impact vectors are limited to immediate consequences of exploitations on confidentiality, integrity, and availability (CIA metrics) of data and services. To assess the risks of vulnerabilities we add a tailored metric to the CVSS metric groups: the damage of vulnerability exploitation on stakeholders’ goals and system requirements, without limiting it to CIA metrics. These goals and requirements are presented in the $i^*$ model.

For example, in the scenario in Figure 5.2, the damage of attack $ATT_1$, that exploits $v_1$ ($EXP_{v_1}$), on goal $G_1$ is High. Exploitation of $v_1$ does not affect any other goal. This helps customize the CVSS metrics per project and in terms of goals that are of importance.

### 5.2.2 Tailoring CVSS for Un-trusted Environments

CVSS is not suitable for assessing risks in un-trusted environment. In an un-trusted environment, the attacker can have a full access to the hardware and the binary executable of a software application on the disk or in the main memory. In such contexts, the attacker can inspect, reverse engineer, and tamper with the binary application being run in the un-trusted environment. For example, game consoles, mobile applications, entertainment contents, and any other application that runs at the user side, resides in an un-trusted environment, and is prone to be reverse engineered and cracked.

CVSS metrics are not designed to consider attacks on the binary code in an un-trusted environment. For example, Access Vector values from the Base group are either Local, through an Adjacent Network, or through a Global Network. In the case of attacks in un-trusted environments, the access to the binary code is already provided, and the attacker often even owns the device or software. The Authentication Metric (Au) in the Base group is also irrelevant.
in the context of un-trusted environments, because in these environments, attackers often have a full access to the system that they are trying to crack, and authentication is not required. In case of attacks in un-trusted environments, the required effort to crack the application depends on the size and complexity of the application binary (See Table 5.2). In the Environmental metrics, we also tailored the Target Distribution from the percentage of damage distribution, to the number or percentage of copies of illegal content or applications being distributed (See Table 5.2).

Table 5.2: Tailoring CVSS metrics for assessing risks of attacks against the binary code in un-trusted environments

<table>
<thead>
<tr>
<th>CVSS Metrics for Un-trusted Environment</th>
<th>Sub-Metrics</th>
<th>Possible Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Metrics</td>
<td>Binary Size</td>
<td>Large (High)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Medium (Medium)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Small (Low)</td>
</tr>
<tr>
<td></td>
<td>Binary Code Complexity</td>
<td>Low</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Medium</td>
</tr>
<tr>
<td></td>
<td></td>
<td>High</td>
</tr>
<tr>
<td>Temporal Metrics</td>
<td>Same as other types of attacks (Table I)</td>
<td></td>
</tr>
<tr>
<td>Environment Metrics</td>
<td>Collateral Damage Potential</td>
<td>Same as other types of attacks (Table I)</td>
</tr>
<tr>
<td></td>
<td>Number of copies distributed</td>
<td>Nearly all revenue space (High)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Considerable revenue space (Medium)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Insignificant revenue space (Low)</td>
</tr>
<tr>
<td>Damage on Goals</td>
<td>Same as other types of attacks (Table I)</td>
<td></td>
</tr>
</tbody>
</table>

The DRM player is an example of an applications in an un-trusted environments. Table 5.3 summarizes the evaluation of vulnerability metrics for $EXP_{vi}$ in the DRM player scenario. For example, the Binary Size of the player is small (Low effort), and its complexity is Medium. By exploiting this vulnerability, a considerable revenue space will be affected by the (illegal) distribution of cracked copies.

5.2.3 Aggregating CVSS Metrics

Aggregating Probability Metrics: Once the Base and Temporal metrics are evaluated, the metrics need to be aggregated into a single probability score. Dealing with the risk factors
Table 5.3: The vulnerability metrics for the risk of exploiting $v_1$ in the DRM player scenario

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Value for EXP $v_1$</th>
<th>Aggregated Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Size</td>
<td>Small (Low)</td>
<td></td>
</tr>
<tr>
<td>Binary Code Complexity</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Exploitability ($E$)</td>
<td>Functional method (Low)</td>
<td></td>
</tr>
<tr>
<td>Remediation Level (RL)</td>
<td>Workaround (Medium)</td>
<td></td>
</tr>
<tr>
<td>Report Confidence (RC)</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Collateral Damage Potential (CDP)</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Number of copies distributed</td>
<td>Considerable revenue space (Medium)</td>
<td></td>
</tr>
<tr>
<td>Impact of exploitation on goal $G_i$: Impact(EXP $v_1$, $G_i$)</td>
<td>High</td>
<td>Max(Medium, Medium, High) = High</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Damage = High</td>
</tr>
</tbody>
</table>

in terms of a probability and damage score instead of multiple vulnerability metrics makes it possible to accommodate multiple vulnerabilities and several other criteria in selecting security solutions.

The Base and Temporal groups (e.g., access, authentication, and exploitability) indicate the effort required to mount a successful attack. All of these factors must be present for a successful exploitation, and thus, to evaluate the total effort needed, we calculate the central tendency of these values. These metrics and impacts are evaluated by ordinal scores of Low, Medium, and High. The average (mean) of ordinal values is mathematically meaningless. The mode of metrics would be a valid value for describing the central tendency of ordinal measures. To measure central tendency, we suggest the following rules to calculate an aggregated value ($a$), where Base and Temporal metric values are a set of ordinal values $V = \{v_1, v_2, ..., v_n\}$, and $v_x$, can be Low, Medium, or High, and $l$, $m$, and $h$ are the number of metrics in the set $V$ which are Low, Medium, and High respectively:

Rule 1: if $l \geq h + m$ then $a = Low$

Rule 2: if $h \geq l + m$ then $a = High$

Rule 3: if $m \geq l + h$ then $a = Medium$
Rule 4: if the conditions of Rules 1-3 do not hold, then \( a = \text{Medium} \)

In this way, the aggregated score is the value that the majority of metrics have. For example, if the majority of metrics or half of them indicate that the effort needed to exploit the vulnerability is Low, then in total, the aggregated effort is Low. If none of the Low, Medium, or High values has the majority of values, then, the aggregated score is Medium.

The higher the Base and Temporal metrics are, the lower the probability of the exploitation is, because Base and Temporal metrics measure the capabilities and effort needed to exploit the vulnerability. This is based on the assumption that complex attacks that require significant effort and resources are less likely to occur. Thus:

- If \( a = \text{Low} \) then probability = High
- If \( a = \text{Medium} \) then probability = Medium
- If \( a = \text{High} \) then probability = Low

For example, the third column of Table 5.3 shows the aggregation of vulnerability metrics to calculate the probability and damage scores for \( EXP_{v1} \). For the Base and Temporal metrics, \( l = 2, m = 3, h = 0 \), Medium is the majority of the metrics value; thus the aggregation of Base and Temporal metrics is Medium, and the probability of \( EXP_{v1} \) is Medium.

Note that the expected value of the target asset in an attack can be a major motivation for the attackers, and the higher the asset value from the point of view of the attacker is, the more likely the attack scenario would be. Instead, we consider the influence of assets on the risk level by incorporating the impact of vulnerability exploitations on these assets or satisfaction stakeholders’ goals on furnishing/consuming the assets.

**Assessing Damage Metrics:** The higher the Environmental metrics and exploitation impacts on stakeholders’ goals and resources, the higher the damage is. If even one side of the damage is High, the overall negative consequences would be highly costly, and thus, we take the worst case scenario (pessimistic) as:

Let \( d_1, d_2, \ldots, d_n \) be the values of damage metrics

\[
\text{Damage} = \text{Max}(d_1, d_2, \ldots, d_n)
\]

For example, in Table 5.3, the damage of \( EXP_{v1} \) would be \( \text{Max}(\text{Low}, \text{Medium}, \text{High}) = \text{High} \).
5.3 Risk Treatment

Requirements analysts usually identify several vulnerabilities and potential risks. Some of the risks are insignificant and thus negligible. To identify critical risks, which require employing security solutions, we suggest a risk ranking approach which is adaptable according to the available budget and risk-taking attitude of analysts and project owners. Figure 5.3 shows the ranking of risks according to their probability and damage. When the budget is low and project owners are risk-averse, only the top-ranked risks can be analyzed. If project owners are cautious and the budget is enough, less critical risks can be addressed as well.

![Figure 5.3: Risk ranking based on probability and damage]

For example, in Figure 5.2, we identified two main vulnerabilities and three potential exploitation scenarios. By analyzing the vulnerabilities and their exploitation scenarios using the adapted CVSS metrics, $EXP'_{v_2}$ and $EXP_{v_1}$ are critical risks (rank 2), while $EXP_{v_2}$ is less critical (rank 3):

<table>
<thead>
<tr>
<th>Risk</th>
<th>Probability</th>
<th>Damage</th>
<th>Risk Rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>$EXP_{v_1}$</td>
<td>Medium</td>
<td>High</td>
<td>2</td>
</tr>
<tr>
<td>$EXP_{v_2}$</td>
<td>Medium</td>
<td>Medium</td>
<td>3</td>
</tr>
<tr>
<td>$EXP'_{v_2}$</td>
<td>Medium</td>
<td>High</td>
<td>2</td>
</tr>
</tbody>
</table>

Let us assume that the DRM player project does not have enough security budget and the product owners take risks, thus we only focus on the risk of $EXP'_{v_2}$ and $EXP_{v_1}$ (Tampering the binary code to bypass the license checks and Extracting the player key by static analysis of binary code).
5.3.1 Identifying and Analyzing Security Solutions

In the final step of the risk analysis process, requirements analysts brainstorm alternative security solutions against critical risks and re-evaluate risks with the presence of each solution.

Identifying Alternative Security Solutions:

To protect the system against an attack or vulnerability exploitation, often multiple and alternative security mechanisms are available. Each alternative has some advantages and disadvantages. For example, in the DRM player scenario, the main protection strategies against Static code analysis and tampering attacks are [25]:

1. **Code Obfuscation**: By obfuscating a program, the code is transformed into a form that is more difficult for an adversary to understand or change than the original code. Obfuscation adds an overhead to the code which causes performance drop downs. In a large code base, obfuscation needs to be done by automated tools that guarantee the obfuscated code behaves the same as the original program.

2. **Tamper Proofing**: Even with a strong obfuscation, crackers may be able to reverse
engineer the code and try to modify the program. Tamper proofing algorithms detect that the program has been modified (usually by computing a checksum over the code and comparing it to an expected value). Once tampering has been detected, a tamper response is executed which usually causes the program to fail.

3. **Distribution with Physical Token:** Physical tokens are hardware-based protections that try to provide a safe environment for data, code and execution. By employing a physical token, the user needs to show possession of a token to use the software.

Figure 5.4 shows above alternative security solutions and their side-effects on portability, delay, cost, etc. The model depicts that by applying **Code Obfuscation**, the vulnerability of **Hard-coded credentials** gets patched (although hard-coded credentials are still in the code, they are obfuscated and thus hard to find). By **Tamper Proofing** the impacts of the vulnerability exploitation are alleviated. Finally, the use of **Physical Tokens** forces the **Cracker** to **Clone existing physical tokens** in addition to successful code tampering, which increases the effort for a successful attack. Modeling and analyzing the mitigating impacts of solutions later provides explicit rationale for analysts to re-assess the probability and damage of vulnerability exploitations with the presence of security solutions.

**Re-Assessing Risks with the Presence of Security Solutions:**

Once alternative security solutions are identified and analyzed, their mitigating effects on the probability and damage of exploitations are evaluated. We repeat the risk assessment process using the CVSS metrics in Table 5.1 and Table 5.2 with the presence of security solutions. Table 5.4 shows the risk factors for $EXP_{v_2}'$ with the presence of each solution.

Consequences of alternatives on risk factors (Table 5.4) are not the only factors for selecting a solution among alternatives. As depicted in Figure 5.4, different solutions have side-effects on the cost of the product, portability, and player delays. In the next chapter, we introduce a decision analysis method for analyzing and comparing different options. We consider consequences of solutions on risk factors, their side-effects on goals, and preferences of stakeholders over security objectives and other goals.
Table 5.4: Consequences of alternative security solutions on probability and damage of $EXP'_{v2}$

<table>
<thead>
<tr>
<th>Risk of $EXP'_{v2}$ with Presence of Solutions</th>
<th>Probability</th>
<th>Damage</th>
</tr>
</thead>
<tbody>
<tr>
<td>No security solution</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Code Obfuscation</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Tamper Proofing</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>Distribution with Physical Token</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

5.4 Case Study

We illustrated the risk assessment method by analyzing a Digital Rights Management player. We identified three exploitation scenarios in this example:

1. $EXP_{v1}$: Extract the player key by static analysis of binary code

2. $EXP_{v2}$: Extract the valid code by static analysis of binary code

3. $EXP'_{v2}$: Tamper the binary code to bypass the license check

In the previous sections, we only analyzed the risk of exploiting $V_1$ (See Table 5.3). In this section, we analyze the risk of $EXP_{v2}$ and $EXP'_{v2}$ exploitation scenarios.

5.4.1 Risk Assessment by CVSS

Table 5.2 presents suggested metrics for assessing the risks of vulnerability exploitation in untrusted environments. Table 5.5 and Table 5.6 summarize evaluation of risk metrics for $EXP_{v2}$ and $EXP'_{v2}$ according to Table 5.2.

**Assessment Results:** $EXP'_{v2}$ (tampering to bypass licence check) is a more challenging exploitation scenario, because exploitability of $EXP'_{v2}$ is at a “functional method” level, which requires Low effort. $EXP_{v2}$ is at a “proof of concept” level, which requires Medium effort. However, due to the aggregation of vulnerability metrics, the probability of both exploitation scenarios is Medium. This shows that although our method simplifies the risk assessment by
aggregating risk metrics into a probability and damage value, the aggregated values are less accurate than the individual metrics.

Table 5.5: The vulnerability metrics for the risk of exploiting $v_2$ ($EXP_{v_2}$) in the DRM player scenario

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Value for $EXP_{v_2}$</th>
<th>Aggregated Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Size</td>
<td>Small (Low)</td>
<td></td>
</tr>
<tr>
<td>Binary Code Complexity</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Exploitability (E)</td>
<td>Functional method (Low)</td>
<td></td>
</tr>
<tr>
<td>Remediation Level (RL)</td>
<td>Workaround (Medium)</td>
<td></td>
</tr>
<tr>
<td>Report Confidence (RC)</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Collateral Damage Potential (CDP)</td>
<td>Medium</td>
<td>Max(Medium, Low, Medium) = Medium</td>
</tr>
<tr>
<td>Number of copies distributed</td>
<td>Small revenue space (Low)</td>
<td>Damage = Medium</td>
</tr>
<tr>
<td>Impact of exploitation on goal G1: Impact($EXP_{v_1}, G_1$)</td>
<td>Medium</td>
<td></td>
</tr>
</tbody>
</table>

\[ l = \# \text{Low} = 2 \]
\[ m = \# \text{Medium} = 3 \]
\[ h = \# \text{High} = 0 \]

\[ m > l + h \text{ thus } a = \text{Medium Probability} = \text{Medium} \]

5.4.2 Re-evaluating Risks in the Presence of Security Solutions

In Chapter 5, we introduced three potential security solutions against the tampering attack scenario ($EXP'_{v_2}$). The effects of these alternative solutions on the damage and probability of is summarized in Table 5.4. In this section, we re-evaluate the risk of $EXP'_{v_2}$ in the presence of each solution.

The second column of Table 5.7 shows the risk metrics of $EXP'_{v_2}$ without applying any security solution. This table shows that with applying Code Obfuscation, exploiting $V_2$ would become harder, but the probability of $EXP'_{v_2}$ remains Medium. By applying Tamper Proofing, exploiting $V_2$ would become even more difficult, in such a way that the probability of $EXP'_{v_2}$ drops down to Low. Tamper Proofing can also alleviate the damage of tampering attacks to the Medium level. By employing Physical Tokens, both probability and damage become Low.
Table 5.6: The vulnerability metrics for the risk of exploiting $v_2 \ (EXP'_{v_2})$ in the DRM player scenario

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Value for EXP'_{v_2}</th>
<th>Aggregated Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Size</td>
<td>Small (Low)</td>
<td></td>
</tr>
<tr>
<td>Binary Code Complexity</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Exploitability (E)</td>
<td>Proof-of-concept (Medium)</td>
<td></td>
</tr>
<tr>
<td>Remediation Level (RL)</td>
<td>workaround (Medium)</td>
<td></td>
</tr>
<tr>
<td>Report Confidence (RC)</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Collateral Damage Potential (CDP)</td>
<td>Medium</td>
<td></td>
</tr>
<tr>
<td>Number of copies distributed</td>
<td>Considerable revenue space (Medium)</td>
<td></td>
</tr>
<tr>
<td>Impact of exploitation on goal G_i: Impact(EXP_{v_1}; G_i)</td>
<td>High</td>
<td></td>
</tr>
</tbody>
</table>

Assessment Results: These results match the common belief that hardware tokens can be more secure than software-based solutions.

5.5 Discussion

As discussed in the introduction of this chapter, measuring (or estimating) the probability and damage of vulnerability exploitations is often challenging, or unreliable, even if possible. Evaluating these factors requires the subjective opinion of stakeholders and domain experts; however, they usually cannot provide explicit rationale for their evaluations. In this chapter, we introduced a risk assessment method which helps stakeholders and analysts systematically reason about the damage and probability of vulnerability exploitations in terms of vulnerability criticality metrics. Although analysts and stakeholders’ (subjective) judgement is ultimately polled to evaluate CVSS metrics, the proposed method breaks down the risk value into fine-grained vulnerability metrics, instead of directly asking for a high-level risk value for each vulnerability. The breakdown provides finer-grained variables. This helps analysts justify and
support their judgement more objectively by referring to the metrics.

Evaluation of metrics are then aggregated to reduce the number of factors that analysts need to consider for choosing a proper security solution. However, the main drawback of aggregating metrics into one score is losing the precision of information that analysts have collected; for example, five Base and Temporal metrics, with each metric having three possible values (15 different conditions) are all mapped to three different possible probability scores (Low, Medium, and High). In critical cases, analysts can deal with the individual metrics as a risk factors, and accommodate all metrics in the decision making process to select proper security solutions. The other possible solution is to map five Base and Temporal metrics in Low, Medium, and High, to a probability score in a more granular scale, like Low, MedLow, Medium, MedHigh, and High.

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Value for $\text{EXP}^v_2$</th>
<th>With Code Obfuscation</th>
<th>With Tamper proofing</th>
<th>With Physical Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Size</td>
<td>Small (Low)</td>
<td>Small (Low)</td>
<td>Small (Low)</td>
<td>Small (Low)</td>
</tr>
<tr>
<td>Binary Code Complexity</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>Exploitation (E)</td>
<td>Proof-of-concept (Medium)</td>
<td>Function method (High)</td>
<td>$P = M$</td>
<td>$P = L$</td>
</tr>
<tr>
<td>Remediation Level (RL)</td>
<td>Workaround (Medium)</td>
<td>Workaround (Medium)</td>
<td>Official Fix (High)</td>
<td>Official Fix (High)</td>
</tr>
<tr>
<td>Report Confidence (RC)</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>Collateral Damage Potential (CDP)</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Number of copies distributed</td>
<td>Considerable revenue space (Medium)</td>
<td>Considerable revenue space (Medium)</td>
<td>$d = H$</td>
<td>$d = M$</td>
</tr>
<tr>
<td>Impact of exploitation on goal $G_1$: Impact($\text{EXP}^v_1,G_1$)</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
</tbody>
</table>
Chapter 6

Decision Analysis in the Absence of Numerical Data

“The most important things cannot be measured.” W. Edwards Deming

“If you cannot measure it you cannot control it.” John Grebe

“...“you can’t manage what you can’t measure” - what makes it a fallacy is that we manage things we can’t measure all the time. We manage cancer research. We manage software design. We manage all manner of things that are deeply intellectual, even creative, without any idea of what numbers we ought to have to guide us. Good knowledge worker managers tend to measure qualitatively, not quantitatively.” Robert Glass [56]

Making trade-offs among requirements is ultimately a decision problem with multiple criteria. In an ideal condition, analysts would like to quantitatively measure (or estimate) consequences of alternative solutions on requirements. However, while some requirements can be refined into measurable variables, many non-functional requirements have a soft nature, which
Chapter 6. Decision Analysis in the Absence of Numerical Data

makes them hard, if not impossible, to measure. Obtaining measurements can be challenging and expensive, even in the physical world. Measuring software qualities or design objectives is even harder, because these are often intangible and soft in nature, specially in the early phases of the projects.

For example, a security expert may not be able to numerically evaluate the security of a simple username and password login mechanism versus a sophisticated Biometrics authentication. A universal and absolute scale of measurement for security level of authentication mechanisms is not available. In any case, security is not the only factor to be considered when deciding between two authentication mechanisms. Costs, portability, performance, usability, and privacy issues need to be evaluated and compared as well.

In summary, the main problems when making trade-offs among requirements and deciding over alternative design solutions are:

1. Lack of accurate quantitative data: at the early stages of development, accurate quantitative data about how well a design choice satisfies quality goals and mitigate security risks is often not available.

2. Manual prioritization: extracting stakeholders’ preferences over multiple criteria in terms of numerical importance weights is error-prone and labor-intensive.

3. Incomparable scales: aggregating requirements measures from different scales into a single value is usually error-prone or not possible.

4. Extensive data collection: eliciting required information to make an objective decision usually involves collecting extensive amounts of data from stakeholders.

5. Scalability: the decision problem can become complicated and impossible to be analyzed manually when a large number of requirements and/or alternatives are evaluated.

This chapter tackles these problems, by providing two main decision analysis methods for two different conditions: 1) when quantitative data about the risk and mitigation and satisfaction of goals/requirements is not available, 2) when some factors are measurable quantitatively and the rest (or all factors) are evaluated qualitatively and in different scales. The decision aid methods can be generalized to any decision problem with multiple criteria and a set of
Chapter 6. Decision Analysis in the Absence of Numerical Data

alternative actions. Although this thesis focuses on the impact of security risks on the final decision, the methods and tool introduced in this chapter can be applied to non-security risks as well.

Section 6.1 presents a decision analysis approach, for the first condition, where stakeholders are not able to measure or estimate the risk factors or goal satisfaction (either qualitatively or quantitative). The underlying idea of this method is to compare alternatives rather than to measure their costs and benefits on some absolute scale.

If some requirements can be refined to measurable variables, and some others can only be roughly and qualitatively evaluated (condition 2), we adopt the Even Swaps multi-criteria decision analysis approach [64] to make trade-offs among requirements (Section 6.2). The Even Swaps method works by trading off one requirement for another, thus, it does not require eliciting importance weights of requirements for determining the best solution. Requirements can be evaluated in varieties of scales and by different measurement methods. The Even Swaps method is also employed as part of the method in Section 6.1 for extracting stakeholders’ value trade-offs. We have extensively improved the Even Swap process by automating alternative and swap selection process.

6.1 Comparing Alternatives for Analyzing Requirements Trade-offs

Faced with the typical absence of reliable quantitative data, systematic decision analysis methods are needed for making trade-offs when direct measurement of decision factors is not possible. The underlying idea in this section is to compare alternatives rather than to measure costs and benefits of alternatives on some absolute scale. This section proposes a heuristic and systematic reasoning method to aid decision analysis based on comparing alternatives.
6.1.1 Structuring the Decision Analysis Problem

In order to structure and reason about a decision problem three main concepts need to be modeled and analyzed: 1) Alternatives (potential actions) 2) A family of criteria or goals 3) A problem formulation which captures the consequences of alternatives [45, 129]. The \( i^* \) notation provides expressive-enough elements to represent the requirements and alternative solutions in the form of a decision problem. \( i^* \) goals and softgoals indicate decision criteria such as costs, quality requirements, and design objectives of stakeholders. If such goals, costs, and constraints are for different stakeholders, each individual goal of each stakeholder represents a separate decision criteria in the decision problem formulation. For example, some of the implementation costs are bounced on customers and some are imposed on system developers. These two different costs can be broken into two different decision factors.

**An Illustrative Example:** We illustrate the decision analysis method in this chapter with a scenario at the Ministry of Transportation, Ontario (MTO), Canada. Traffic monitoring operators control and manage the traffic by constantly monitoring traffic videos. MTO managers need to decide whether to keep an existing traffic management system (referred to as \( A_1 \)) or deploy a new web-service based Intelligent Transportation System, called ONE-ITS (referred to as \( A_2 \)). ONE-ITS provides and distributes the data necessary to carry out traffic management operations and amalgamates all of the information sources into one platform.

Figure 6.1 shows the goal model for the Variable Message Sign (VMS) management subsystem of the ONE-ITS project. The VMS sub-system needs to **update the messages easily**, which requires **easy to learn VMS management** and **simple VMS manipulation toolkit**. The alternative systems, \( A_1 \) and \( A_2 \), contribute to requirements of the VMS system differently. For example, ONE-ITS system **displays the VMS devices on an electronic map**, which “helps” satisfy **easy to learn VMS management** and **simple VMS manipulation toolkit**. However, ONE-ITS system **enables operations over a web portal** which “hurts” the **secure modification of messages**.

In this scenario, multiple decision criteria need to be considered to make the final decision. Although the new system facilitates sharing and distributing traffic data, decision makers at the MTO are concerned about unknown security threats against the web-service access to the
traffic data. Managers at the MTO have to deal with other trade-offs. For example, the existing system interacts with the traffic operators through a simple command-line interface, which is easy to use for the expert users. The ONE-ITS provides a graphical user interface which includes electronic maps that show the cameras and other traffic devices graphically. Such an interface is easy to use and learn by novice traffic operators; however, the existing expert users would rather keep the old and familiar command-line interface.

Additionally, MTO managers need to compare the implementation and maintenance costs of the new and old systems. Although developing the ONE-ITS system is costly; the system will dramatically reduce the maintenance costs. In order to decide about switching to the new ITS, MTO managers must examine costs (and other negative impacts) and benefits of the new proposal compared to the existing situation. Measuring the security level, usability, and maintainability of the existing system is challenging (if not impossible); however, measuring those qualities for the ONE-ITS system, which only exists as a system requirements specification, is not feasible. In order to make a final decision, the MTO managers were to rely on estimations of those measures for the ONE-ITS; however, estimations may reduce the confidence of the final decision. In the rest of this paper, we will use our method based on comparing the alternatives instead of measuring them, to propose a solution to the MTO stakeholders.
6.1.2 Method Overview

This section overviews preliminaries, definitions, and underlying ideas of decision analysis based on comparing alternatives.

Making Choices by Comparing Alternatives

“When we deal with intangible factors, which by definition have no scales of measurement, we can compare them in pairs. Making comparisons is a talent we all have.” — Tomas L. Saaty [132]

Comparing intangible factors is a well-known approach in MCDA methods to avoid direct measurements or estimations. The foundation of methods such as Analytic Hierarchy Process (AHP) [131] is based on the intuitiveness of comparison instead of direct measurement. The underlying idea of this contribution is based on comparing alternatives. The criteria to compare the alternative solutions are the impacts of solutions on the satisfaction of stakeholders’ goals and requirements.

For example, between a simple authentication mechanism based on a username and password and a Biometrics system, a security expert can easily decide which authentication mechanism is more robust. The security expert may pick the Biometrics mechanism, because it is “much stronger” than a simple password. Project owners can also recognize the more economical option. Biometrics authentication mechanism is much more costly than a simple username and password implementation. This cognitive ability of humans for comparing decision elements is used as the basis of analyzing requirements trade-off problems in Section 6.1 of this chapter.

Consider the username and password authentication ($A_1$) and the Biometrics authentication mechanism ($A_2$). We ask stakeholders which solution is more secure? Which solution has a better performance? Which solution costs less? Which solution is more usable, and to what extent? It is much easier to elicit answers to these questions from stakeholders as rough comparisons between alternatives (e.g., much more secure, somewhat secure) than quantitative measures about their level of security.
**Remark:** If a solution such as \( A \) is a better choice than \( B \), with respect to a goal like \( G \), we write \( A > B \) on \( G \).

For the sake of simplicity let us compare the authentication mechanisms (\( A_1 \) and \( A_2 \)) with respect to their performance and security only. We use these qualitative comparisons to make an objective decision between \( A_1 \) and \( A_2 \). Possible rankings of \( A_1 \) and \( A_2 \) are limited to:

1. One of the alternatives has a higher level of security as well as a better performance (e.g., 
   \( A_1 > A_2 \) on performance and security)

2. One of them is more secure, and the other has a higher level of performance (e.g., \( A_1 > A_2 \) on performance and \( A_2 > A_1 \) on security)

Deciding between \( A_1 \) or \( A_2 \) in the first situation is straightforward without the need to have any numerical data on their level of security and performance. In the second situation, we need to make a trade-off between performance and security. For example, a password-based authentication (\( A_1 \)) is faster than the Biometrics authentication (\( A_2 \)), but Biometrics are more secure than passwords. If security is a more important concern for stakeholders, then \( A_2 \) (Biometrics) is a better choice.

Comparing two alternatives with respect to two criteria is relatively simple. When multiple criteria affect the final decision, making trade-offs is not as straightforward as this example. The decision problem becomes more complex when stakeholders are able to express the degree of difference between consequences of alternatives. For example, if we know the performance of \( A_1 \) is “highly” better than the performance of \( A_2 \) (\( A_1 > A_2 \) highly on Performance), and the security of \( A_2 \) is only “somewhat” stronger than the security of \( A_1 \) (\( A_2 > A_1 \) somewhat on Security), which solution is a better choice? \( A_1 \) might be preferred to \( A_2 \), because the security of \( A_1 \) and \( A_2 \) is not significantly different, while the performance of \( A_1 \) is remarkably better than the performance of \( A_2 \).

Comparing alternatives in conjunction with priorities over criteria becomes even more complex. We propose a decision aid algorithm to deal with such complexities and make a decision by relying on comparisons instead of direct measurement. To initiate the algorithm, stakeholders compare the alternatives, and the algorithm enumerates and explores possible consequences.
of alternatives with respect to the extent of difference between consequences of alternatives on different criteria. In the following section, we establish a set of definitions and notations to explain the method.

**Preliminaries**

The Scale for Comparing Alternatives: Psychological evidence from empirical studies, which are the foundations of the AHP method, shows that humans can easily distinguish high/medium/low, and then subdivide again into high/medium/low within each interval [132]. This in total results in nine different levels for distinguishing the decision elements, which is cognitively the maximum level of comparisons that humans can easily perform [132]. Due to efficiency concerns with the decision aid algorithm, we use a less granular scale for comparing the consequences of alternatives: 0, Low, Medium Low, Medium, Medium High, and High. The difference between two successive intervals is equal to Low. The maximum difference of the relative orderings of two alternatives is High. The maximum level of satisfaction of goals is High, and 0 indicates a negligible difference between two alternatives.

Comparing Alternatives: We ask stakeholders to specify which alternative in the pair has a stronger impact on a given goal. Using the introduced scale, stakeholders are asked to estimate the extent of the difference between consequences of alternatives. For example, if the performance of alternative $A_1$ is highly better than the performance of $A_2$, we write $\frac{A_1}{A_2} = \text{High}$ on performance, and if $A_2$ is more secure than $A_1$ and the difference between the security of $A_1$ and $A_2$ is Medium, we write $\frac{A_2}{A_1} = \text{Medium}$ on security.

Remark: $\text{Sat}(g, A)$ denotes the satisfaction level of goal $g$ that alternative $A$ achieves. When we write $\frac{A_2}{A_1} = \text{Medium}$ on security, it indicates that $A_2 > A_1$ on security and also $\text{Sat}(\text{Security}, A_2) - \text{Sat}(\text{Security}, A_1) = \text{Medium}$.

Remark: Note that it is necessary to assume the distance between Low and Medium is the same as the distance between Medium and High. We acknowledge that stakeholders may not interpret the ordinal scales such as Low, Medium, High as equal intervals; however, our method is based on assuming comparisons are made on an interval scale.
The Concept of “Possible” Consequences: When stakeholders state that \( \frac{A_2}{A_1} = \text{Medium} \) on security, on the scale of 0 to High, it is not known whether the security level of \( A_2 \) is High or Medium. We can only conclude that the security level of \( A_1 \) is less than \( A_2 \) (by the extent of Medium). To deal with the lack of definite (numerical) data about the satisfaction level of criteria, we enumerate all possible consequences of alternatives on stakeholders’ goals. To illustrate the enumeration of possible consequences, consider \( \frac{A_2}{A_1} = \text{Medium} \) on security. We can make various guesses about the possible consequences of \( A_1 \) and \( A_2 \):

1) \( \text{Sat}(\text{Security}; A_2) = \text{Medium} \) and \( \text{Sat}(\text{Security}; A_1) = 0 \)
2) \( \text{Sat}(\text{Security}; A_2) = \text{Medium High} \) and \( \text{Sat}(\text{Security}; A_1) = \text{Low} \)
3) \( \text{Sat}(\text{Security}; A_2) = \text{High} \) and \( \text{Sat}(\text{Security}; A_1) = \text{Medium Low} \)

Possible consequences of \( A_1 \) and \( A_2 \) are limited to these three guesses. For example, in the third possible consequence, where \( \text{Sat}(\text{Security}; A_2) = \text{High} \), the security level of \( A_1 \) is lower than \( A_2 \), by the extent of Medium, which means:

\[
\text{Sat(\text{security}; A_1)} = \text{High} - \text{Medium} = \text{Medium Low}
\]

Remark: Note that in the above calculations, the interval scale of 0 to High is mapped to the numbers between 0 to 5.

Consider another criterion: Performance of \( A_1 \) is highly better than the performance of \( A_2 \):
\( \frac{A_1}{A_2} = \text{High} \) on Performance. We enumerate possible consequences of \( A_1 \) and \( A_2 \) on performance by making guesses about the performance of \( A_1 \) and calculating the possible consequence of \( A_2 \) on performance accordingly. For example, first we assume:

\[
\text{Sat(\text{Performance}; A_1)} = \text{High}; \quad \text{thus } \text{Sat(\text{Performance}; A_2)} = 0
\]

Because \( \text{Sat(\text{Performance}; A_2)} = \text{Sat(\text{Performance}; A_1)} - \text{High} = \text{High} - \text{High} = 0 \). Since 0 is the minimum and High is the maximum satisfaction level on the scale, possible consequences of \( A_1 \) and \( A_2 \) on performance are limited to the above case.

In a general case, stakeholders compare a pair of alternatives such as \( A \) and \( B \) with respect to \( m \) different criteria. Alternatives \( A \) and \( B \) may have up to five different possible consequences on each criteria (if their extent of difference is Low). By considering \( m \) different criteria, in total, we can enumerate up to \( 5^m \) different possible consequences for \( A \) and \( B \). One of these
possible consequences is the actual consequence of $A$ and $B$ on the criteria; however, that is not known. Thus, each of these possibilities is treated as the actual consequences of $A$ and $B$. Then, the final decision between $A$ and $B$ depends on analyzing all possible consequences of $A$ and $B$.

**Using Possible Consequences in Decision Analysis:** Each possible consequence is a guess about actual consequences of alternatives. These possibilities are limited due to the relative rankings of alternatives and the extent of difference between their impacts on goals. Only one of the enumerated possibilities is the actual consequence of alternatives, but this is not known. However, it is possible that finding the actual consequences of alternatives among all possibilities is not needed to make a final decision; for example, if one alternative is a better choice with respect to all possible consequences, then it is the best alternative in any case.

The decision analysis approach is ultimately heuristic, in the sense that if by applying the Even Swaps method we conclude that one of the alternatives is the preferred solution for all or a majority of possible consequences, we can assume it is probably the overall best solution as well. If neither of the alternatives in the pair is the best solution for all possible consequences, then the final decision depends on further interactions with domain experts and stakeholders for eliciting more information.

Later, we discuss the detailed algorithm for analyzing enumeration of possible consequences to decide between a pair of alternatives. The bottom line in this process is identifying the best alternative in the pair for each possible consequence. For this purpose, we adopt the Even Swaps [64] decision analysis method.

**Basics of the Even Swaps Method**

The Even Swaps decision analysis method (implicitly) extracts stakeholders’ value trade-offs, i.e. how much stakeholders would give up on one goal for more satisfaction of another. The trade-offs that people are willing to make are an important indication of their preference model [126]. For example, one possible consequence of $A_1$ and $A_2$, with respect to Performance and Security, is as following:
Deciding between $A_1$ and $A_2$ involves making a trade-off between security and performance. Making this trade-off depends on stakeholders’ preferences. In this example, analysts need to understand the extent of performance that stakeholders are willing to sacrifice for better security. In an even swap, analysts ask stakeholders a hypothetical question:

*If we could (hypothetically) reduce the performance of $A_1$ from High to 0, how much (hypothetical) improvement on the security level of $A_1$ would you expect?*

Let us assume stakeholders declare that they would be willing to sacrifice performance and agree to reduce it from High to 0, and in return, they expect that the security level of $A_1$ would be increased from Medium Low to Medium High. Making these changes is not actually possible. In other words, a solution with the performance of 0 and security of High is not available, but it implicitly reflects the value trade-offs of the stakeholders. This is a virtual, hypothetical alternative which we call $A_1'$. The virtual alternative, $A_1'$ is as preferred as $A_1$, and can be used as its substitute.

\[
\begin{array}{ccc}
\text{Performance} & \text{Security} \\
A_1 & \text{High} & \text{Medium Low} \\
A_2 & 0 & \text{High} \\
\end{array}
\]

Now deciding between $A_1'$ and $A_2$ is straightforward, because the performance level of both is 0; thus performance is irrelevant in the decision problem. With respect to the remaining factor (security) $A_2$ is a better choice. In sum, we can conclude that although performance of $A_1$ is High, $A_2$ is preferred to $A_1$ because $A_2$ has a High security, which indicates security is more important for stakeholders.

Generally, in an even swap, the decision analyst, collaborating with the stakeholders, hypothetically changes the consequence of an alternative on one requirement, and compensates this change with a preferentially equal change in the satisfaction level of another requirement.

**Notation Remark.** A swap between two goals $g_x$ and $g_y$ that changes the satisfaction value of $g_x$ from $x$ to $x'$ and compensates this change by modifying the satisfaction level of $g_y$.
from $y$ to $y'$ is written as:

$$ (g_x : x \rightarrow x' \iff g_y : y \rightarrow y') $$

Swaps aim to either make criteria irrelevant, in the sense that both alternatives have equal consequences on the criteria, or create a dominant alternative. Alternative $A$ dominates alternative $B$, if $A$ is better than (or equal to) $B$ on every criteria [116]. Irrelevant goals and dominated alternatives can both be eliminated, and the process continues until the most preferred alternative remains [116].

**Remark:** Most often different decision criteria originate from different and potentially competing stakeholders. If there is a disagreement among stakeholders about the swap results, we assume they would resolve it through negotiation methods [128, 73, 72]. In this thesis, we assume that stakeholders can reach a consensus about the swap results and resolving the preference conflicts of stakeholders is outside the scope of our work.

### 6.1.3 The Decision Analysis Method based on Comparing Alternatives

This section describes the heuristic decision analysis process. A decision analyst applies the method and interacts with stakeholders to get their input.

To start the analysis, first the decision problem is structured into a set of criteria, solutions, and consequences. In each cycle of the algorithm, a pair of alternatives is compared. Each cycle consists of four steps. Figure 6.2 summarizes the dynamics of the proposed heuristic decision aid method:

1. **Comparing a pair of alternatives:** In the first step of a cycle, the algorithm takes a pair of alternatives, and given stakeholders' goals and decision criteria, the tool interacts with stakeholders, and asks them to compare consequences of alternatives on each goal.

2. **Enumerating possible consequences of alternatives:** In the second step, all possible consequences of alternatives are enumerated based on the extent of difference between alternatives. In other words, the comparisons made between alternatives help to narrow down possible consequences that the pair of alternatives could have.
3. **Determining the best solution for each possible consequence:** In the third step, the algorithm applies the Even Swaps method to determine the preferred solution in the pair for every enumerated case. This solution is called the dominant one (for the possible consequence being analysed).

4. **Determining the overall best alternative in the pair:** In the fourth step, the overall dominant alternative in the pair is determined with respect to the results of the Even Swaps analysis (for all possible consequences that are enumerated).

Figure 6.2: Steps of the decision analysis method based on comparing alternatives

If an alternative is the dominant one for all possible consequences, then that alternative is definitely the overall best solution in the pair. Otherwise, further information needs to be
collected from stakeholders to determine the overall best solution in the pair of alternatives. The overall dominant solution is kept in the list of alternative solutions, and the dominated solution is removed from the list. Then a new cycle starts with a new pair of alternatives and the cycles continue until only one solution remains in the problem structure, which is the best available solution.

Step 1: Comparing Consequences of Alternatives

In this step, stakeholders compare consequences of two alternatives such as $A$ and $B$ on goal $g$, and specify which alternative has a stronger contribution on $g$ and what the extent of difference between $A$ and $B$ is ($\frac{A}{B}$ or $\frac{B}{A}$).

**Illustrative Example:** Two alternative traffic management systems, $A_1$ and $A_2$, are compared in Figure 6.3: $A_2$ is highly better than $A_1$ for Secure modification of messages ($\frac{A_2}{A_1} =$ High on $G_1$). The Performance of $A_1$ is better than the Performance of $A_2$, but the difference between $A_1$ and $A_2$ on performance is Low ($\frac{A_1}{A_2} =$ Low on $G_2$).

Figure 6.3: Comparisons of two alternative systems at MTO.
Step 2: Enumerating Possible Consequences of Alternatives

In the second step, possible consequences of alternatives are enumerated with respect to the extent of difference between the pair of alternatives. Let’s call the pair of solutions $A$ and $B$ which affect a set of goals, $G = \{g_1, g_2, \ldots, g_m\}$. One possible consequence of $A$ on the goals is the set of $\{a_1, a_2, \ldots, a_m\}$. The corresponding consequence of $B$ on the goals is calculated with respect to the differences of $A$ and $B$ on the goals (consequence of $B$ on $\{g_1, g_2, \ldots, g_m\}$ is $\{b_1, b_2, \ldots, b_m\}$):

If $\frac{B}{A} = d_x$ on $g_x$ then:

$d_x \leq b_x \leq \text{High}$ and $a_x = \text{High} - b_x$

If $\frac{A}{B} = d_x$ on $g_x$ then:

$d_x \leq a_x \leq \text{High}$ and $b_x = \text{High} - a_x$

Note that there are at most five different values for $a_x$ (and $b_x$) for every $g_x$. If $\frac{B}{A} = \text{Low}$, $b_x$ has five different values and if $\frac{B}{A} = \text{High}$, $b_x$ can only have one value (High).

**Illustrative Example:** For example, in Figure 6.3, $d_1 = \frac{A_2}{A_1} = \text{High}$ on $G_1$, thus possible consequences of $A_2$ are values in the range between the value of $d_1$ and High. The only possible consequence of $A_2$ on $G_1$ is thus High. Accordingly, the only possible consequence of $A_1$ on $G_1$ is High—$Sat(G_1, A_2) = 0$. Possible consequences of $A_1$ and $A_2$ on the rest of the goals in Figure 6.3 are as follows:

On goal $G_2$, $d_2 = \text{Low}$, thus five different consequences are possible:

1) $Sat(G_2, A_1) = \text{Low}$ \hspace{1cm} $Sat(G_2, A_2) = 0$
2) $Sat(G_2, A_1) = \text{Medium Low}$ \hspace{1cm} $Sat(G_2, A_2) = \text{Low}$
3) $Sat(G_2, A_1) = \text{Medium}$ \hspace{1cm} $Sat(G_2, A_2) = \text{Medium Low}$
4) $Sat(G_2, A_1) = \text{Medium High}$ \hspace{1cm} $Sat(G_2, A_2) = \text{Medium}$
5) $Sat(G_2, A_1) = \text{High}$ \hspace{1cm} $Sat(G_2, A_2) = \text{Medium High}$

On goal $G_3$, $d_3 = \text{High}$, thus only one consequence is possible:

1) $Sat(G_3, A_1) = \text{High}$ \hspace{1cm} $Sat(G_3, A_2) = 0$

On goal $G_4$, $d_4 = \text{High}$, thus only one consequence is possible:

1) $Sat(G_4, A_1) = \text{High}$ \hspace{1cm} $Sat(G_4, A_2) = 0$

On goal $G_5$, $d_5 = \text{Medium}$, thus three different consequences are possible:
1) \( \text{Sat}(G_5, A_1) = \text{Medium} \quad \text{Sat}(G_5, A_2) = 0 \)
2) \( \text{Sat}(G_5, A_1) = \text{Medium High} \quad \text{Sat}(G_5, A_2) = \text{Low} \)
3) \( \text{Sat}(G_5, A_1) = \text{High} \quad \text{Sat}(G_5, A_2) = \text{Medium Low} \)

On goal \( G_6 \), \( d_6 = \text{Medium} \), thus three different consequences are possible:
1) \( \text{Sat}(G_6, A_2) = \text{Medium} \quad \text{Sat}(G_6, A_1) = 0 \)
2) \( \text{Sat}(G_6, A_2) = \text{Medium High} \quad \text{Sat}(G_6, A_1) = \text{Low} \)
3) \( \text{Sat}(G_6, A_2) = \text{High} \quad \text{Sat}(G_6, A_1) = \text{Medium Low} \)

On goal \( G_7 \), \( d_7 = \text{Medium High} \), thus two different consequences are possible:
1) \( \text{Sat}(G_7, A_2) = \text{Medium High} \quad \text{Sat}(G_7, A_1) = 0 \)
2) \( \text{Sat}(G_7, A_2) = \text{High} \quad \text{Sat}(G_7, A_1) = \text{Low} \)

Possible consequences of \( A_1 \) and \( A_2 \) are limited to all unique combinations of consequences of \( A_1 \) and \( A_2 \) on different goals. In total, 90 possible consequences for the pair of \( A_1 \) and \( A_2 \) can be uniquely enumerated by combining the consequences of \( A_1 \) and \( A_2 \) on \( G_1, G_2, ..., G_7 \):
\[ 1 \times 5 \times 1 \times 1 \times 3 \times 3 \times 2 = 90. \]
For instance, one possible consequence of \( A_1 \) and \( A_2 \) on \( G_1, G_2, ..., G_7 \) is given in Table 6.1.

<table>
<thead>
<tr>
<th>Alternatives</th>
<th>( G_1 )</th>
<th>( G_2 )</th>
<th>( G_3 )</th>
<th>( G_4 )</th>
<th>( G_5 )</th>
<th>( G_6 )</th>
<th>( G_7 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A_1 )</td>
<td>0</td>
<td>Low</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( A_2 )</td>
<td>High</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Medium</td>
<td>Medium High</td>
</tr>
</tbody>
</table>

**Step 3: Determining the Dominant Alternative for each Possible Consequence**

In the third step, the algorithm finds the dominant alternative in the pair for each possible consequence. If an alternative is dominant with respect to all criteria of comparison, it is the absolute dominant one:

**Definition 1.** Consider a set of goals as \( G = \{g_1, g_2, ..., g_m\} \). Out of the pair of \( A \) and \( B \) that affect goals in \( G \), alternative \( A \) is called the absolute dominant, if
\[ \forall g_x \in G, \quad \text{Sat}(g_x, A) > \text{Sat}(g_x, B) \]

**Remark:** Under the above condition, \( B \) is absolutely dominated by \( A \), and we write \( A > B \).

For example, in Table 6.1, neither \( A_1 \) nor \( A_2 \) is absolutely dominant. This is a typical
situation, because every alternative solution satisfies some of the goals better than the other alternative. We use the Even Swaps [64] method to determine the dominant alternative, when neither of them is the absolute dominant for a possible consequence.

Illustrative Example of the even swaps: Consider the consequence of $A_1$ and $A_2$ in Table 6.1. We ask stakeholders:

"if interoperability of $A_1$ was reduced from High to 0, how much improvement would you expect on security in return?"

This swap enquiry is written as $(G_4 : High \rightarrow 0 \Leftrightarrow G_2 : 0 \rightarrow x)$. The value of $x$ depends on the importance of security and interoperability for stakeholders. If we assume that from stakeholders’ point of view security is more important than interoperability, then stakeholders would agree to reduce the level of interoperability to 0, and in return, have a more secure solution, e.g., they expect Medium security (improving security from 0 to Medium). After applying the swap, consequences of $A_1$ (now $A'_1$) are modified as follows:

<table>
<thead>
<tr>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_3$</th>
<th>$G_4$</th>
<th>$G_5$</th>
<th>$G_6$</th>
<th>$G_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A'_1$</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>0</td>
<td>Medium</td>
<td>0</td>
</tr>
<tr>
<td>$A_2$</td>
<td>High</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Medium</td>
</tr>
</tbody>
</table>

Now $G_4$ is an irrelevant criterion and can be removed from the problem. The Even Swaps process continues by swapping next goals, until one of the alternatives absolutely dominates the other:

- Ask stakeholders $(G_6 : 0 \rightarrow Medium \Leftrightarrow G_5 : Medium \rightarrow x)$

- Stakeholders agree $x = 0$

<table>
<thead>
<tr>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_3$</th>
<th>$G_4$</th>
<th>$G_5$</th>
<th>$G_6$</th>
<th>$G_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A'_1$</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
<td>0</td>
<td>Medium</td>
<td>0</td>
</tr>
<tr>
<td>$A_2$</td>
<td>High</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Medium</td>
</tr>
</tbody>
</table>

- $G_5$ and $G_6$ are irrelevant criteria and can be removed from the problem.

- Ask stakeholders $(G_1 : Medium \rightarrow High \Leftrightarrow G_3 : High \rightarrow x)$

- Stakeholders agree $x = Low$

<table>
<thead>
<tr>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_3$</th>
<th>$G_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A'_1$</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>$A_2$</td>
<td>High</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- $G_1$ is an irrelevant criterion and can be removed from the problem.
- Ask stakeholders ($G_3: Low \rightarrow 0 \Leftrightarrow G_7: 0 \rightarrow x$)

- Stakeholders agree $x = \text{Low}$

<table>
<thead>
<tr>
<th></th>
<th>$G_2$</th>
<th>$G_3$</th>
<th>$G_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A'_1$</td>
<td>Low</td>
<td>0</td>
<td>Low</td>
</tr>
<tr>
<td>$A_2$</td>
<td>0</td>
<td>0</td>
<td>Medium High</td>
</tr>
</tbody>
</table>

- $G_3$ is an irrelevant criterion and can be removed from the problem.

- Ask stakeholders ($G_2: Low \rightarrow 0 \Leftrightarrow G_7: Low \rightarrow x$)

- Stakeholders agree $x = \text{Medium Low}$

<table>
<thead>
<tr>
<th></th>
<th>$G_2$</th>
<th>$G_7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A'_1$</td>
<td>0</td>
<td>Medium Low</td>
</tr>
<tr>
<td>$A_2$</td>
<td>0</td>
<td>Medium High</td>
</tr>
</tbody>
</table>

- $G_2$ is an irrelevant criterion and can be removed from the problem.

- $A_2$ absolutely dominates $A'_1$ with respect to the remaining criteria

- Thus $A_2$ is the overall dominant solution.

**Step 4: Determining the Overall Dominant Alternative in the Pair**

We discussed earlier that we analyzed each possible consequence of alternatives by applying the Even Swaps method [64]. After each individual possible consequence is analyzed, the proposed heuristic decision analysis method identifies the overall dominant solution out of the pair of alternatives by applying a set of simple reasoning rules:

**Reasoning Rule 1:** As shown in Figure 6.2, if by applying the Even Swaps method we realize that an alternative is the dominant one for all possible consequences of a pair of alternatives, that alternative is the overall best solution as well. In such a case, the final decision is made independent of identifying the actual consequence of the alternatives among the enumerated possibilities.

**Reasoning Rule 2:** However, it is possible that in the pair of alternatives such as $A$ and $B$, $A$ is dominant for a number of possible consequences and $B$ is dominant for some others. Let us assume that $A$ is dominant for $x$ number of enumerated consequences and $B$ is dominant for $y$ number of possibilities. If $x$ is sufficiently small, those $x$ enumerations are probably exceptional cases for which $B$ is not dominant. These small number of possibilities can be verified by consulting with decision stakeholders. Stakeholders evaluate whether those consequences are
possible, based on their domain knowledge.

Typically, there are patterns of consequences where one solution is dominant and the other is dominated. The tool provides these patterns of exceptional cases to domain experts, analysts, and stakeholders for further exploration. The user needs to evaluate whether exceptional patterns of consequences are valid. For example, assume solution $B$ dominates $A$ for all possibilities, except for the cases in which the security level of $B$ is lower than Medium. Such cases are exceptional patterns of consequences, and if recognized as invalid by stakeholders, a straightforward final decision can be made: if security of $B$ is not lower than Medium, those exceptional cases can be removed from the pool of possible consequences of alternatives, and $B$ is recognized as the overall best solution.

**Reasoning Rule 3:** Stakeholders and domain experts may also state that the exceptional patterns of consequences are actually valid. For example, analysts may state that the security level of $B$ is actually Low (or worst). This knowledge indicates which enumerations of consequences are valid (match the analyst’s judgement) and which of the enumerated consequences must be removed.

Now the set of possible consequences are again checked against Rules 1 to 3 to find the overall best solution. This process continues until one solution is dominant for all remaining possible consequences.

**Illustrative Example:** We apply the introduced reasoning method to analyze 90 possible consequences of $A_1$ and $A_2$, and determine the overall dominant solution. We analyzed one of the possible consequences (Table 6.1) and by applying the Even Swaps method, we concluded that $A_2$ dominated $A_1$. By applying the Even Swaps to the remaining possible consequences, we observed that:

- $A_2$ is dominant for 45 of the possible consequences where $Sat(G_7, A_2) = \text{Medium High}$ and $Sat(G_7, A_1) = 0$.
- $A_1$ dominates $A_2$ for the other 45 cases where $Sat(G_7, A_2) = \text{High}$ and $Sat(G_7, A_1) = \text{Low}$.

Therefore, the final decision between $A_1$ and $A_2$ depends on collecting further information
about the consequences of $A_1$ and $A_2$ on $G_7$ ($G_7$ indicates how well each solution minimizes the development costs). This observation indicates that the consequence of alternatives on $G_7$ influences the final decision significantly, in a way that if the level of $G_7$ is Low, $A_1$ is preferred to $A_2$, while if the level of $G_7$ is 0, $A_1$ is no longer an acceptable solution. Assume we ask stakeholders about the impact of alternatives on $G_7$, and assume stakeholders believe the level of $G_7$ by employing $A_1$ is at least Low or better. Thus, we can conclude that enumerated consequences where $Sat(G_7, A_1) = 0$ are invalid and can be removed. In the remaining pool of enumerated cases, $A_1$ dominates $A_2$ for all cases; thus, finally we can conclude that $A_1$ is overall a better choice compared to $A_2$.

6.1.4 Identifying the Dominant Alternative by Even Swaps

In the third step, the algorithm takes advantage of the Even Swaps decision analysis method to find the dominant alternative in the pair for each possible consequence. Applying the Even Swaps, in the way that the original method works, is problematic in the context of the proposed decision aid method, because:

- Although the Even Swaps method helps identify stakeholders’ value trade-offs and compare alternatives systematically, analysts need to decide which goals to swap in each step based on their personal judgement. For example, in the MTO case scenario, given seven criteria, 42 different swaps are possible to start the swapping process.

- When several criteria are considered, determining the right swaps to make among numerous possibilities is hard for human decision makers [116]. The Even Swaps process does not provide a systematic method or guidelines for stakeholders/analysts to carry out the process; thus non-expert users may not know which goals to swap next. The Even Swaps process may turn to a process of making numerous swaps in a tedious and long process without creating a dominated and a dominant alternative.

- We apply the Even Swaps method for analyzing several, enumerated possible consequences of alternatives. For example, for the MTO decision scenario, we enumerated 90
possible consequences on seven goals. For the first possible consequences, we swapped five goals; with some approximations, we may need to swap 450 goals in total to find the dominant solution for all 90 possible consequences. In general, at most, five different consequences on each goal can be enumerated (High, Medium High, Medium, Medium Low, and Low); thus, given $m$ goals, the Even Swaps process may be invoked up to $5^m$ times for analyzing possible consequences of every pair of alternatives. Since the swapping process involves user interactions, $5^m$ number of even swaps quickly become a labor-intensive and tedious process.

To solve the scalability issues we propose a set of rules for automatically suggesting proper swaps to decision stakeholders. We provide a semi-automated algorithm that aims to reduce the number of swap enquiries through reusing swaps. If purposefully suggested, swaps can help “create” an absolute dominant alternative with the fewest swaps that are easy to make as well.

**Creating an Absolute Dominant Alternative**

Swaps make one of the decision criteria irrelevant, which helps remove one goal from the decision problem in each step. Removing criteria one by one is a time-consuming approach to apply the Even Swaps process. Our tool suggests swaps that help toward making one of the alternatives dominant. That means if an alternative such as $A$ is dominant for $m$ goals like $g_1$, $g_2$, ...$g_m$, and $B$ is a better solution with respect to only one goal, $g_n$, then we need to remove $g_n$ by swapping it with one of those $m$ goals. By removing $g_n$, solution $A$ might still be dominant with respect to all goals ($g_1$, $g_2$, ...$g_m$). On the other hand, swapping any two goals from the set of $g_1$, $g_2$, ...$g_m$ will not change the situation between $A$ and $B$, i.e. neither of them becomes dominated (dominant) with respect to all goals.

**Rule 1, Making swaps that help toward creating a dominant alternative:** Given a set of goals as $G = \{g_1, g_2, ...g_m\}$, if consequences of alternatives $A$ and $B$ on the goals in $G$ are $\{a_1, a_2, ...a_m\}$ and $\{b_1, b_2, ...b_m\}$, and $n_1 =$ number of goals where $a_x > b_x$ and $n_2 =$ number of goals where $b_x > a_x$ (for $1 \leq x \leq m$), then $n_1 \times n_2$ swaps exist that can potentially reduce the number of even swaps for making an alternative dominant, where one of those $n_1$ goals must
be swapped with one of those $n_2$ goals.

Applying this rule for selecting goals to swap guarantees that after a chain of successful swaps, the algorithm converges to one alternative as the dominant one. Because firstly, each swap removes one decision criterion. Secondly, each swap subtracts the total number of remaining swaps required to make one alternative dominant.

**Illustrative Example:** Consider consequences of $A_1$ in Table 6.1. $A_1 > A_2$ on four goals: $G_2$, $G_3$, $G_4$, $G_5$, and $A_2 > A_1$ on $G_1$, $G_6$, $G_7$. By swapping $G_2$ and $G_3$, one of them can be removed (as an irrelevant goal), but still, three more swaps are needed. On the other hand, by swapping $G_2$ and $G_1$ and removing $G_1$, two more swaps are left to create a dominant alternative. Therefore, by following Rule 1, 12 swaps exist that help toward creating a dominated (dominant) alternative: swapping a goal from the list of \{$G_2, G_3, G_4, G_5$\} with a goal from the list of \{$G_1, G_6, G_7$\}, aiming to remove one of $G_1$, $G_6$, or $G_7$. We need at least three swaps to remove $G_1$, $G_6$, and $G_7$ and make $A_1$ the absolute dominant.

**Suggesting the Most Reusable Swap**

Swaps that stakeholders have previously made can be reused for other cases, without further consultation with human stakeholders, which prevents repeating the same swap enquiry from stakeholders.

**Rule 2, pick the most reusable swaps:** When there are several goal tuples like $(g_x, g_y)$ for the next swap, the algorithm selects the most reusable swap. Assume stakeholders have made a swap such as $(g_x : x \rightarrow x' \Leftrightarrow g_y : y \rightarrow y')$ in a previous cycle. Now to decide between two alternatives such as $A$ and $B$ in a different cycle, this swap is reusable iff:

- $Sat(g_x, A) = x$ and $Sat(g_y, A) = y$ (Consequences of $A$ on $g_x$ and $g_y$ are $x$ and $y$)
- $Sat(g_x, B) = x'$ (Consequences of $B$ on $g_x$ is $x'$)

Under the above conditions, alternative $A$ can be replaced with $A'$ where consequences of $A'$ on $g_x$ and $g_y$ are revised to be $x'$ and $y'$. This results in an irrelevant goal $(g_x)$ which can be removed.
**Illustrative Example:** Consider the consequences of $A_1$ in Table 6.1. We discussed that 12 potential swaps help toward creating a dominant alternative:

$(G_1, G_2), \ (G_1, G_3), \ (G_1, G_4), \ (G_1, G_5)$

$(G_6, G_2), \ (G_6, G_3), \ (G_6, G_4), \ (G_6, G_5)$

$(G_7, G_2), \ (G_7, G_3), \ (G_7, G_4), \ (G_7, G_5)$

In order to suggest the next proper swap, the tool applies rule 2 and finds the most reusable swaps. For example, the swap between $G_1$ and $G_2$ would be as:

$(G_1 : 0 \rightarrow \text{High} \leftrightarrow G_2 : \text{Low} \rightarrow x)$

This swap is reusable for every other possible consequence of $A_1$ and $A_2$ where $Sat(G_1, A_1) = 0$, $Sat(G_1, A_2) = \text{High}$, and $Sat(G_2, A_1) = \text{Low}$. Among all 90 possible consequences for $A_1$ and $A_2$, 18 have the above satisfaction levels for $G_1$ and $G_2$, and thus the swapping $G_1$ and $G_2$ is reusable to 18 other possible consequences. Swapping $(G_1, G_3)$ and $(G_1, G_4)$ is reusable to all 90 possible consequences. Thus among 12 candidate tuples of goals for the next swap, $(G_1, G_3)$ and $(G_1, G_4)$ are the most reusable ones and the tool suggests swapping these goals to stakeholders.

**Suggesting Easy Swaps**

In addition to considering swaps reusability, the algorithm suggests swaps that decision stakeholders would be willing to make. Hammond et al. [64] suggest making the easiest swaps first, e.g., money is an easy goal to swap. In general, what would make a swap easy for stakeholders?

*Rule 3, swap minimally satisfied goal with maximally satisfied ones:* Stakeholders may easily agree to improve on a goal that is not sufficiently satisfied and compensate it with decreasing the satisfaction level of a requirement that is highly satisfied, intending to reach a balance among software requirements. This rule states that when comparing a pair of alternatives such as $A$ and $B$, two goals such as $g_1$ and $g_2$ should be swapped, where the consequence of $A$ on $g_1$ is minimal compared to any other goal and $g_2$ has the highest satisfaction level compared to any other goal.
Illustrative Example: Consider the consequences of $A_1$ in Table 6.1. Minimally satisfied goals are $G_1$, $G_6$, and $G_7$. Stakeholders would probably agree with improving such goals and in return sacrifice on goals that are maximally satisfied ($G_3$ and $G_4$).

Reusing Known Dominant and Dominated Alternatives for Other Pairs

Once an alternative is recognized as the dominant one, this knowledge can be reusable for deciding between other pairs of alternatives without additional swaps.

Rule 4, Reuse the known dominant and dominated alternatives to other pairs:

Consider two alternatives such as $A$ and $B$ where one possible consequences of $A$ and $B$ on a set of $m$ goals is \{a_1, a_2, ..., a_m\} and \{b_1, b_2, ..., b_m\}. Consider another possible consequence of $A$ and $B$ as \{a'_1, a'_2, ..., a'_m\} and \{b'_1, b'_2, ..., b'_m\}. If $A$ is recognized as the dominant one in the pair of $A$ and $B$ for the first enumerated consequence, then $A$ is dominant for the second possible consequence too, iff:

- $\forall \ a_x$ and $a'_x \ a'_x \geq a_x$, and
- $\forall \ b_x$ and $b'_x \ b'_x \leq b_x$

Note the knowledge of known dominant and dominated alternatives is reusable to different alternatives as well. For example, consider four alternative solutions such as $A$, $B$, $C$, and $D$ and three goals $g_1$, $g_2$, and $g_3$, where consequences of alternatives on the goals are as:

$A = \{High, Medium, Low\}$

$B = \{Medium High, Low, High\}$

$C = \{High, Medium, Medium Low\}$

$D = \{Medium, Low, High\}$

Assume by applying a chain of swaps, the algorithm concludes that $A > B$. We can conclude that $C > D$ as well, because $C$ dominates $A$ on all three goals, and $B$ dominates $D$ on all three goals. Since dominance is a transitive relation, we can conclude that $C$ dominates $D$.

The Automatic Even Swaps Algorithm

The automated even swaps algorithm applies the discussed rules iteratively to make one of the alternatives absolutely dominant. Given a set of goals, $G = \{g_1, g_2, ..., g_m\}$, and one of the
possible consequences of alternatives $A$ and $B$ on the goals \( \{a_1, a_2, \ldots, a_m\} \) and \( \{b_1, b_2, \ldots, b_m\} \), the automated Even Swaps algorithm for determining the optimum solution in the pair of alternatives consists of six steps. The steps are repeated until one of the alternatives is recognized as the dominant solution for the given possible consequence. Within these steps, the algorithm generates the next swap, asks for the swap from users, and stores user’s responses in a Knowledge Base (KB). A list of candidate goals to swap is kept as tuples in temporary array lists called $L$, $L'$, and $L''$.

While NOT ($A$ is dominant OR $B$ is dominant)

Step 1: Remove irrelevant goals

For all $g_x$ in $G$:

If $a_x = b_x$ Then remove $g_x$ from the list of goals

Step 2: Reuse swaps

For all swap in Swaps KB

If swap is reusable Then

Apply swap

Repeat Step 1: Remove irrelevant goals after swapping

Step 3: Apply Rule 1, create a dominant alternative

For all $g_x$ and $g_y$ in $G$

If $a_x < b_x$ AND $a_y > b_y$ Then $T$.add($g_x, g_y$)

If $a_x > b_x$ AND $a_y < b_y$ Then $T$.add($g_y, g_x$)

Step 4: Apply Rule 2, pick the most reusable swap

For all $(g_x, g_y)$ in $T$

$T''$.add(the most reusable $(g_x, g_y)$)

If NOT exist a reusable $(g_x, g_y)$ in $T$ Then $T' = T$.

Step 5: Apply Rule 3, swap minimally and maximally satisfied goals

For all $(g_x, g_y)$ in $T''$

If $a_x$ is Min AND $a_y$ is Max Then $T''$.add($g_x, g_y$)

Step 6: Ask the swap from stakeholders

$(g_x, g_y) =$ random tuple in $T''$

in swap $(g_x : a_x \rightarrow a'_x \leftrightarrow g_y : a_y \rightarrow a'_y)$ Ask $a'_y$ value

$sat(g_x, A) = a'_x$, $Sat(g_y, A) = a'_y$

$SwapsKB$.add $(g_x : a_x \rightarrow a'_x \leftrightarrow g_y : a_y \rightarrow a'_y)$
End While

Return the dominant alternative

6.2 Semi-Automated Even Swaps with the Mixture of Qualitative and Quantitative Values

Section 6.1 of this chapter tackles decision analysis in the absence of any measurement. This section focuses on the situations where some of the decision criteria are measured quantitatively and some are evaluated qualitatively. Ideally, analysts would like to quantitatively measure (estimate) consequences of alternative solutions on various requirements. Some requirements can be refined into measurable variables, while many non-functional requirements have a soft nature, which makes them hard to measure on an absolute scale. Such requirements can be evaluated by using qualitative values, such as ordinal Low, Medium, and High measures or by qualitative labels such as partially satisfied (√), sufficiently satisfied (√), partially denied (×), and fully denied (x).

Stakeholders may choose to evaluate some requirements numerically, and some requirements qualitatively, on different scales, e.g., absolute values, percentages, ordinal numbers, or qualitative labels. When different requirements are measured (or estimated) on different scales, normalizing inconsistent types of measures into a single scale is troublesome and may not result in a meaningful utility value. Besides, extracting correct numerical importance weights is time-consuming and labor-intensive, especially when several criteria need to be considered.

As discussed earlier, the Even Swaps method [64] circumvents the need to measure requirements and consequences of solutions. The Even Swaps method works by trading off one requirement for another, thus, it does not require eliciting importance weights of requirements for determining the most preferred solution. Requirements can be evaluated in a mixture of scales and by different measurement methods.

However, the Even Swaps method may fail in practice due to scalability issues: when several requirements and alternative solutions need to be considered, determining the right swaps to make among numerous possibilities is hard for human decision makers [116]. The Even Swaps...
process does not provide a systematic method or guideline for stakeholders to carry out the process, and recognizing the best pair of alternatives to analyze or conflicting goals to swap can be challenging for non-expert users. The Even Swaps process may lead to making numerous swaps in a tedious and long process. In what follows, the Even Swaps method is enhanced by an algorithm that:

- Semi-automates the Even Swaps process, in the sense that the process is still interactive with stakeholders while swaps are suggested by the algorithm and applied automatically.
- Decides which pair of alternatives should be compared in each step.

### 6.2.1 Illustrative Example

We illustrate our work with another scenario at MTO (See Section 6.1). Let us assume that MTO decision makers need to select an Intelligent Transportation System (ITS) among three hypothetical alternative proposals. We refer to these alternatives as $A_1$, $A_2$, and $A_3$. Some of these factors are measurable; for example, the implementation cost of solutions is known and performance can be accurately estimated. On the other hand, the security level and usability of these design solutions are not numerically measurable until the system is actually deployed.

**Decision Criteria:** To select a solution among alternative ITSs, stakeholders consider nine main criteria, indicating costs, security level, usability, and performance of each proposal:

- $T_1$: Time between a change in the traffic and notifying the operator
- $T_2$: Time between requesting a feed and showing the video on the monitor
- $S_1$: Authorized control of cameras (security)
- $P_1$: Percentage of the time cameras are connected
- $G_1$: Ease of viewing the video feeds
- $G_2$: Ease of locating the cameras on the road
- $G_3$: Ease of changing the cameras settings
• $C_1$: Implementation costs

• $C_2$: Maintenance costs

Table 6.2: The consequence table of alternative ITS for the MTO scenario

<table>
<thead>
<tr>
<th>Alternative</th>
<th>$T_1$</th>
<th>$T_2$</th>
<th>$S_1$</th>
<th>$P_1$</th>
<th>$G_1$</th>
<th>$G_2$</th>
<th>$G_3$</th>
<th>$C_1$</th>
<th>$C_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>2 ms</td>
<td>3 ms</td>
<td>$\checkmark$</td>
<td>99%</td>
<td>H</td>
<td>M</td>
<td>$\checkmark$</td>
<td>$$2\ m$</td>
<td>MH</td>
</tr>
<tr>
<td>$A_2$</td>
<td>3 ms</td>
<td>4 ms</td>
<td>$\checkmark$</td>
<td>94%</td>
<td>L</td>
<td>M</td>
<td>$\checkmark$</td>
<td>$$1.5\ m$</td>
<td>M</td>
</tr>
<tr>
<td>$A_3$</td>
<td>1 ms</td>
<td>2 ms</td>
<td>$\checkmark$</td>
<td>95%</td>
<td>ML</td>
<td>L</td>
<td>$\checkmark$</td>
<td>$$1.8\ m$</td>
<td>M</td>
</tr>
</tbody>
</table>

**Consequence Table:** We aggregate consequences of alternatives on criteria of decision making in a table, which we refer to as the consequence table. Table 6.2 shows the consequence table for $A_1$, $A_2$, and $A_3$, and the impacts of these alternatives on the decision criteria. Table 6.2 also specifies which criteria need to be minimized and which one need to be maximized.

A consequence table can contain heterogeneous data, i.e. different goals that are evaluated in different scales and by different techniques. Some of the criteria are measurable variables that need to be minimized or maximized. For example,

• $T_1$ and $T_2$ are measurable performance variables and stakeholders are able to estimate **Time between a change in the traffic and notifying the operator** ($T_1$) in milliseconds based on the properties and specification of $A_1$, $A_2$, and $A_3$.

• $S_1$ is a security goal that is not directly measurable, and $P_1$ is security variable that stakeholders can estimate.

• $G_1$, $G_2$, and $G_3$ are usability goals, and stakeholders do not have enough information to quantitatively measure them, so they evaluate consequences of $A_1$, $A_2$, and $A_3$ on $G_3$ by using qualitative labels of partially satisfied ($\checkmark$), sufficiently satisfied ($\checkmark$), partially denied ($\times$), and fully denied ($\times$). Some requirements such as $G_1$ and $G_2$ are evaluated in the ordinal scale of Low, Medium Low, Medium, Medium High, and High (which are abbreviated as L, ML, M, MH, and H respectively).
• $C_1$ and $C_2$ are cost factors. Implementation costs are known, while maintenance costs can only be estimated qualitatively. We illustrate our method by analyzing trade-offs among ITS requirements and selecting a solution among $A_1$, $A_2$, and $A_3$.

### 6.2.2 The Algorithm Overview

Given a consequence table, the proposed algorithm suggests a chain of swaps to determine the overall best alternative. The algorithm consists of several Even Swaps cycles. These cycles continue until one alternative remains, which is the best solution overall:

1. In the beginning of each cycle, the algorithm selects a pair of alternatives for the next Even Swaps process.
2. The automated swap suggestion algorithm takes the pair of alternatives and suggests a chain of swaps to stakeholders intending to find the preferred alternative in the pair.
   - The chain of swaps continues until one of the solutions dominates the other.
3. The dominant alternative is kept in the list of solutions, and the dominated alternative is removed, and will not be considered in the rest of the decision analysis process.
4. The algorithm selects another pair of alternatives to compare and a new cycle starts.

### 6.2.3 Selecting a pair of Alternatives for Even Swaps

When $n$ alternatives remain in the decision process, $\frac{n(n-1)}{2}$ different pairs of alternatives can be analyzed in the next Even Swaps process. Does it make a difference which pair is analyzed next in the Even Swaps process? In this section, we discuss two main reasons why the choice of alternative pairs would affect the performance of the algorithm and usability of interactions with stakeholders: 1) Some pairs may require fewer swapping steps to find the dominant solution than other pairs. 2) Two alternatives that have highly similar consequences on decision criteria will not be easily distinguishable in the swapping process.

To reduce the number of even swap steps and make the swapping process easier for decision makers, the pair of alternatives for the next Even Swaps process needs to be carefully picked. We
set two main criteria for selecting a suitable pair of alternatives: 1) Minimum number of swaps needed to make one alternative dominant, 2) Maximum dissimilarity between consequences of the alternatives (so in the swapping process, stakeholders can easily distinguish the alternatives).

**Criterion 1: Minimum number of swaps**

The algorithm aims to select a pair of alternatives that requires a minimum number of swaps to make one of the alternatives dominated. Out of the pair of $A$ and $B$, if $A$ is a better alternative for $n_1$ goals (where $a_x > b_x$) and $B$ is the better alternative for $n_2$ goals (where $b_x > a_x$), then the number of swaps needed to make one of the alternatives dominated is at least $\min(n_1, n_2)$.

For example, among three ITS alternatives, $A_1$, $A_2$, $A_3$, the possible pairs of alternatives for the Even Swaps process are $(A_1, A_2)$, $(A_1, A_3)$, and $(A_2, A_3)$. Table 2 shows that in the pair of $(A_1, A_2)$, $A_1$ is a better solution for five goals and $A_2$ is better for three goals. $(A_1, A_2)$ and $(A_2, A_3)$ both require at least three swaps to create a dominated alternative, so we should pick one of these pairs for the next Even Swap cycle. But minimizing the number of swaps is not the only goal; we need to pick a pair of alternatives for the next Even Swaps process that stakeholders are able to easily distinguish their consequences, and swap goals.

<table>
<thead>
<tr>
<th>Pairs</th>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$\triangle$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(A_1, A_2)$</td>
<td>5</td>
<td>3</td>
<td>-</td>
<td>2.93</td>
</tr>
<tr>
<td>$(A_1, A_3)$</td>
<td>5</td>
<td>-</td>
<td>4</td>
<td>2.99</td>
</tr>
<tr>
<td>$(A_2, A_3)$</td>
<td>-</td>
<td>3</td>
<td>4</td>
<td>2.57</td>
</tr>
</tbody>
</table>

**Criterion 2: Maximum divergence between consequences**

The underlying idea of this criterion for selecting a pair of alternatives is to select a pair whose consequences are easily differentiable, so when stakeholders are asked to make a swap, consequences of alternatives are divergent enough that stakeholders can make an explicit trade-off between the requirements. To illustrate this idea, consider the ITS example where we need to determine the dominant solution in the pair of $A_2$ and $A_3$ by swapping $G_2$ and $P_1$. If
stakeholders are asked how much they would sacrifice on $G_2$, for increasing $P_1$ 1% from 94% to 95%, they may not find this 1% improvement enough to relax the satisfaction level of $G_2$. This problem stems from an insignificant difference between $A_2$ and $A_3$’s consequence on $P_1$: 94% is barely distinguishable from 95%. The algorithm therefore avoids selecting a pair of alternatives for the Even Swaps process that have similar consequences on requirements.

In order to measure how distinguishable a pair of alternatives is, we calculate a “distance factor”, $\triangle(A, B)$ between consequences of $A$ and $B$ on goals:

$$\triangle(A, B) = \frac{1}{\max g_x} \sum_{x=1}^{m} |a_x - b_x|$$

The delta of consequences on goal $g_x$ ($|a_x - b_x|$) is divided by the maximum satisfaction level on $g_x$ in the consequence table ($\max g_x$). In this way, the delta values are normalized to a value in the range of 0 to 1, so they can be summed up into one value. For example, the distance factor of $(A_1, A_2)$ is calculated as:

$$\triangle(A_1, A_2) = \frac{2-3}{3} + \frac{3-4}{4} + \frac{|\kappa - \kappa'|}{99} + \frac{|99-94|}{99} + \frac{|H-L|}{H} + \frac{|M-M|}{M} + \frac{\kappa - \kappa'}{\kappa} + \frac{2-1.5}{2} + \frac{|MH-M|}{MH}$$

**Calculation Method:** Calculating deltas for absolute measures is straightforward. However, measuring the differences of two qualitative labels as $|\kappa - \kappa'|$ is troublesome, because these labels do not bear a quantitative implication. Based on the relative rankings ($\varphi > \kappa > \kappa' > \kappa''$) we can conclude $|\kappa - \kappa'| > |\kappa - \kappa'|$. Semantically, $\varphi$ is the opposite of $\kappa$ and $\kappa''$ is the opposite of $\kappa'$. Thus, we assume $|\kappa - \kappa'| = |\kappa - \kappa''|$ and $|\kappa - \kappa'| = |\kappa - \kappa'|$. In order to calculate the relative distance factor, we assign an arbitrary numerical representation to $\varphi$, $\kappa$, $\kappa'$, and $\kappa''$ that solves the ordering relations we assumed above. For example, +2, +1, -1, -2 or 4, 3, 2, 1 are possible arbitrary representations for the qualitative labels of $\varphi$, $\kappa$, $\kappa'$, and $\kappa''$ respectively (we use 4, 3, 2, 1 in the algorithm).

In the scale of Low to High, Medium is in the middle of Low and High (their average). Medium Low and Medium High are other finer grained middle values. Thus, the values of Low to High can be represented by any interval scale that fits the discussed relations (we used the interval scale of 1 to 5 in the calculations).

**Imprecision:** The underlying purpose of calculating the deltas is to measure the extent of dissimilarity of two alternatives on each goal. Note that normalizing these values does not
provide us with means for calculating a meaningful utility for the alternatives. For example, the percentage of the time that cameras are connected ($P_1$) is 95% for $A_3$ (Table 6.2); however, the utility of $A_3$ on $P_1$ is not reflected by the relation of 95% to the practical maximum connectivity percentage (99%) or absolute maximum connectivity (100%). If availability of the connection to cameras is highly critical, the utility of $A_2$ on $P_1$ is not proportional to $\frac{95}{99}$, and the utility of any connectivity lower than 99% (for example) could even be zero.

With the same argument, one can claim that Formula 1 does not measure the dissimilarity of two alternatives correctly, because if the utility of 95% is zero, the utility of $A_2$ on $P_1$ (94%) is also zero, and although on the face they are dissimilar, $A_2$ and $A_3$ are indifferent with respect to $P_1$. Nevertheless, the distance factors of the pairs of alternatives are comparable because all distance factors are calculated with equal imprecision. Thus if $\Delta(A_1, A_2) = 2.93$ and $\Delta(A_2, A_3) = 2.57$ (Table 6.3), we conclude that consequences of $A_1$ and $A_2$ are more dissimilar than the consequences of $A_2$ and $A_3$, although both numbers are inaccurate. In the next section, we introduce the automated swap suggestion algorithm to select the dominant solution in the pair of $(A_1, A_2)$.

6.2.4 Automatically Suggesting Even Swaps

In each cycle of the algorithm, two alternatives are analyzed by applying a chain of swaps, until one becomes dominated. One of the main goals of the proposed algorithm is to reduce the number of swapping steps through reusing swaps from previous cycles. We adopt and adapt the swap suggestion rules which we introduced in Section 6.1 for automatically suggesting swaps between goals where some are measured quantitatively and some are evaluated qualitatively.

Creating an Absolute Dominant Alternative

The swapping process intends to make one of the alternatives dominated or dominant (Rule 1). For example, in the Even Swaps process, between the pair of $A_1$ and $A_2$, $A_1$ is a better solution with respect to $T_1$, $T_2$, $P_1$, $G_1$, and $G_3$, and with respect to $S_1$, $C_1$, and $C_2$, $A_2$ is the better solution. Note that the satisfaction level of both alternatives on $G_2$ is Medium, so $G_2$ is
an irrelevant goal for deciding between \( A_1 \) and \( A_2 \).

To reduce the number of swaps needed in the next step, a goal such as \( g_x \) needs to be swapped with a goal such as \( g_y \) where \( g_x \in \{ S_1, C_1, C_2 \} \) and \( g_y \in \{ T_1, T_2, P_1, G_1, G_3 \} \). Therefore, 15 tuples in the form of \((g_x, g_y)\) exist as the candidate pair of goals to be swapped. By making a swap between such goals, one of three goals among \( S_1 \), \( C_1 \), and \( C_2 \) becomes irrelevant; thus, in the next step, at least two more swaps are needed to make one of the alternatives dominant.

**Suggesting the Most Reusable Swap**

When stakeholders make a swap, their value trade-offs can be reused for another alternative, without further consultation with human stakeholders (if the goals are the same and the satisfaction level of goals in the swap are the same with the new alternative).

None of those 15 candidate swaps for \( A_1 \) and \( A_2 \) are reusable for the consequences of \( A_3 \). To illustrate the concept of reusability, consider three hypothetical solutions \( A \), \( B \), and \( C \), and three goals \( g_1 \), \( g_2 \), and \( g_3 \), which we aim to maximize. Let us assume the consequences of alternatives on the goals are:

\[
A = \{ H, M, L \}, \quad B = \{ MH, L, H \}, \quad C = \{ H, M, ML \}
\]

Consider the swap \((g_1 : H \to MH \leftrightarrow g_2 : M \to x)\) for deciding between \( A \) and \( B \). Assume the stakeholder has previously agreed to reduce the satisfaction level of \( A \) on \( g_1 \) from \( H \) to \( MH \), and in return, increase the satisfaction level of \( g_2 \) from \( M \) to \( x = H \). This swap is reusable for deciding between \( B \) and \( C \) as well, where the satisfaction level of \( C \) on \( g_1 \) and \( g_2 \) can be modified according to this swap without the need for asking another swap from stakeholders.

**Suggesting Easy Swaps**

In addition to considering swaps reusability, the algorithm intends to suggest swaps that decision stakeholders would be willing to make. We discussed earlier that stakeholders may easily agree to increase a goal that is not sufficiently satisfied and compensate it with decreasing the satisfaction level of a requirement that is highly satisfied, intending to reach a balance among software requirements.
In addition, if consequences of two alternatives on the first goal of the swap are close, with an insignificant change on the consequence of one alternative, that goal can become irrelevant. In this way, the goals that do not differentiate alternatives are eliminated from the problem earlier. Since these goals may not affect the final decision dramatically, they can be eliminated earlier in the decision analysis process.

To compensate for the insignificant change on the first goal, the algorithm searches for another goal, on which consequences of alternatives are highly differentiable. To understand the philosophy of this rule, assume \( A \) is the dominant alternative (in the pair of \( A \) and \( B \)) with respect to all goals, except a goal such as \( g_x \). By removing \( g_x \) in a swap, \( A \) can become the overall dominant. Suppose the satisfaction level of \( g_x \) is increased, and as an even compensation, the satisfaction level of another goal such as \( g_y \) is reduced: \( (g_x : a_x \rightarrow a'_x \Leftrightarrow g_y : a_y \rightarrow a'_y) \), where \( a'_x > a_x \) and \( a'_y < a_y \). By this swap, \( g_x \) has become irrelevant and is removed from the list of decision criteria, but it is possible that \( A' \) is now dominant with respect to all goals except \( g_y \). This means the value of \( a_y \) has reduced so much that now \( B \) is the dominant one with respect to \( g_y \). In this situation, the swap has not been effective for creating a dominant alternative and still another swap is needed to make solution \( A \) dominant. If the impact of \( A \) on \( g_y \) is much higher than the impact of \( B \) on \( g_y \), it is more probable that after reducing \( a_y \) to \( a'_y \), still \( A' \) is the better solution than \( B \) with respect to \( g_y \). That is why we search for a goal, on which consequences of alternatives are highly differentiable.

**Rule 3**, **make the easiest swap**: This rule summarizes the characteristics which we discussed above. When comparing the pair of \( A \) and \( B \), two goals such as \( g_1 \) and \( g_2 \) should be swapped where the satisfaction level of \( A \) on \( g_1 \) is minimum compared to any other goal, and at the same time, \( |a_1 - b_1| \) is minimum. The second goal, \( g_2 \), has the highest satisfaction level compared to other goals, and at the same time \( |a_2 - b_2| \) is maximum.

In this way, the satisfaction level of \( g_1 \) is relatively low, and the satisfaction level of \( g_2 \) is relatively higher than other goals; thus stakeholders probably agree to increase consequences of \( A \) on \( g_1 \) (from \( a_x \) to \( b_x \)) and compensate it with reducing the satisfaction level of \( g_2 \). In addition, since consequences of \( A \) and \( B \) on \( g_1 \) are close, with an insignificant change on \( g_1 \), it can become an irrelevant goal. Consequences of \( A \) and \( B \) on \( g_2 \) are divergent (\( |a_2 - b_2| \) is maximum); thus,
by applying the swap and reducing the satisfaction level of \( g_2 \), \( A \) (or \( B \)) remains dominant.

How can these objectives be formulated? We define a distance factor for alternatives for a given goal, which aggregates these desired properties into a value. The distance factor on the goal \( g_x \) is \( \triangle(A, B, g_x) \) and calculated as:

\[
\triangle(A, B, g_x) = \left| \frac{a_x - b_x}{\max_{g_x}} \right| + \frac{a_x}{\max_{g_x}}
\]

where \( \max_{g_x} \) is the maximum satisfaction level of \( g_x \) in the consequence table.

The purpose of rule 3* is to swap a goal with minimum satisfaction level (minimum \( a_x \) in formula (2)) and the smallest distance from \( B \) (minimum \( |a_x - b_x| \) in formula (2)) with a goal that has maximum satisfaction level and maximum distance from \( B \).

This formula, therefore, does not reflect the minimum and maximum satisfaction for the goals and variables that stakeholders aim to minimize. For such goals, the higher \( a_x \) is, the less the satisfaction level of \( g_x \) is. Hence, if \( g_x \) is a goal or variable that needs to be minimized, then the algorithm replaces \( a_x \) with \( \frac{1}{a_x} \) in formula (2). For example, \( \text{Sat}(T_1, A_1) = \frac{1}{2} \), \( \text{Sat}(T_1, A_2) = \frac{1}{3} \), and \( \text{Sat}(T_1, A_3) = \frac{1}{4} \), so \( \max_{T_1} = 1 \). (To be able to apply this formula, we assume that all goals and variables have a non-zero value)

Figure 6.4 shows the distance factors of \( A_1 \) and \( A_2 \) on the goals in Table 6.2. For example, the distance factors of \( A_1 \) and \( A_2 \) on \( T_1 \) and \( S_1 \) are:

\[
\triangle(A_1, A_2, T_1) = \left| \frac{1}{2} - \frac{1}{3} \right| + \frac{1}{2} = 0.67
\]

\[
\triangle(A_1, A_2, S_1) = \left| \frac{1}{4} - \frac{1}{3} \right| + \frac{1}{4} = 1
\]

Figure 6.4: Distance factors of alternatives \( A_1 \) and \( A_2 \) on ITS goals

Previously, we identified 15 candidate swaps for analyzing \( A_1 \) and \( A_2 \): these 15 candidates are tuples like \((g_x, g_y)\), where \( g_x \) can be any of \( S_1, C_1, C_2 \), and \( g_y \) can be any of \( T_1, T_2, P_1, G_1, G_3 \). Based on rule 3*, \( g_x \) shall have the minimum distance factor. Since the distance factor of \( A_1 \) and \( A_2 \) on \( S_1, C_1 \), and \( C_2 \) is 1, \( g_x \) can be any of these three goals. Rule 3* also
indicates that $g_y$ shall have the maximum distance factor; therefore, $G_1$ is the best choice for $g_y$ (in Figure 6.4, $\triangle(A_1, A_2, G_1) = 1.8$ which is the maximum among $T_1, T_2, P_1, G_1, G_3$). In sum, by applying rule 3, candidate goal tuples for the next swaps are reduced to three tuples: $(C_1, G_1), (C_2, G_1), (S_1, G_1)$.

Which of these three possible goals should be suggested for the next swap? The final rule for selecting the next swap suggests swapping goals that are measured in more granular and accurate scales, because dealing with tangible factors is easier for stakeholders [116, 115, 78]. For example, costs in terms of money is more tangible than the usability level expressed as Medium, Low, High. In this regard, rule 4* states:

**Rule 4*, pick preferred scales:** Goals that are measured in absolute values are preferred to the goals measured in percentages, percentages are preferred to ordinal values, and the least preferred scale is qualitative labels.

For example, based on rule 4*, the scale of $C_2$ (Low, Medium, High) is preferred to the scale of $S_1$ (qualitative labels) and the scale of $C_1$ is preferred to $C_2$ (million dollars). Therefore, among three options ($(C_1, G_1), (C_2, G_1), (S_1, G_1)$), the algorithm selects $C_1$ and $G_1$ to be swapped, and asks the value of $x$ in the swap: $(C_1 : $2m → $1.5m ⇔ G_1 : H → x)$.

The aim of this swap is to elicit how much stakeholders would give up on $G_1$ for paying a lower price ($C_1$), so $C_1$ is reduced from $2m$ to $1.5m$, and as a compensation, $G_1$ is decreased from $H$ to a lower level ($x$). Suppose stakeholders agree with $x = L$. The consequence table is revised with these new values (the satisfaction level of $G_1$ under the consequence of $A_1$ is now $L$, and the implementation cost of $A_1$ is now $1.5$ million). $C_1$ can be removed from the list of goals that need to be considered for deciding between $A_1$ and $A_2$, because after the above swap $A_1$ and $A_2$ are indifferent over $C_1$. (Not deliberately, the swap makes $G_1$ irrelevant as well.) The chain of swaps and removing irrelevant goals continues until one of the alternatives becomes dominated.
Reusing Known Dominant and Dominated Alternatives to Other Pairs

We discussed in Section 6.1.4, once an alternative is decided as the dominant one, this knowledge might be reusable for deciding between other pairs of alternatives without the need for the even swaps process. We apply this reusing method to the decision analysis problems discussed in this section as well.

The Semi-Automatic Even Swaps Algorithm

Assume a consequence table that includes a set of goals \( G = \{g_1, g_2, \ldots, g_m\} \), and two alternatives \( A \) and \( B \) to be analyzed in the next cycle of the algorithm. The algorithm suggests swaps, gets the stakeholder’s input, applies the swap, and stores the swaps in a Knowledge Base (KB). In each step, the algorithm keeps the tuples of candidate goals for the next swap in temporary array lists of \( L, L', L'', \) and \( L_f \). The process for determining the optimum solution between \( A \) and \( B \) consists of eight main steps as follows.

**While NOT(\( A \) is dominant OR \( B \) is dominant)**

1. **Step 1: Remove irrelevant goals**
   
   For all \( g_x \) in \( G \):
   
   If \( a_x = b_x \) Then remove \( g_x, a_x, b_x \) from \( G, A, B \)

2. **Step 2: Reuse swaps**
   
   For all \( swap \) in \( SwapsKB \)
   
   If \( swap \) is reusable to \( A \) OR \( B \) Then
   
   Apply \( swap \)
   
   Repeat Step 1: Remove irrelevant goals

3. **Step 3: Apply Rule 1, create a dominance situation**
   
   For \( x = 1 \) To \( m \)
   
   If \( a_x > a_y \) AND \( b_x > b_y \) Then \( L.add((g_x, g_y)) \)

4. **Step 4: Apply rule 2, find the most reusable swaps**
   
   For all \( (g_x, g_y) \) in \( L \)
   
   \( L'.add(\)the most reusable \( (g_x, g_y))\)
   
   If NOT exist a reusable \( (g_x, g_y) \) in \( L \) Then \( L' = L \)

5. **Step 5: Apply rule 3\(^*\), make the easiest swaps**
For all $g_x$ in $G$ ($1 \leq x \leq m$)

If $g_x$.Target = MINIMIZE Then

For all Alternatives $A_i$ in Consequence Table

$Sat(g_x, A_i) = \frac{1}{c_x}$

For all $(g_x, g_y)$ in $L'$

If $\frac{|a_x-b_x|+a_x}{\max_{x \in x}}$ is Min AND $\frac{|a_y-b_y|+a_y}{\max_{y \in y}}$ is Max Then

$L''.add((g_x, g_y))$

Step 6: Apply rule $4^*$, select the preferred scales

For all $(g_x, g_y)$ in $L''$

If $g_x$ and $g_y$ have the most preferred scales

Then $L_f.add((g_x, g_y))$

Step 7: Ask the swap from stakeholders

$(g_x, g_y) = \text{random tuple in } L_f$

Ask value of $a'_y$ in swap $(g_x : a_x \rightarrow b_x \Leftrightarrow g_y : a_y \rightarrow a'_y)$

$Sat(g_x, A)=b_x$

$Sat(g_y, A)=a'_y$

$SwapsKB.add(g_x : a_x \rightarrow b_x \Leftrightarrow g_y : a_y \rightarrow a'_y)$

End While

If $A$ is dominant OR $B$ is dominant

Step 8: Reuse the dominance situation

For all Alternatives $A_x, A_y$ in Consequence Table

If $A, B$ dominance is reusable in $A_x, A_y$ Then

Apply the dominance to $A_x, A_y$

6.3 Case Studies, Experiments, and Evaluation

This section focuses on evaluating the utility and usefulness of two decision aid methods in Section 6.1 and 6.2. First we report on the results of the case study within the Ministry of Transportation Ontario (MTO). In order to evaluate the correctness of the decision made by our methods, we use two quantitative case studies ([73, 101]) that were adopted in other research contributions as the test case. This section compares the results of the methods used
in the original contributions with the results of our methods. This comparison shows whether the proposed methods in this thesis determine the best solution correctly and with reasonable effort (reasonable number of swap enquiries).

6.3.1 The Case Study at MTO

We applied the proposed method in this paper to analyze the trade-offs of switching to the ONE-ITS system from the existing traffic monitoring system at MTO.

Applying the comparison-based method: The MTO expert who collaborated in this study described the goals of his department for employing traffic monitoring systems. In a separate interview session the MTO expert compared the alternative systems with respect to a number of goals.

Figure 6.3 in Chapter 6 shows the consequences of alternative solutions on goals of the VMS application. The tool generated 90 different possible consequences for the pair of alternatives. The automated Even Swaps algorithm was used to decide on the dominant alternative system for each of the 90 cases, and in total 26 swaps were asked from stakeholder (12 swaps were denied by the users as not possible to make). The final decision required further information gathering about the consequences of alternatives on development costs. Finally we concluded that the ONE-ITS \( (A_1) \) is a better choice than the current system. The MTO expert confirmed the results, which provides a support for the correctness of the decision suggested by our method.

Comparison with AHP

In order to provide evidence that the suggested solution by the tool is actually the best available choice, we analyzed the MTO decision scenario by applying the AHP method as well. AHP is a preference elicitation method based on comparing alternatives. In using AHP, the scale of priorities is derived from pair-wise comparison of goals preferences. Paired comparisons are made by specifying the preference of each criteria over every other one. These preferences are expressed on the AHP absolute fundamental scale of 1-9, where 1 indicates that two elements are of equal priority and 9 indicates favoring one element over another with the highest order.
of affirmation.

In order to make the proposed method comparable with the AHP, we elicited stakeholders’ preferences from the even swaps. We observed that stakeholders’ preferences are: \( G_1 > G_7 > G_2 > G_6 > G_5 \) and preference of \( G_5 = G_4 = G_3 \), based on the swaps made previously. The pair-wise comparison of preferences over goals was done based on the ranking of preferences. Figure 6.5 gives two different paired comparisons of goals. The last column in the matrices shows the final importance weight calculated by the Eigenvalue of the matrices.

**Results:** By using the importance weights of goals in the matrix (a) in Figure 6.5 and calculating the final utility of \( A_1 \) and \( A_2 \), \( A_1 \) (the ONE-ITS) is recognized as the overall optimum solution. We slightly modified some of the AHP pair-wise comparisons (highlighted in Figure 6.5 (a) and (b)). By using the new importance weights in the matrix (b) in Figure 6.5, \( A_2 \) (the existing system) is determined as the overall optimum solution. This contradiction (also in contrast with the results of our method) shows that the final decision suggested by the AHP is highly sensitive to the ordinal paired comparisons. For example, by specifying that the preference of \( G_1 \) to \( G_3 \) is 7 in the AHP ordinal scale (Matrix (b)), the final importance weight for \( G_1 \) is calculated 7 times greater than \( G_3 \)’s weight. This means a qualitative description of the comparison in an ordinal scale is transformed to its proportional numerical representation. The numerical representation may not reflect the actual intentions behind those qualitative comparisons made by the human user; for instance, \( G_1 \)’s importance weight may not necessarily be 7 times greater than \( G_3 \)’s, and by converting the human description of the preferences to AHP ordinal scale, the comparisons are exaggerated.

![Figure 6.5: Pair-wise comparison of goals preferences using AHP.](image-url)
6.3.2 Test Cases from Previous Work

We worked out two decision analysis case scenarios that were originally analyzed in two requirements decision analysis proposals [73, 101].

Test Case One ([73])

In [73], four alternatives are compared with respect to five main criteria: Functionality, Training, Process, Time, and Cost. Alternatives are analyzed using the SMART [146] and AHP methods. The assessment of alternatives and importance weights of criteria are provided in the original work and a utility value for each alternative \( A_1, A_2, A_3, A_4 \) is calculated accordingly (the left hand side of Figure 6.6). In this case study [73], \( A_2 \) is recognized as the overall optimum choice.

**Applying the Comparison-based Method:** The comparison of alternatives on the interval scale corresponds to their numerical assessments given in the table in Figure 6.6. For example, \( \frac{A_1}{A_2} = L \) for the Functionality because the utility of \( A_1 \) and \( A_2 \) are 0.9 and 0.7, and their difference (0.2) on the scale of 0-1 matches the value of Low in the scale of Low to High. By comparing \( A_1 \) and \( A_2 \) 600 possible consequences are generated by the tool, and through three swaps, \( A_2 \) is recognized as the better choice. \( A_2 \) dominates \( A_3 \), thus, the tool does not need to analyze possible consequences or make any swap. Then \( A_2 \) is compared with \( A_4 \). On three criteria, the difference between \( A_2 \) and \( A_4 \) is insignificant (Low), and thus, the tool generated several possible consequences (2000 cases). In total 18 swaps are made to identify the best choice. By applying our method, we identified \( A_2 \) as the best solution (the right hand side of Figure 6.6).

**Results:** The case study showed that our approach can perform better than purely numerical methods, when consequences of alternatives on decision criteria are easily distinguishable. For example, by comparing \( A_1 \) and \( A_2 \), 600 possible consequences are created, but only three swaps are needed to make a decision between \( A_1 \) and \( A_2 \). In a numerical approach, given five criteria, at least five importance weights need to be extracted, while our method required only three swaps. On the other hand, the insignificant difference between two alternatives makes
decision analysis difficult. For example, to analyze $A_2$ and $A_4$, due to the insignificant difference between the consequences of $A_2$ and $A_4$ on three criteria, 2000 possible consequences were created, and 18 swaps were needed to identify the dominant solution.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>w</th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>A4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Funct.</td>
<td>0.17</td>
<td>0.9</td>
<td>0.7</td>
<td>0.7</td>
<td>0.5</td>
</tr>
<tr>
<td>Cost</td>
<td>0.33</td>
<td>0.2</td>
<td>0.9</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Time</td>
<td>0.25</td>
<td>0.2</td>
<td>0.6</td>
<td>0.4</td>
<td>0.8</td>
</tr>
<tr>
<td>Train</td>
<td>0.17</td>
<td>0.3</td>
<td>0.7</td>
<td>0.6</td>
<td>1</td>
</tr>
<tr>
<td>Process</td>
<td>0.08</td>
<td>0.5</td>
<td>0.7</td>
<td>0.6</td>
<td>1</td>
</tr>
<tr>
<td>Total</td>
<td>0.360</td>
<td>0.775</td>
<td>0.551</td>
<td>0.698</td>
<td></td>
</tr>
</tbody>
</table>

Figure 6.6: Applying the comparison-based method to a test case borrowed from [73]

**Applying the Automated Even Swaps Algorithm:** In the previous part, we analyzed the decision analysis in [73], in the situation where we cannot measure alternatives’ consequences and we cannot elicit numerical importance weights. In this section, we assume alternatives are evaluated on the scale of 0 to 1 as the table in Figure 6.6, but importance weight of criteria are not available. To analyze the decision problem in this situation, we applied the Even Swaps method to determine the overall best solution.

The numerical importance weights in the original example are used to interpret the preferences over goals. For example, the importance weights of Time and Functionality in [73] are 0.25 and 0.17, so Time is more important than Functionality. Thus, in the swap

\[
\begin{align*}
(Time : 0.2 \rightarrow 0.4 & \iff Functionality : 0.9 \rightarrow x) \\
\text{increasing Time from 0.2 to 0.4 shall be compensated with reducing Functionality from 0.9 to a level } x, \text{ where } 0.9 - x & > 0.4 - 0.2; \text{ because Time is more important (according to the importance weights). With a small change to Time from 0.2 to 0.4, Functionality needs to be reduced relatively more than 0.2, thus } x \text{ can be 0.6.}
\end{align*}
\]

In the above swap, if $x = 0.6$, the algorithm determines $A_3$ as the dominant alternative, and $A_1$ is removed from the consequence table. $A_2$ and $A_4$ are selected for the next Even Swap process, and by applying one swap, the algorithm concludes that $A_2$ is the dominant one, and $A_4$ is removed. The only swap to make this decision is as follows:
The weights of Time and Process are 0.25 and 0.08, so Time is highly preferred to Process; hence, increasing Time by 0.2 levels should be compensated with a large reduction in Process. Thus we apply $x = 0.2$, and by applying the swap, the algorithm concludes that $A_2$ is the dominant one, and $A_4$ is removed. In order to decide between $A_2$ and $A_3$, the Even Swaps process is not invoked, because $A_2$ dominates $A_3$, and the algorithm determines $A_2$ as the overall best alternative.

**Results:** The proposed method in this work asks the stakeholders to answer two swap queries, while in order to extract the numerical weights of five criteria at least five (direct assessment) queries are needed and by using AHP, at least ten comparison enquiries need to be asked from stakeholders. In this study, the Even Swaps process along with the automated swap suggestion algorithm reduces the burden and complication of extracting preferences and determines the best alternative correctly (compared to the final conclusion in [73]). Table 6.4 summarizes and compares the results of applying the Even Swaps algorithm with the results in the original contributions.

**Test Case Two ([101])**

The second test case is retrieved from preference model driven service selection method in [101]. In this case study, three alternatives $A$, $B$, and $C$ are compared against five different criteria (as shown at the left hand side of Figure 6.7) using AHP and a utility additive function. In this example case, alternative $B$ has the highest utility value.

**Applying the Comparison-based Method:** At the right hand side of Figure 6.7 the comparison of the $A$, $B$, and $C$ is given. The heuristic method generated 600 possible conse-
Figure 6.7: Applying the comparison-based method to a test case borrowed from [101]

quences for $A$ and $B$, and the sixth swaps found $A$ as the optimum solution. In total, the tool asked for ten swaps, four of which the user was not able to answer. In order to decide between $B$ and $C$, 250 possible consequences are generated and after asking for one swap, the algorithm selected $B$ as the optimum choice.

**Applying the Automated Even Swaps Algorithm:** In this section, we assume that alternatives’ consequences are measured, however, numerical importance weights of criteria are not available. First, the algorithm selects $B$ and $C$ for the Even Swaps process. The first swap suggested by the algorithm is as follows:

$$(\text{Management: } 0.748 \rightarrow 0.910 \Leftrightarrow \text{Reputation: } 0.904 \rightarrow x)$$

The value of $x$ depends on preferences of stakeholders, and since, Reputation is more important than Management, stakeholders would not sacrifice a lot on Reputation for improving Management from 0.748 to 0.910. For example, 0.162 unit of improvement on Management is exchanged for 0.05 unit of decrease on Reputation, and thus $x = 0.854$. By removing Management, the algorithm determines $B$ as the dominant solution. Two more swaps are made to make a decision between $B$ and $A$. In total, three swaps are made and the algorithm picks the best solution correctly ($B$).

**Results:** Our observations in the second test case are similar to the results of the first test case. The comparison-based method can even perform better than numerical methods, if consequences of alternatives are easily distinguishable. We can also conclude that when two alternatives have insignificant difference on several criteria, comparing their consequences and generating possible consequences will require numerous swap queries. Finally, if consequences
of alternatives can be evaluated in quantitative and qualitative ways, the Even Swaps method can perform better than extracting numerical importance weights.

### 6.3.3 Experiments

The comparison-based decision algorithm works based on the idea of generating possible consequences of alternatives with respect to their paired comparison over the criteria. If the scale of comparison has five different intervals and the pair of alternatives is compared over \( m \) criteria, the algorithm in the worst case, generates \( 5^m \) possible consequences (if the difference of the alternatives on every criterion is Low). By automating and improving the Even Swaps method, we estimate that many of these possible consequences are not directly examined, and the number of even swaps that stakeholders must make can be significantly less than \( 5^m \). This section presents the results of a series of experiments to statically and empirically evaluate the number of even swaps that stakeholders must make to evaluate and provide support for our claims.

**Experiment Design**

**Experiment Variables:** In this experiment, we run the comparison-based algorithm for diverse decision problem formulations. The controlled variables in the experiments are:

- Number of criteria
- Preference rankings of criteria
- Difference of the pair of alternatives on each criteria

The number of possible consequences that the algorithm generates grows exponentially with respect to the number of criteria. The number of possible consequences that the algorithm enumerates also depends on the difference of alternatives’ consequences on the criteria. We are interested in investigating the number of even swaps that the stakeholders need to make to resolve the trade-offs between criteria. Swaps highly depend on the preferences of stakeholders; thus, we need to assume a fixed preference ranking between the criteria to compare the number of swaps in different test cases. Dependant variables that we observe as the algorithm performance indicators are:
- Number of even swaps asked from stakeholders

- Number of swaps that decision stakeholders rejected to make

*Performance Remark:* The run time speed of the algorithm can be an important concern for specific scenarios such as real-time decision analysis; however, in the security requirements trade-offs problem, few strategic decisions need to be made in early stages of the problem, and we assume computation delays in a tool with human interactions can be tolerable. The main concern in this work is the number of swap queries that stakeholders must answer and the number of suggested swaps that stakeholders reject.

**Experiment Test Cases:** We compare the pair of hypothetical alternatives, \( A \) and \( B \), with respect to \( m \) different goals: \( G_1, G_2, \ldots, G_m \), for \( 2 \leq m \leq 8 \), for different set of test cases. We assume the fixed preference ranking of \( G_1 > G_2 > G_3 > \ldots > G_m \). We also assume \( A \) is a better solution for some of the goals and \( B \) is a better solution for some other.

Figure 6.8 illustrates the difference of \( A \) and \( B \) on the criteria. In the worst case (Figure 6.8 a), the difference of alternatives on every criteria is Low. Thus for \( m \) criteria, \( 5^m \) different possible consequences would be generated. In another challenging set of test cases (Figure 6.8 b), the difference of alternatives on every goal is Medium, thus \( 3^m \) cases would be generated. In an easier set of decision problems (Figure 6.8 c) the difference of alternatives’ consequences are in the range of Low to High, which in total requires generating and analyzing \( 5 \times 4 \times \ldots \times 1 \) possible consequences. In the best scenario (Figure 6.8 d), the pair of alternatives are highly distinguished and thus only one possible consequence can be generated.

![Figure 6.8](image-url)  

Figure 6.8: The difference of alternatives’ consequences on the decision criteria for different test cases
Experiment Results

We run each test case in Figure 6.8 with different number of criteria and different relationships between \(A\) and \(B\).

**Test case group a:** Table 6.5 summarizes the first group of test cases, where the difference of alternatives on every criterion is Low. Due to the exponential growth of possible consequences \((5^m)\), we only tested the algorithm for up to five criteria. The results of these tests show that making a decision in general and making even swaps specifically is hard when the consequences of alternatives on decision criteria are not easily distinguishable (Low difference on every criteria). The even swaps suggested by the tool mainly suggest a minimal change on one goal to be traded with a significant change on another, and thus, these types of even swaps were frequently rejected by the decision stakeholders.

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Possible Cases</th>
<th>(G_1)</th>
<th>(G_2)</th>
<th>(G_3)</th>
<th>(G_4)</th>
<th>(G_5)</th>
<th>Suggested alternative</th>
<th>Swaps</th>
<th>Rejected swaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m = 2)</td>
<td>25</td>
<td>(A) =L (B) =L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(A)</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td>(m = 3)</td>
<td>125</td>
<td>(A) =L (B) =L (B) =L (A) =L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(A)</td>
<td>20</td>
<td>1</td>
</tr>
<tr>
<td>(m = 4)</td>
<td>625</td>
<td>(A) =L (B) =L (B) =L (A) =L (A) =L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(A)</td>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>(m = 4)</td>
<td>625</td>
<td>(A) =L (B) =L (B) =L (B) =L (A) =L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(A)</td>
<td>65</td>
<td>6</td>
</tr>
<tr>
<td>(m = 5)</td>
<td>3125</td>
<td>(A) =L (B) =L (B) =L (B) =L (B) =L (B) =L</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>(A)</td>
<td>48</td>
<td>7</td>
</tr>
</tbody>
</table>

* \(A\) is the better choice for 600 possible consequences, and \(B\) is the better choice for 25 cases. The decision depends on verifying the possibility (validity) of those 25 cases.

We also observed that the number of swaps needed to make a decision is not only related to the number of possible consequences being enumerated. As shown in Table 6.5, making a decision between \(A\) and \(B\) over 4 criteria requires less swaps than when \(m = 3\). We also observed
that the number of swaps can depend on the balance between advantages and disadvantages of A and B over the criteria. In the second test case with 4 criteria, B is a better choice for majority of criteria that do not have high priority. Our hypothesis is that this unbalance makes the trade-offs complex to handle. In this test case, the algorithm does not provide a definite answer and returns a set of possible consequences for which B could be a better choice.

**Test case group b:** Table 6.6 summarizes the results of the second group of test cases where the difference of alternatives on every criterion is Medium. Making swaps and deciding over the alternatives is easier compared to the first group, because the alternatives’ impacts on the decision criteria are more distinguishable. The swaps are also easier to make (fewer swaps were rejected).

Table 6.6: Summary of test cases’ results for test case b, in Figure 6.8

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Possible Cases</th>
<th>G1</th>
<th>G2</th>
<th>G3</th>
<th>G4</th>
<th>G5</th>
<th>G6</th>
<th>G7</th>
<th>G8</th>
<th>Suggested alternative</th>
</tr>
</thead>
<tbody>
<tr>
<td>m = 2 9</td>
<td>( \frac{A}{B} = M ) ( \frac{B}{A} = M )</td>
<td>A 3 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 3 27</td>
<td>( \frac{A}{B} = M ) ( \frac{B}{A} = M )</td>
<td>A 11 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 4 81</td>
<td>( \frac{A}{B} = M ) ( \frac{B}{A} = M ) ( \frac{B}{A} = M ) ( \frac{A}{B} = M )</td>
<td>A 6 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 4 81</td>
<td>( \frac{A}{B} = M ) ( \frac{B}{A} = M ) ( \frac{B}{A} = M ) ( \frac{B}{A} = M )</td>
<td>* 35 5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 5 243</td>
<td>( \frac{A}{B} = M ) ( \frac{B}{A} = M ) ( \frac{A}{B} = M ) ( \frac{B}{A} = M ) ( \frac{B}{A} = M )</td>
<td>A 13 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 6 729</td>
<td>( \frac{A}{B} = M ) ( \frac{B}{A} = M ) ( \frac{B}{A} = M ) ( \frac{B}{A} = M ) ( \frac{A}{B} = M )</td>
<td>A 9 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 6 729</td>
<td>( \frac{A}{B} = M ) ( \frac{A}{B} = M ) ( \frac{B}{A} = M ) ( \frac{B}{A} = M ) ( \frac{B}{A} = M )</td>
<td>A 25 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 7 2187</td>
<td>( \frac{A}{B} = M ) ( \frac{B}{A} = M ) ( \frac{A}{B} = M ) ( \frac{B}{A} = M ) ( \frac{B}{A} = M )</td>
<td>A 22 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 8 6561</td>
<td>( \frac{A}{B} = M ) ( \frac{B}{A} = M ) ( \frac{A}{B} = M ) ( \frac{B}{A} = M ) ( \frac{B}{A} = M ) ( \frac{A}{B} = M )</td>
<td>A 14 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* A is the better choice for 15 possible consequences, and B is the better choice for 22 cases. For the rest of the cases, alternatives were equally preferred, thus the final decision depends on verifying the possibility (validity) of those 15 cases that A is a better choice.
This group of test cases confirms our conclusions about the previous set of test cases where the number of required swaps did not necessarily grow when the number of possible consequences increased. Repeating the tests for \( m = 4 \) and \( m = 6 \) with two different sets of \( A \) and \( B \)'s consequences shows that the number of swaps needed to make a decision can also depend on the balance of \( A \) and \( B \)'s consequences. \( A \) and \( B \)'s consequences are balanced when \( A > B \) for half of the (high priority) criteria and \( B > A \) for the other half of (high priority) criteria. The tests for \( m = 4 \) and \( m = 6 \) in Table 6.6 show that if two alternatives are not competitive on important criteria (i.e. unbalanced), the number of swaps needed to resolve the trade-offs increases. In the last test case with eight criteria, the algorithm slows down and the delays are sensible for the human user.

**Test case group c:** Table 6.7 summarizes the results of the tests in group c, where the difference of alternatives on criteria varies from Low to High. Making swaps and deciding over the alternatives is easier compared to the first and second groups, and in general, fewer number of swaps need to be made, and fewer swaps are rejected.

**Test case group d:** Table 6.8 summarizes the results of the tests in group d where the difference of alternatives on every criterion is High. Although in this set of tests, only one possible consequence is enumerated and few swaps are needed to make the final decision, suggesting the right swap is challenging. Due to significant difference of alternatives’ consequences on the criteria, the algorithm is forced to start with suggesting swaps that require significant sacrifice on a criterion that is Highly satisfied. If by chance, this is a high priority criterion, the algorithm faces several rejected swaps. Examples of this situation are observed in the test cases in Table 6.8, for \( m = 7 \) and \( m = 8 \).

**Conclusions:** Based on the results of the experiments we draw some conclusions and form some hypothesis:

- The number of required even swaps to make a decision over a pair of alternatives does not directly and proportionally grow by increasing the number of possible consequences of alternatives that the algorithm enumerates.
Table 6.7: Summary of test cases’ results for test case c, in Figure 6.8 (E indicates that the two alternatives are equally preferred)

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Possible Cases</th>
<th>Suggested alternative swaps</th>
<th>Rejected swaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>m = 2 5</td>
<td>$A_B = L \quad B_A = H$</td>
<td>$B$</td>
<td>1 0</td>
</tr>
<tr>
<td>m = 2 8</td>
<td>$A_B = ML \quad B_A = MH$</td>
<td>(E)</td>
<td>2 0</td>
</tr>
<tr>
<td>m = 3 24</td>
<td>$A_B = ML \quad B_A = M \quad A_B = MH$</td>
<td>$A$</td>
<td>3 0</td>
</tr>
<tr>
<td>m = 3 24</td>
<td>$A_B = ML \quad B_A = M \quad B_A = MH$</td>
<td>$A$</td>
<td>5 0</td>
</tr>
<tr>
<td>m = 4 24</td>
<td>$A_B = ML \quad B_A = M \quad B_A = MH \quad A_B = H$</td>
<td>$A$</td>
<td>5 0</td>
</tr>
<tr>
<td>m = 4 90</td>
<td>$A_B = L \quad B_A = ML \quad B_A = M \quad A_B = MH$</td>
<td>$A$</td>
<td>9 1</td>
</tr>
<tr>
<td>m = 5 120</td>
<td>$A_B = L \quad B_A = ML \quad B_A = M \quad B_A = MH \quad A_B = H$</td>
<td>$A$</td>
<td>6 0</td>
</tr>
<tr>
<td>m = 5 120</td>
<td>$A_B = L \quad A_B = ML \quad B_A = M \quad B_A = MH \quad B_A = H$</td>
<td>$B$</td>
<td>6 0</td>
</tr>
<tr>
<td>m = 6 120</td>
<td>$A_B = L \quad A_B = ML \quad B_A = M \quad B_A = MH \quad B_A = H \quad A_B = H$</td>
<td>$A$</td>
<td>5 1</td>
</tr>
<tr>
<td>m = 6 120</td>
<td>$A_B = L \quad A_B = ML \quad A_B = M \quad B_A = MH \quad A_B = H \quad B_A = H$</td>
<td>$A$</td>
<td>14 3</td>
</tr>
<tr>
<td>m = 6 120</td>
<td>$A_B = L \quad A_B = ML \quad A_B = M \quad B_A = MH \quad B_A = H \quad B_A = H$</td>
<td>$B$</td>
<td>25 0</td>
</tr>
<tr>
<td>m = 7 600</td>
<td>$A_B = L \quad B_A = L \quad A_B = ML \quad B_A = M \quad A_B = MH \quad B_A = H \quad A_B = H$</td>
<td>$A$</td>
<td>9 0</td>
</tr>
<tr>
<td>m = 7 600</td>
<td>$A_B = L \quad A_B = L \quad B_A = ML \quad A_B = M \quad B_A = MH \quad B_A = H \quad B_A = H$</td>
<td>$A$</td>
<td>4 1</td>
</tr>
<tr>
<td>m = 7 600</td>
<td>$A_B = L \quad A_B = L \quad A_B = ML \quad B_A = M \quad B_A = MH \quad B_A = H \quad B_A = H$</td>
<td>$B$</td>
<td>7 0</td>
</tr>
<tr>
<td>m = 8 1000</td>
<td>$A_B = L \quad A_B = L \quad A_B = ML \quad B_A = M \quad B_A = MH \quad B_A = H \quad B_A = H$</td>
<td>$B$</td>
<td>5 1</td>
</tr>
<tr>
<td>m = 8 1800</td>
<td>$A_B = L \quad B_A = L \quad A_B = ML \quad B_A = M \quad A_B = M \quad B_A = MH \quad A_B = H \quad B_A = H$</td>
<td>$A$</td>
<td>15 1</td>
</tr>
</tbody>
</table>

- In extreme conditions where the difference between the impacts of alternatives is Low over all criteria (or High over all criteria), the suggested swaps by the algorithm involve minimal (or significant) trade-offs, and thus, in these conditions, decision makers reject swaps more frequently.
Table 6.8: Summary of test cases’ results for test case d, in Figure 6.8

<table>
<thead>
<tr>
<th>Criteria</th>
<th>Possible Cases</th>
<th>G₁</th>
<th>G₂</th>
<th>G₃</th>
<th>G₄</th>
<th>G₅</th>
<th>G₆</th>
<th>G₇</th>
<th>G₈</th>
<th>Suggested alternative</th>
<th>Swaps</th>
<th>Rejected swaps</th>
</tr>
</thead>
<tbody>
<tr>
<td>m = 2</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 3</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 3</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = M</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 4</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 4</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 5</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 5</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 6</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 6</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 7</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 7</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 7</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 7</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 8</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 8</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>m = 8</td>
<td>1</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td>B₁ = H</td>
<td>A₁ = H</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- If consequences of alternatives on important criteria are not balanced, resolving the trade-offs require more swap queries. In a balanced condition, one alternative is a better choice for half of (highly important) criteria, and the other alternative is a better choice for the other half of (highly important) criteria. Our hypothesis to explain this observation is:

  - The algorithm does not have any knowledge about the preferences of decision makers
over the criteria; thus, the algorithm suggests swapping highly important criteria with goals that have minimum priority. As a result, the swaps may not effectively eliminate irrelevant criteria, and the number of swaps grows.

6.4 Discussion

This section discusses the main benefits of the proposed method, considering the complexity and effort required to apply the method. The emphasis on participation of stakeholders in the decision analysis process can turn into the main practicality barrier. We also discuss other threats to validity and the usefulness of the proposed method.

6.4.1 Benefits of the Comparison-based Method

Compared to the Utility Theory, where an alternative is evaluated by the satisfaction value it provides to the decision stakeholders, our comparison-based method’s main benefit is enabling an objective decision analysis, in the absence of any measurement or evaluation of alternatives. Extracting utility of alternatives and numerical importance weights and calculating a utility value is computationally less complex; however, if even one piece of information in the utility formula is missing, it is not possible to calculate the utility value.

To collect utility values per criteria, stakeholders need to estimate consequences of every alternative, which in total requires $m \times n$ knowledge-intensive and cognitively hard queries from stakeholders for $m$ criteria and $n$ different alternatives. Extracting stakeholders’ preferences over decision criteria requires $m$ subjective numerical importance weights or $m^2$ AHP preferences comparison queries. Given $m$ goals and $n$ alternatives, by applying our method, decision stakeholders need to make $m \times (n - 1)$ comparisons. In our method, stakeholders’ preferences are accommodated into the decision process by even swaps that stakeholders make; therefore, analysts can avoid extracting subjective numerical importance weights.

The focus of this thesis is on analyzing the impact of security risks on the final decision; however, the decision aid method can be applied for analyzing any decision problem that involves
a set of criteria and a family of alternative actions, independent of the target domain or the source of risks (e.g., IT risks, business risks, etc.).

### 6.4.2 Threats to Validity and Usefulness of the Comparison-based Method

**Exponential Growth:** A major threat to practicality of our method is the exponential grow rate of possible consequence enumerations (in the order of $5^m$, if the scale of comparison has five different levels and the pair of alternatives are compared over $m$ criteria). To analyze a large number of criteria and alternatives, the algorithm may ask numerous swaps from stakeholders, and return several exceptional patterns of possible consequences to human experts for a final judgement. As discussed in the experiment section earlier. In reality, however, the number of swaps does not necessarily grow exponentially. Stakeholders are only asked about a few exceptional patterns of consequences (not all $5^m$ of them). The number of swaps that stakeholders need to make can be significantly reduced by reusing previous swaps (rule 2 in Section 6.2.4).

**The Scale of Comparisons:** A major threat to the validity of this work is the choice of interval scale of Low to High for making comparisons. We assumed that decision makers would interpret the scale uniformly across different criteria and they would make consistent swaps among goals according to this scale of evaluation. In practice, however, stakeholders are not always consistent in their judgements. In the future work section, we will discuss that these assumptions need to be tested in practice, and we intend to enhance our tool to trace the inconsistencies in even swaps.

**Local Values and Local Decision:** In this method, the difference between consequences of alternatives are used to enumerate possible consequences that a pair of alternatives can have. The comparison of each pair of alternatives (in the scale of 0 to High) is independent from other alternative pairs and the enumerated consequences of one alternative on one goal is not relevant to the consequences of the alternative on other goals. Enumerated consequences are local values for analyzing two competing alternatives and eliminating one from the decision process. Thus,
to make these comparisons the stakeholder can only focus on one pair of alternatives and avoid considering the rest.

In contrast, when stakeholders are asked to estimate the consequences of each solution on the decision criteria, the extracted numerical utilities are global values which are comparable and relevant to other values. Thus, stakeholders need to consider all alternatives and criteria when providing utility values of alternatives.

### 6.4.3 Usefulness and Feasibility of Even Swaps

**The Notion of Swap Reusability:** Marginal Rate Substitution (MRS) indicates the maximal amount of a goal satisfaction level that stakeholders are willing to sacrifice for a unit of increase in another goal. Even swaps implicitly capture the MRSs of the goals that stakeholders swap. Generally, MRSs of two goals at two different satisfaction levels can be different [149]. The notion of swap reusability, which we discussed in the previous section, is based on the assumption that MRSs are not static, and the changes made to goals in an even swap depend on the satisfaction levels of the goals or the context in which the swap is asked from stakeholders; this restricts the reusability of swaps, because the algorithm cannot rely on a static weight ratios between the decision criteria.

In what follows, we explain the underlying reasons for assuming that MRSs are not static. In the swap \((g_x : i_x \rightarrow i'_x \Leftrightarrow g_y : i_y \rightarrow i'_y)\), changing \(g_x\) by \(d_x\) levels \((d_x = |i'_x - i_x|)\) is compensated with changing \(g_y\) by \(d_y\) levels \((d_y = |i'_y - i_y|)\). Stakeholders have a mental utility function like \(F\), and based on \(F\), they trade \(|F(i'_x) - F(i_x)|\) with \(|F(i'_y) - F(i_y)|\). However, the utility function, \(F\), is not linear; hence, in a situation where the satisfaction level of \(g_x\) is not equal to \(i_x\), changing \(g_x\) by \(d_x\) levels cannot be compensated with changing \(g_y\) by \(d_y\) levels. \(F\) is not linear because people have different “mental” spending accounts for different goals [137], i.e., if a stakeholder reaches the maximum amount that he/she is willing to spend on satisfying a goal \(g_x\), he/she will not sacrifice on other goals for better satisfaction of \(g_x\), which is already good enough. Even though stakeholders may have previously increased the satisfaction level of \(g_x\) and compensated it by decreasing other goals, as soon as they psychologically feel that too much or enough is spent on \(g_x\), the MRS they are willing to spend on \(g_x\) would change in the
next Even Swaps queries.

For example, if stakeholders previously made the swap \((T_1 : 2ms \rightarrow 3ms \Leftrightarrow C_1 : $2m \rightarrow $1.8m)\), it does not indicate that whenever \(T_1\) is increased 1 ms, then, the cost of \(C_1\) must be reduced $0.2 m. For instance, if humans cannot sense the difference between the 2 ms and 1 ms delays, reducing \(T_1\) to less than 2 ms does not provide any extra value on performance. Therefore, the swap \((T_1 : 2ms \rightarrow 1ms \Leftrightarrow C_1 : $2.2m \rightarrow $2m)\) is not valid and reusable, which limits the reusability of swaps and increases the number of swap queries needed.

It has been shown in other multi-criteria decision analysis methods that preferences of stakeholders are largely context-dependent [141]. Based on such evidence and the discussion of static MRSs above, a major benefit of our automation of Even Swaps method is eliciting value trade-offs in the context-dependant process.

**Context-Dependant Preference Elicitation:** One of the main advantages of our automated Even Swaps is extracting stakeholders’ value trade-offs dynamically and context-dependently. We discussed under the notion of swap reusability that preferences of stakeholders are largely context-dependant. This contradicts the established belief in the theory of rational choice that preference between options does not depend on the presence or absence of other options [126]. However, stakeholders preference models are not likely to be complete [123] and stakeholders are unlikely to make the effort to express preferences for decision criteria until a pertinent context appears [126]. In this regard, the strength of Even Swaps is circumventing a preference model that is extracted independently from alternatives and their consequences. Instead Even Swaps only extracts value trade-offs in the context of a solution, when the swap can actually influence the final decision.

**The Notion of Discrete Alternatives:** We have assumed that the decision analysis algorithm is given a limited number of “independent” alternatives which have conflicting consequences on some decision criteria. We assume alternatives are not a variable that needs to be decided, and the definition and consequences of alternatives are independent of each other.

In practice, however, varieties of real-world problems exist in which one or more variables need to be optimized by considering the impact of these variables on a set of criteria. In other word, selecting a solution among alternatives is translated to determining the right value for
one or more variables among a continuous set of possible values (or possible alternatives). For example, assume stakeholders need to decide on the length of the password phrase or number of password check mechanisms for a successful authentication process. The longer a password phrase is (or the more password checks a system requires), the higher the security would be, but at the same time, the lower the usability of the authentication mechanism is. In this scenario, the aim of a decision analysis method is to identify an optimum password length that balances security and usability. The alternative solutions are not any more discrete, and in theory, can have infinite possible values. Such decision scenarios can be categorized under optimization problems, and example methods to tackle such problems are dynamic programming and linear programming. Our method focuses on problems where stakeholders need to select discrete and fixed alternatives, that are not specified through unset variables.

**Extracting (Consistent) Value Trade-offs from Stakeholders:** Making trade-offs by even swaps may require substantial cognitive abilities and domain knowledge. If stakeholders are not able to swap suggested goals, the algorithm suggests the next best swap, and stores the rejected swap in a “black list”, and in the rest of the process, would not ask that swap from the stakeholder. In practice, stakeholders may reject every suggested swap, which results in several tedious swap queries to find the one that the stakeholder is able to answer. The process may become time-consuming or never end.

To prevent the dead-ends, after a number of rejected suggestions, the tool asks the users to select the goals for the next swap themselves. Nevertheless, if stakeholders are not able to specify the maximal amount of a goal satisfaction level that they are willing to sacrifice for a unit of increase in another goal, then it is probable that they will not be able to numerically specify preferences over the goals either.

We have also assumed that stakeholders would make consistent swaps among goals. In practice, however, stakeholders are not always consistent in their judgements. Another threat to practicality of the algorithm is the diversity of evaluation scales in the consequence table. Stakeholders may not be able to swap a goal measured in absolute values with a goal that is evaluated by qualitative labels.
6.4.4 Decision Analysis Feedback on Models and Assessments

There are some obstacles in applying the decision aid methods that can be used as feedback to the security requirements models and assessment results. These obstacle scenarios are:

1. Decision stakeholders do not agree with the suggested solution by the algorithm.
2. Decision stakeholders are not able to make the suggested swap.
3. Decision stakeholders are not able to evaluate the risk of a vulnerability or impact of a solution on decision criteria.

First scenario: disagree with the suggested solution: It is possible that decision stakeholders do not agree with the solution suggested by the decision aid method as the overall best available option. For example, if the decision analysis method ultimately suggests a solution that is costlier than the stakeholders’ tolerance, the decision problem formulation can have limitations or incompleteness in three possible areas:

1) It is possible that the security requirements model does not reflect all or correct decision criteria that are of importance for stakeholders, and thus, some influential goals have been neglected in the consequence table. This requires either further decomposition of goals and softgoals to more concrete and tangible decision criteria, identifying missing (soft)goals in the security requirements model, or identifying entirely new actors (stakeholders) whose goals have not been considered.

2) The consequences of alternatives collected in the consequence table and in the CVSS re-assessment of vulnerabilities’ may not correctly reflect the impacts of different solutions on decision criteria. This situation requires checking the consistency of security and risk evaluations. If analysts did not encounter inconsistencies in re-evaluation of consequences, one other possibility is incompleteness or inaccuracy of the security requirements model. The model might not be detailed enough to reflect the consequences of solutions and vulnerability exploitations.

3) The other possible explanation for this situation is that stakeholders’ input into the Even Swaps process would not reflect their actual value trade-offs and preferences over decision criteria. This can stem from two possible scenarios:
(a) The cognitive demands to make even swaps consistently.

(b) Vagueness and abstract definition of decision criteria.

To address the issue (a), a new swap needs to be suggested that demands less reasoning to answer. In case (b), goals and requirements are not refined enough into tangible criteria; thus, making trade-offs over such abstract and intangible parameters can result in arbitrary and inconsistent even swaps.

**Second scenario: unable to make swaps:** We earlier discussed that our decision aid methods primarily rely on the ability of decision stakeholders to trade one goal for another in even swaps. Three different issues can cause stakeholders to reject the swap suggested by the algorithm:

1) The cognitive demands to make even swaps consistently, which are explained in the previous scenario, case 3(a).

2) Vagueness and abstract definition of decision criteria, which is explained in the previous scenario, case 3(b).

3) Incorrect estimation of desired value trade-offs by the algorithm. Currently, the algorithm does not extract relative rankings of decision criteria based on the swaps that stakeholders have made previously. Thus, it is possible that the algorithm suggests significant sacrifice on a goal that is highly important for the stakeholder in trade for improving a goal that is not of importance for stakeholders. The immediate solution to this problem is reversing the suggested swap, i.e., if the original swap is \((gx : x \rightarrow x' \leftrightarrow gy : y \rightarrow ?)\), the algorithm can suggest the reverse swap as: \((gy : y \rightarrow y' \leftrightarrow gx : x \rightarrow ?)\). If stakeholders cannot make the reversed swap either, then inability to make the swaps can be caused by issue 1 or 2.

**Third scenario: unable to evaluate solutions or risks:** The final obstacle in applying the proposed decision aid method is the lack of risk and solution evaluations. Risk of vulnerabilities and consequences of solutions are evaluated on the basis of goal-oriented models as well as CVSS metrics. The inability of stakeholders to evaluate these parameters can be caused by various issues:

1) Lack of security knowledge in the domain of analysis.
2) Incompleteness or inaccuracy of the security requirements model, either:

(a) Insufficient goal refinement, so high level goals are intangible and cannot be evaluated.

(b) Missing consequence links that represent impacts of vulnerabilities and security solutions.

**Summary:** Applicability and correctness of the decision analysis results significantly depend on the completeness and accuracy of the security requirements model, sufficient refinement of decision criteria, knowledge, and cognitive capabilities of decision analysts and stakeholders. The result of the decision analysis can indicate the need for further changes or refinements on the requirements model.
Chapter 7

Evaluation and Contributions

“The criterion of the scientific status of a theory is its falsifiability, or refutability, or testability.” Karl Popper

“No amount of experimentation can ever prove me right; a single experiment can prove me wrong.” attributed to Albert Einstein

Previous chapters presented three main components for security requirements and risk analysis: a modeling notation for expressing security risks and requirements, a risk assessment method for analyzing vulnerabilities, and a decision analysis algorithm for selecting good-enough security countermeasures.

Evaluating utility and usability of the proposed framework through empirical (controlled) experiments in a lab setting is challenging. Given the task of model understanding, the outcome is cognitive and therefore not directly observable. Effectiveness of the modeling notation and analysis methods can be evaluated by measuring understanding and performance of the analysts. This can be done through problem solving tasks that require participants to reason about the problem being represented in the model [50]. The performance of analysts in the problem solving tasks is an indicator of the impacts of adopting the modeling and analysis technique.

However, measuring performance and understanding of the subjects is troublesome, because various biases threaten the validity of the results as discussed in Section 4.6.1 in Chapter 4.
For example, familiarity of experiment’s subjects with requirements modeling in general, i* Framework particularly, their level of security knowledge, and previous background about the domain being used in the scenarios can dramatically change the results we would observe in a controlled experiment.

We worked out case study scenarios and studies, drawn from practical problems or through collaboration with industrial partners, to provide objective evidence about the utility of the proposed framework. This chapter focuses on applying the entire framework (all three components) to analyzing top web application vulnerabilities and selecting proper security solutions for each risk. We report details of this study, including models, risk assessment and the decision analysis results. We compare the result of the risk assessment using our method with common assumptions about those vulnerabilities in the security engineering community.

Based on the results of case studies presented in current and previous chapters and analysis of expressiveness and reasoning power of our methods, this chapter compares the contributions of this framework with similar existing methods for modeling and analyzing security requirements, risk assessment, and trade-off decision analysis.

7.1 Case Study: Analysis of Web Application Vulnerabilities

The Open Web Application Security Project (OWASP) [121] collects top vulnerabilities of web applications, assesses criticality of vulnerabilities, and analyzes possible attack vectors. However, the impacts of exploitations in OWASP repositories are not usually expressed objectively. In this section, we apply the proposed framework in this thesis for incorporating web application vulnerabilities from OWASP knowledge bases into a simple web application requirements model, analyze the risk of those vulnerabilities, and suggest proper security countermeasures.

In this case study, we identify and model main elements of a web application independent of a business domain. The generic web application model is expressed in the form of an i* agent- and goal-oriented model. Using the modeling approach introduced in Chapter 4, we
model common and critical web application vulnerabilities. This approach helps trace where and when a vulnerability is introduced to the application, how vulnerabilities are propagated to other elements and agents, what tasks can exploit the vulnerability, and what the consequences of exploitations are on higher goals.

We apply the risk assessment methods in Chapter 5 to evaluate the risk of exploitation scenarios, and based on the assessment results, we investigate proper security solutions. Finally, using the decision analysis method of Chapter 6, we suggest the best security solution against web application vulnerabilities.

Figure 7.1 presents a generic web application model, that captures main actors and interactions in a general web application. This $i^*$ model helps express the main interaction parties as well as their goals, tasks, and requirements. In what follows, we analyze two top web application vulnerabilities in the OWASP list, add them to the model in Figure 7.1. We analyze the propagation vulnerabilities and their exploitation scenarios, and finally assess their risks and potential security solutions.

![Figure 7.1: The generic agent- and goal-oriented model of web applications](image-url)
7.1.1 Injection attack: OWASP Number One Vulnerability

In the OWASP list of top ten vulnerabilities, injection attacks are the most serious vulnerability in 2010\(^1\). Injection flaws occur when an application sends un-trusted data to an interpreter. Injection flaws are very prevalent, particularly in legacy code, often found for example in SQL queries. According to OWASP, to exploit this vulnerability, the attacker sends simple text-based input that exploits the syntax of the targeted interpreter. Almost any source of data can be an injection vector.

Modeling Vulnerabilities, Attacks, and Security Requirements

In the simple web application scenario in this case study, the attacker can be any of the legitimate users of the web application. Figure 7.2 shows how an SQL injection vulnerability is introduced to the system, through User input (highlighted by a circle around the vulnerability). The arrows on the model depict how the vulnerability is propagated to the Browser. Figure 7.3 shows how the vulnerability is propagated from the Browser to the Web server.

Figure 7.2: Introduction of SQL injection vulnerability to the web applications

Figure 7.4 depicts that SQL injection at the server side can help an adversary Retrieve sensitive information from database. (The model does not depict the impacts of this exploitation on the goals and functionalities of the Web server or Browser)

\(^1\)https://www.owasp.org/index.php/Top_10_2010-A1
Risk Assessment

We evaluate the risk of exploiting SQL injection vulnerability, by referring to the CVSS metrics Table 5.1 in Chapter 5. Table 7.1 shows the evaluation of probability and damage of SQL injection based on adapted CVSS risk metrics. (This assessment excludes the impacts of vulnerability exploitation on the goals of Web server or Browser.)

The result of risk assessment in Table 7.1 shows that the exploitation of SQL injection is highly probable and can cause a high level of damage. To avoid this attack, we investigate
two main security countermeasures: Parametrized Queries and Stored Procedures. By applying Parametrized Queries, developers must be careful to set the parameters of SQL queries from the user input correctly, otherwise, this solution is not totally useful against SQL injection. Stored Procedures of databases is a stronger solution by preventing execution of any query that is not stored in the database. Figure 7.2 shows the re-evaluation of risk factors in the presence of these two solutions and also in the situation that both are combined.

Table 7.1: Evaluating the risk of SQL injection by CVSS

<table>
<thead>
<tr>
<th>Metrics</th>
<th>SQL injection</th>
<th>Aggregated Score (a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Vector (AV)</td>
<td>General network (Low)</td>
<td></td>
</tr>
<tr>
<td>Access Complexity (AC)</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Authentication (Au)</td>
<td>None (Low)</td>
<td></td>
</tr>
<tr>
<td>Exploitability (E)</td>
<td>Functional method (Low)</td>
<td></td>
</tr>
<tr>
<td>Remediation Level (RL)</td>
<td>Official Fix (High)</td>
<td></td>
</tr>
<tr>
<td>Report Confidence (RC)</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Collateral Damage Potential (CDP)</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Target Distribution (TD)</td>
<td>High percentage (High)</td>
<td></td>
</tr>
<tr>
<td>Impact of exploitation on goals</td>
<td>Not Analyzed</td>
<td></td>
</tr>
</tbody>
</table>

\[
\begin{align*}
    l &= \# Low = 4 \\
    m &= \# Medium = 0 \\
    h &= \# High = 2 \\
    a &= \text{Low Probability} = \text{High} \\
    h &> l + m \text{ thus}
\end{align*}
\]

Parametrized Queries reduce the probability of exploiting SQL injections; however, if an attacker is able to turn around the Parametrized Queries, the damage of the attack is still severe. Stored Procedures are a better solution, since the probability of mounting a successful attack would be lower, and the damage caused by an SQL injection is controlled by having only pre-defined procedures executed. So the question is, why SQL injection attacks are still the most common and dangerous types of attacks according to OWASP? The answer may stem from trade-offs and conflicts of goals in applying these two solutions. In the next section, we discuss these trade-offs and apply the decision analysis approaches introduced in Chapter 6 to select a proper solution against SQL injection.
Table 7.2: Re-evaluating the risk of SQL injection attack under the presence of security solutions

<table>
<thead>
<tr>
<th>Metrics</th>
<th>SQL Injection</th>
<th>With Parameterized Queries</th>
<th>With Stored Procedures</th>
<th>With both solutions combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Vector (AV)</td>
<td>General network (Low)</td>
<td>General network (Low)</td>
<td>General network (Low)</td>
<td>General network (Low)</td>
</tr>
<tr>
<td>Access Complexity (AC)</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Authentication (Au)</td>
<td>None (Low)</td>
<td>None (Low)</td>
<td>None (Low)</td>
<td>None (Low)</td>
</tr>
<tr>
<td>Exploitability (E)</td>
<td>Functional method (Low)</td>
<td>Proof-of-Concept (Medium)</td>
<td>Unproven method (High)</td>
<td>Unproven method (High)</td>
</tr>
<tr>
<td>Remediation Level (RL)</td>
<td>Official Fix (High)</td>
<td>Official Fix (High)</td>
<td>Official Fix (High)</td>
<td>Official Fix (High)</td>
</tr>
<tr>
<td>Report Confidence (RC)</td>
<td>High</td>
<td>Medium</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Collateral Damage Potential (CDP)</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Target Distribution (TD)</td>
<td>High percentage (High)</td>
<td>High percentage (High)</td>
<td>Medium percentage (Medium)</td>
<td>Low percentage (Low)</td>
</tr>
<tr>
<td>Impact of exploitation on goals</td>
<td>Not Analyzed</td>
<td>Not Analyzed</td>
<td>Not Analyzed</td>
<td>Not Analyzed</td>
</tr>
</tbody>
</table>

Decision Analysis

*Stored Procedures* encapsulate the logic of accessing and retrieving data and expose an interface that is accessed via a component, so changes to a database server are only possible through pre-defined procedures. *Stored Procedures* are easy for code maintenance; for example, when new queries are added, if the data structure of results is not changed, developers can avoid re-compiling the application. However, *Stored Procedures* are usually hard to set up and more difficult than *Dynamic Queries* and *Parametrized Queries* to write. It is usually difficult for developers to write complex queries in *Stored Procedures*, and they also make debugging more complex.
Parametrized Queries require less setup than Stored Procedures. Parametrized Queries can be cached too, which helps the performance of executing queries, but Stored Procedures’ big advantage in performance is keeping the data on the database server.

Figure 7.5 depicts the impact of Parametrized Queries, Stored Procedures, and the usual query creation (Dynamic Queries that are the result of concatting strings dynamically) on goals of web application Developers. By using Parametrized Queries and Stored Procedures together, developers can benefit from advantages of each method at the right place and avoid some of the disadvantages. This model (Figure 7.5) helps rationalize about these four alternatives: Parametrized Queries, Stored Procedures, using both together, and Dynamic Queries.

Figure 7.5: Impacts of alternative solutions against SQL injection

We do not have numerical data that indicates the satisfaction level of Developer’s goals in Figure 7.5, thus we compare consequences of alternatives on $G_1$, $G_2$, $G_3$, $G_4$, $G_5$. Table 7.3 summarizes consequences of these alternatives on the probability and damage of the SQL injection risk. We compared Dynamic Queries and Parameterized Queries (the first two rows). For example, Dynamic Queries ($A_1$) is a better choice for Changing queries easily ($G_1$), but both solutions are indifferent with respect to compiling issues ($\frac{\Delta_1}{\Delta_2}$ on $G_1$ is Medium Low (ML) and $\frac{\Delta_2}{\Delta_2}$ on $G_4$ is Zero). Table 7.3 also compares Stored Procedures with combining Stored Procedures and Parameterized Queries.
Table 7.3: Consequences of alternative solutions against SQL injection

<table>
<thead>
<tr>
<th>Alternative</th>
<th>Minimize</th>
<th>Maximize</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prob SQL Injection</td>
<td>Damg SQL Injection</td>
</tr>
<tr>
<td>A1: Dynamic Queries</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>A2: Parameterized Queries</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>A3: Stored Procedures</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>A4: Combination of A2 &amp; A3</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Comparing Dynamic Queries and Parameterized Queries:** Since Dynamic Queries and Parameterized Queries are indifferent with respect to $G_4$ and the damage of SQL injection, the decision analysis is done with respect to five criteria. For example, $A_1$ is a better choice with respect to $G_1$ with the difference of Medium Low, which indicates four potential consequences are possible:

<table>
<thead>
<tr>
<th></th>
<th>$G_3$</th>
<th>$G_3$</th>
<th>$G_3$</th>
<th>$G_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_1$</td>
<td>Medium Low</td>
<td>Medium</td>
<td>Medium High</td>
<td>High</td>
</tr>
<tr>
<td>$A_2$</td>
<td>Low</td>
<td>Medium Low</td>
<td>Medium</td>
<td>Medium High</td>
</tr>
</tbody>
</table>

Respectively, the comparison-based method generates $4 \times 5 \times 5 \times 2 = 200$ possible consequences for $A_1$ and $A_2$. Deciding over these 200 possible consequences by using Even Swaps strongly depends on preferences of stakeholders. For example, we assumed lowering the probability of SQL injection is the most important goal, and next, performance is more important than maintenance and ease of development. With six swaps in total, the algorithm finds $A_2$ a better choice for all 200 possible consequences. Thus, we can remove $A_1$ and continue the analysis with $A_2$, $A_3$ and $A_4$.

The algorithm then generates 540 possible consequences for the comparisons of $A_3$ and $A_4$, and by making six swaps, $A_4$ is recognized as the better choice for all 540 cases. The remaining solutions are $A_2$ and $A_4$. Table 7.4 depicts the comparison of these two alternatives. The algorithm creates 750 possible consequences and by asking seven swaps, $A_4$ is recognized as the best solution.
Table 7.4: Comparison of Parameterized Queries with combined solutions

<table>
<thead>
<tr>
<th>Alternative</th>
<th>Minimize</th>
<th>Maximize</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prob SQL Injection</td>
<td>Damg SQL Injection</td>
</tr>
<tr>
<td>A2: Parameterized Queries</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>A4: Combination of A2 &amp; A3</td>
<td>Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

**Results:** Combining Stored Procedures and Parameterized Queries is recognized as the best available choice to protect against SQL injection. However, this is not always the best solution, because in making this decision, we favored security and performance to other criteria.

### 7.1.2 Cross-Site Scripting: OWASP Second Top Vulnerability

Cross-Site Scripting (XSS) flaws occur when an application includes user supplied data in a page sent to the browser without properly validating or escaping that content. Thus, attackers can inject and execute scripts in the victim’s browser to hijack user sessions, deface web sites, insert hostile content, redirect users, hijack the user’s browser using malware, etc.

#### Modeling Vulnerabilities, Attacks, and Security Requirements

Figure 7.6 shows how the Malicious script vulnerability is introduced to the web application, through the User input. Malicious script is propagated from the Browser to the Web server. Figure 7.7 shows how Malicious script moves from the Web server to the Victim Browser. In this scenario, the Victim Browser Runs required script including the Malicious script, and for example, this can help the attacker Steal victim cookies.

#### Risk Assessment

We evaluate the risk of exploiting XSS vulnerability, by referring to the CVSS metrics in Table 5.1 in Chapter 5. Table 7.5 shows the evaluation of probability and damage of Cross-Site Scripting based on adapted CVSS risk metrics. This assessment excludes the impacts of vulnerability exploitation on the goals of Web server and Victim Browser.
Figure 7.6: Introduction of Cross-Site Scripting vulnerability to the simple web application scenario

The result of risk assessment in Table 7.5 shows that the exploitation of Cross-Site Scripting is highly probable and the damage would be high. To avoid this attack, we investigate a number of security mechanisms: HTML entity encoding, Black list input validation, and White list input validation.

HTML entity encoding ensures the data sent to the browser is not going to be interpreted by the browser as mark-ups and should be treated as user data. HTML encoding usually means encoding < as &lt;, > as &gt;, & as &amp;, etc. However, HTML encoding is not usually enough protection against Cross-Site Scripting attack, when putting un-trusted data inside a <script> tag or in an URL.

Black list input validation filters out the malicious characters from the user input. On the contrary, White list input validation checks whether the output being sent back to the browser only consists of acceptable characters. Thus White list input validation is
Figure 7.7: Cross-Site Scripting vulnerability exploitation at the Victim Browser

Table 7.5: Evaluating the risk of Cross-Site Scripting by CVSS

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Cross-Site Scripting</th>
<th>Aggregated Score (a)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Vector (AV)</td>
<td>General network (Low)</td>
<td></td>
</tr>
<tr>
<td>Access Complexity (AC)</td>
<td>Low</td>
<td></td>
</tr>
<tr>
<td>Authentication (Au)</td>
<td>None (Low)</td>
<td></td>
</tr>
<tr>
<td>Exploitability (E)</td>
<td>Functional method (Low)</td>
<td></td>
</tr>
<tr>
<td>Remediation Level (RL)</td>
<td>Official Fix (High)</td>
<td></td>
</tr>
<tr>
<td>Report Confidence (RC)</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Collateral Damage Potential (CDP)</td>
<td>High</td>
<td></td>
</tr>
<tr>
<td>Target Distribution (TD)</td>
<td>High percentage (High)</td>
<td></td>
</tr>
<tr>
<td>Impact of exploitation on goals</td>
<td>Not Analyzed</td>
<td></td>
</tr>
</tbody>
</table>

\[ l = \# \text{ Low} = 4 \\
 m = \# \text{ Medium} = 0 \\
 h = \# \text{ High} = 2 \\
 h > l + m \text{ thus } \\
 a = \text{Low Probability} = \text{High} \\
 \text{Max(Low, High, ?) = High} \\
 \text{Damage} = \text{High} \]

more powerful than Black list filtering. Table 7.6 shows the re-evaluation of risk factors in the presence of these solutions.
Table 7.6: Re-evaluating the risk of Cross-Site Scripting attack under the presence of security solutions

<table>
<thead>
<tr>
<th>Metrics</th>
<th>Cross-Site Scripting</th>
<th>with HTML Encoding</th>
<th>with Black List Validation</th>
<th>with White List Validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access Vector (AV)</td>
<td>General network (Low)</td>
<td>General network (Low)</td>
<td>General network (Low)</td>
<td>General network (Low)</td>
</tr>
<tr>
<td>Access Complexity (AC)</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>Authentication (Au)</td>
<td>None (Low)</td>
<td>None (Low)</td>
<td>None (Low)</td>
<td>None (Low)</td>
</tr>
<tr>
<td>Exploitability (E)</td>
<td>Functional method (Low)</td>
<td>Proof-of-Concept (Medium)</td>
<td>Proof-of-Concept (Medium)</td>
<td>Unproven method (High)</td>
</tr>
<tr>
<td>Remediation Level (RL)</td>
<td>Official Fix (High)</td>
<td>Official Fix (High)</td>
<td>Official Fix (High)</td>
<td>Official Fix (High)</td>
</tr>
<tr>
<td>Report Confidence (RC)</td>
<td>High</td>
<td>High</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>Collateral Damage Potential (CDP)</td>
<td>High</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>Target Distribution (TD)</td>
<td>High percentage (High)</td>
<td>High percentage (Medium)</td>
<td>Medium percentage (Medium)</td>
<td>Low percentage (Low)</td>
</tr>
<tr>
<td>Impact of exploitation on goals</td>
<td>Not Analyzed</td>
<td>Not Analyzed</td>
<td>Not Analyzed</td>
<td>Not Analyzed</td>
</tr>
</tbody>
</table>

**HTML entity encoding** slightly reduces the probability of Cross-Site Scripting attack; however, if it fails, the damage would be as high as having no countermeasure in place. The security benefits of this solution are considerable, and it is easy to implement because most programming languages provide libraries for HTML encoding. **Black list input validation** could be trickier to implement, but it prevents the damage of malicious characters by filtering them out. **White list input validation** is usually challenging to properly implement, but it filters many more potential malicious characters and reduces the damage of malicious input more than Black list input validation. We apply the decision analysis approaches introduced in Chapter 6 to select a solution among these options.
Decision Analysis

We aggregated benefits and costs of the alternative security solutions against Cross-Site Scripting in Table 7.7. To select a solution among these four options, we apply the automated Even Swaps method (See Section 6.2). Note that the damage and probability of XSS with using $A_2$ is High according to the CVSS metrics. Although the CVSS metrics show the damage is still High, because the Target Percentage is reduced from High to Medium (compare $A_1$ and $A_2$), we evaluated the damage in the scale of Low to High, as Medium High (based on our judgement). With the same reasoning, since the Exploitability with the presence of $A_2$, $A_3$, and $A_4$ is more challenging, the probability of Cross-Site Scripting with $A_4$ is Medium Low and with $A_2$ or $A_3$ is Medium High (Look at Table 7.7).

Table 7.7: Consequences of alternative solutions against Cross-Site Scripting

<table>
<thead>
<tr>
<th>Alternative</th>
<th>Prob XSS</th>
<th>Damg XSS</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1: No filtering</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>A2: HTML entity encoding</td>
<td>High (CVSS)</td>
<td>High (CVSS)</td>
<td>Medium Low</td>
</tr>
<tr>
<td>A3: Black list validation</td>
<td>High (CVSS)</td>
<td>Medium High (judgment)</td>
<td>Medium</td>
</tr>
<tr>
<td>A4: White list validation</td>
<td>Medium (CVSS)</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

Selecting a Pair of Solutions for Even Swaps: In the first step, the automated Even Swaps method picks $A_1$ and $A_4$, because these two are the most distinguished pairs of alternatives, among all six possible pairs (The distance factor of $A_1$ and $A_4$ is calculated as 2.2, which is the maximum among other pairs).

The Chain of Swaps: First, the algorithm suggests the following swap:

(Prob XSS: High $\rightarrow$ MedLow $\iff$ Effort: Low $\rightarrow$ X)

Let us assume changing the probability of Cross-Site Scripting from High to Medium Low is worth increasing the effort from Low to Medium High. Thus the consequences of $A_1$ and $A_4$ are revised as follows:
Then, the algorithm suggests the following swap:

\[\text{Damage XSS: High} \rightarrow \text{Low} \Leftrightarrow \text{Effort: Medium High} \rightarrow X\]

This swap might not be applicable in many cases, because it suggests a significant change in the damage of the XSS, while the effort cannot be increased more than one unit from Medium High; thus we reject this swap and suggest making the reverse of this as:

\[\text{Effort: Med High} \rightarrow \text{High} \Leftrightarrow \text{Damage XSS: High} \rightarrow X\]

Let us assume that slightly more effort can be traded with a small reduction of damage, for example X= Medium High. Thus the consequences of \(A_1\) and \(A_4\) are revised as follows:

<table>
<thead>
<tr>
<th>Prob XSS</th>
<th>Damage XSS</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A_1)</td>
<td>Medium Low</td>
<td>High</td>
</tr>
<tr>
<td>(A_4)</td>
<td>Medium Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

This indicates that without applying any security mechanism, the required effort is low, but reducing the damage and probability of XSS is worth the effort for implementing \(A_4\), and thus \(A_1\) can be removed from the list of possible actions. Then \(A_4\) and \(A_2\) are compared for the next chain of swaps. The algorithm first suggests the following swap:

\[\text{Prob XSS: Medium High} \rightarrow \text{Medium Low} \Leftrightarrow \text{Effort: Med Low} \rightarrow X\]

Let us assume changing the probability of Cross-Site Scripting from Medium High to Medium Low is worth increasing the effort from Medium Low to High. Thus the consequences of \(A_2\) and \(A_4\) are revised as follows:

<table>
<thead>
<tr>
<th>Prob XSS</th>
<th>Damage XSS</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>(A_2)</td>
<td>Medium Low</td>
<td>Medium High</td>
</tr>
<tr>
<td>(A_4)</td>
<td>Medium Low</td>
<td>Low</td>
</tr>
</tbody>
</table>

The results indicate that although HTML encoding \((A_2)\) requires a Medium Low effort, the reduction of damage and probability of XSS is worth the efforts of implementing \(A_4\), and thus
A_2 is also removed from the list of possible actions. Then A_4 and A_3 are compared for the next chain of swaps. The algorithm first suggests the following swap:

\[(\text{Prob XSS: MediumHigh} \rightarrow \text{MediumLow} \Leftrightarrow \text{Effort: Medium} \rightarrow X)\]

Again making this swap is not possible, because the reduction of the probability of XSS from Medium High to Medium Low is considered a significant improvement. For such a considerable risk reduction, increasing the effort from Medium to the maximum level (High) is not enough. Thus we make the reverse of this swap as follows:

\[(\text{Effort: Medium} \rightarrow \text{High} \Leftrightarrow \text{Prob XSS: MediumHigh} \rightarrow X)\]

Let us assume that by changing the implementation effort of A_3 from Medium to High, it is expected that the probability of Cross-Site Scripting is reduced from Medium High to Medium. Thus, the consequences of A_3 and A_4 are revised as follows:

<table>
<thead>
<tr>
<th></th>
<th>Prob XSS</th>
<th>Damage XSS</th>
<th>Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>A_2</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>A_4</td>
<td>Medium Low</td>
<td>Low</td>
<td>High</td>
</tr>
</tbody>
</table>

**Results:** The series of swaps discussed above indicate that A_4 dominates A_2, A_1, and A_3, and thus, A_4 is the overall best solution. Note that this conclusion may not be always valid: that White List Input Validation (A_4) is the best solution to protect against Cross-Site Scripting, because we favored reduction of damage and probability of the attack to the effort required to implement the solution. In cases where the implementation effort is a priority, the solution suggested by our method may not be preferred.

Also note that our method suggested two non-applicable swaps. The problem stems from the design of the algorithm that always selects the minimally satisfied goals and suggests swapping it with another goal, without checking whether the second goal can be further reduced or improved. For example, effort was Medium High, and increasing the effort more than two units was not possible. Currently, this problem is alleviated by storing rejected swaps, so the tool would not suggest them in the next steps.
7.2 Comparison of Contributions with Existing Work

This section compares contributions of the framework proposed in this thesis with existing similar methods. This section evaluates the framework by summarizing and comparing the results of case studies as well analytical comparison of advantages and disadvantages of existing methods for modeling security requirements, risk assessment, and requirements decision analysis.

7.2.1 Security Requirements Modeling Component

In Chapter 4, we evaluated the modeling notation through an exploratory study and several modeling case studies. We compared the expressiveness and reasoning power of the proposed modeling notation with similar goal-oriented notations such as [113, 98], as well as Misuse cases and CORAS [140, 29].

Different security modeling approaches enable expressing certain aspects and lack conceptual modeling constructs to represent some other. Table 7.8 compares existing modeling notations based on the security concepts that they express and usage of the models. In this table, N indicates that the modeling notation does not consider the concept in its conceptual modeling constructs and Y indicates the concept is considered (implicitly or explicitly) in the notation.

Some of these conceptual modeling frameworks express vulnerabilities in various ways. Table 7.9 compares capabilities of existing vulnerabilities in conceptual structures. In this table, N indicates that the concept or relation is not considered, and Y indicates the relation is considered explicitly in the notation. P means the relation is implicitly or partially considered or its semantics are not well defined.

The main contribution of our proposal compared to the surveyed approaches is that it expresses how vulnerabilities enter into the system and spread out. The existing modeling notations do not provide means to link vulnerabilities to the actions that actors perform or assets they use. Among these modeling notations, CORAS [29] does not express which design choices, requirements, or processes have brought the vulnerabilities to the system, and the semantics of relationships among vulnerabilities, and between vulnerabilities and threats are not defined. Similar to CORAS, Tropos-based models in [105, 106] do not specify how vulnerabilities are
Table 7.8: Summary and comparison of security modeling notations

<table>
<thead>
<tr>
<th>Security Modeling Notations</th>
<th>Attack</th>
<th>Security solution</th>
<th>Vulnerability</th>
<th>Asset</th>
<th>Functional Req.</th>
<th>NFRs</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attack Tree [135]</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Refining and assessing attacks</td>
</tr>
<tr>
<td>Fault Tree [144]</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Assessing likelihood of failures</td>
</tr>
<tr>
<td>Attack Nets [107]</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Modeling steps, concurrency, and attack progress</td>
</tr>
<tr>
<td>Attack Graph [124]</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Vulnerability and attack modeling and analysis in computer networks</td>
</tr>
<tr>
<td>Abuse Case [108]</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Abstract modeling of abuses of functionalities</td>
</tr>
<tr>
<td>Misuse case [140]</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Modeling attacks in conjunction with functional requirements</td>
</tr>
<tr>
<td>Extended Misuse Case [130]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Modeling vulnerabilities to identify possible threats</td>
</tr>
<tr>
<td>CORAS Risk Modeling Notation [17]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Model-based risk analysis</td>
</tr>
<tr>
<td>Secure Tropos [114, 113]</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Incorporating security concerns into the agent-oriented development</td>
</tr>
<tr>
<td>Improvements on Secure Tropos [105]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Security risk management in early phases of development</td>
</tr>
<tr>
<td>Improvements on Secure Tropos [113]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Security risk management in early phases of development</td>
</tr>
<tr>
<td>Anti-goal [142]</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Analysing and resolving anti-requirements and vulnerabilities in KAOS [28]</td>
</tr>
<tr>
<td>Dependency network analysis [98]</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Identifying and analysing threats in actors’ dependency networks</td>
</tr>
<tr>
<td>UMLsec [81]</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Expressing security relevant information within UML diagrams</td>
</tr>
<tr>
<td>SecureUML [99]</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Model-driven development of secure systems</td>
</tr>
<tr>
<td>Proposal in this thesis</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Model-based security requirements trade-off analysis</td>
</tr>
</tbody>
</table>
Table 7.9: Comparison of modeling notations for expressing security vulnerabilities

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Conceptual Foundation</td>
<td>Extended KAOS framework [143, 28]</td>
<td>CORAS UML-profile based models</td>
<td>Secure Tropos</td>
<td>i* framework</td>
<td>Misuse case models</td>
<td>i* framework</td>
</tr>
<tr>
<td>Vulnerability graphical representation</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relation of vulnerabilities to vulnerable elements</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>P</td>
<td>Y</td>
</tr>
<tr>
<td>Relation of vulnerabilities to other vulnerabilities</td>
<td>N</td>
<td>P</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>Propagation of vulnerabilities to other system elements</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Effects of vulnerabilities</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>P</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Severity of vulnerabilities</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>Relation of vulnerabilities and attacks (exploitation)</td>
<td>Y</td>
<td>P</td>
<td>P</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Countermeasures’ impacts on vulnerabilities</td>
<td>Y</td>
<td>P</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Steps of vulnerability exploitation (sequence)</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
</tbody>
</table>

brought to the system (and by what actions and actors).

The link between attacks and vulnerabilities are implicitly and explicitly modeled in all of the surveyed approaches. However, among the modeling notations that provide explicit constructs for modeling vulnerabilities, only a few frameworks such as CORAS [29], and extensions of misuse case models [130], and our proposed notation relate the countermeasures to vulnerabilities. The semantics of the countermeasures’ impact in [29, 130] is not well defined, and the model cannot be used to evaluate the impact of countermeasures on the overall security of the system, higher level requirements, and business goals. CORAS models express impacts of vulnerability exploitations as unwanted incidents, while our modeling approach enables propagating the impacts of vulnerability exploitations and countermeasures through a goal graph.
and a network of dependencies. This enables evaluating the ultimate risks of vulnerabilities and security gains of countermeasures.

Although modeling and analyzing the sequence of actions to accomplish an attack may affect the implementation of effective countermeasures, many existing frameworks for security requirements engineering (including our proposal) do not incorporate the concept of sequence (temporal order) into their meta-model. The purpose of our security modeling approach is to analyze and compare ultimate consequences of alternative security solutions on risks and requirements; thus, we omit temporal information and focus on impacts of solutions and attacks in a (final) stable state of the system.

7.2.2 Risk Assessment Component

Varieties of security requirements engineering frameworks support security risk assessment, i.e. identification, managements, and evaluation of threats and their probability and damage. Contributions and limitations of the risk assessment component in this thesis, compared to existing methods are as follows:

- **Risk assessment by vulnerability metrics:** the risk assessment method in this thesis takes advantage of the CVSS vulnerability evaluation metrics for evaluating the probability and damage of potential attacks. Compared to Secure Tropos [105], SAEM [22], CORAS [30] and industrial practices such as OCTAVE [4], Microsoft SDL [57], and NIST SP 800-64 [59], our method provides the analysts with concrete metrics that can be evaluated objectively, because the risk score is broken into finer-grained indicators.

- **Qualitative CVSS assessment:** the original CVSS [26] and the SAEM risk assessment method [22] rely on availability and accuracy of quantitative risk metrics or indices. In our adaptation of CVSS, we evaluate the risk of vulnerabilities using qualitative labels of Low, Medium, and High. This helps to deal with unavailability or inaccuracy of risk measures in the early requirements phases.

- **Model-based risk evaluation:** a common advantage of our method and other model-based risk assessment approaches (e.g., CORAS [30], Microsoft SDL [57], Secure Tropos
Chapter 7. Evaluation and Contributions

[105]) is using security models as the risk assessment reference model. Security models in general help reason about the criticality and potential harm of threats. The i*-based security models (compared to CORAS [30] and Microsoft SDL [57]) enable tracing the impacts of vulnerability exploitations on system requirements and stakeholders’ goals through the dependency network and goals hierarchy.

- **Knowledge-intensive assessment process:** the analysis of security requirements and risks is a knowledge-intensive process, that requires analysts have a minimum understanding of security principles and threats for identifying vulnerabilities and assessing their risks. Our method, compared to other risk assessment methods, relies less on the security knowledge of the analysts and more on the models and metrics on which security requirements are analysed. For example, we argued that tasks and resources expressed in i* models are the source of vulnerabilities. The models guide the analyst in searching the vulnerability databases. The advantage of CVSS is that it provides a set of vulnerability analysis metrics, which helps requirements analysts to estimate the risk of threats without an extensive background in security.

- **Scalability issues:** although model-based risk assessment approaches, as suggested in this thesis and other proposals such as Secure Tropos [105] and CORAS [30] facilitate risk assessment, developing accurate security models can become a labor-intensive bottle-neck in the project. i* and Tropos models tend to be more complex than CORAS models and CORAS models are more complex than threat models in Microsoft SDL [57]. On the other hand, complex goal models are more expressive than simple data flow diagrams in Microsoft SDL [57] or CORAS risk models, and help analysts assess risks with respect to consequences of malicious intentional elements.

7.2.3 Decision Analysis Component

In Chapter 2, we reviewed a wide variety of requirements trade-off analysis and decision making methods. This section discusses advantages and disadvantages of existing methods compared to the decision aid methods we introduced in Chapter 6.
**Requirements Preference Elicitation Methods:** The trade-off formulation and analysis approaches surveyed in Chapter 2 express trade-offs at different levels of abstraction. In many engineering fields, mathematical formulas model the utility functions and relationships between trade-off parameters. Quantitative approaches such as Utility Theory can help make a decision between alternative solutions. Preference elicitation methods such as AHP [131], SMART [146], Weighted Set Value Statistics [148], enable extracting importance weights of criteria to calculate the utility values. For example, the preference model-driven service selection method in [101] applies Weighted Set Value Statistics to decide between alternative services expressed in $i^*$ models. Both methods in [73, 101] address conflicting preferences among different stakeholders and provide methods for negotiating and resolving such conflicts. These methods ultimately calculate a utility value for each solution to select the best option. The requirements decision analysis method in [44] is also based on a quantitative model elicited by the “coarse quantification” of relevant risk factors and their interactions. The main benefits of our method compared to these approaches are:

- Our decision aid methods do not rely on extracting preferences in terms of numerical importance weights.
- Preferences of stakeholders can strongly depend on the context of possible solutions and their impacts [126], and most of the preference elicitation methods extract these preferences independent of such contexts. Our method is based on Even Swaps for extracting value trade-offs of stakeholders in the context of a solution, a set of criteria, and the impacts of solutions on the criteria.
- Our method enables a systematic decision analysis process even if utility values or utility functions for different solutions cannot be defined, or some of the criteria are measured quantitatively and some are evaluated qualitatively.

**Trade-off Analysis Methods:** By using the AORDD [70] and SVDT [68] methods, a fixed set of trade-off parameters are expressed in a Bayesian Belief Nets (BBN). The BBN topology visually expresses the relation between trade-off parameters. In this way, a single
fixed formulation of relationships between trade-off parameters is reused for multiple projects, which reduces the effort to extract and analyze security trade-offs dramatically. The BBN is fed with quantitative input about static security level, dynamic security level, standards, policies, laws and regulations, priorities, business goals, security risk acceptance criteria, time-to-market, and ultimately the BBN calculates a Return on Security Investment (RoSI) for each security solution, which helps objectively decide between alternative solutions. The main benefits of our method compared to these approaches are:

- Our method does not rely on a fixed set of trade-off criteria, and analysts can define project-specific goals as the trade-off parameters.
- Our method does not rely on availability of quantitative input data to calculate a final utility or return on investment, thus, analysts can make a systematic decision if such data are not available or are costly to collect.

Using the ATAM [88] and CBAM [13] approaches, trade-offs are recorded by the lists of tactics, scenarios and quality attributes in tables. Trade-offs are analyzed by an architecture expert that evaluates each design tactic in terms of its costs and benefits. These approaches are labor-intensive, and ATAM analysis is entirely based on experts’ judgement. The main benefits of our proposal compared to ATAM and CBAM are:

- Our method does not rely on subjective judgement of analysts about the benefits and costs of different architectural solutions.
- Our methods are supported by an algorithm for extracting and applying preferences over trade-off criteria to select a solution rather than experts negotiations (as in ATAM).

Using the misuse case models for trade-off analysis [5], one can explicitly express the (mis)use cases that have negative or positive impacts on other (mis)uses. Although trade-offs are expressed explicitly, the (mis)use case analysis is not supported with a decision analysis approach to choose between alternative security use cases. The main benefit of our decision aid method compared to trade-off analysis based on misuse cases are:
• The input to our decision analysis methods are extracted from goal-oriented models. These models help to reason about the vulnerability metrics as well as consequences of threats and security solutions.

**Goal-Oriented Models for Decision Analysis:** There are a variety of requirements decision analysis methods that take advantage of goal-oriented modeling and reasoning. For example, Bresciani et al. [21] suggest a quantitative security requirements analysis on Secure Tropos. In this method, to calculate the security criticality of the system, a value within the range of 1-5 is assigned to each security constraint in a dependency relationship. Aguilar et al. [3] propose a goal-oriented approach for optimizing non-functional requirements based on finding Pareto front configurations that maximize softgoals satisfactions. In this approach, the value of \( i^* \) Help contribution is +1, Some+ is equal to +2, and Make is +4, and the negative impacts of Hurt, Some-, and Break are -1, -2, and -4. The common limitation of these methods is arbitrary assignment of numbers to \( i^* \) and Tropos consequences. The benefit of our decision analysis method compared to [21, 3] is:

• Our method does not rely on assigning arbitrary numerical weights to goal model consequences. These numbers do not have any objective interpretation in the application domain [95]. Although they are detailed, these numbers are mostly based on the subjective opinion of analysts rather than measurements or information collection.

Similar to the underlying philosophy of our work, Letier and Lamsweerde [95] also argue that goal models usually lack accurate and measurable goal formulations. They argue that the rules for impact propagation through goal models are not universally agreed. They claim domain-specific quality variables are needed to reason on partial goal satisfaction. They specify goals in a precise probabilistic way, and the impacts of alternative decisions on the degree of goal satisfaction are analyzed numerically [95]. The introduction of quality variables in [95] is similar to the use of vulnerability metrics in our method. Our decision analysis method can provide additional benefits compared to the method in [95]:

• Our method enables making a decision when quality variables, decision criteria, and vulnerability metrics are measured with a mixture of quantitative or qualitative values.
Even if coarse qualitative evaluations are not available, our method enables systematic decision analysis based on comparing alternatives and applying some heuristics.

- Our method implicitly extracts and applies preferences of stakeholders to the decision problem.

Techne [80] is a requirements modeling language for expressing desired conditions as goals, softgoals, and constraints, as well as domain assumptions, such as optionality and conflicts of goals and preferences of stakeholders over the goals. Techne models can be analyzed to find a combination of tasks and domain assumptions that satisfy all compulsory goals and quality, also aiming to satisfy as many preferred optional goals as possible. Techne reasoning can be beneficial when alternative solutions cannot be measured quantitatively or qualitatively and the preferences of stakeholders cannot be expressed by relative rankings or importance weights. However, our method overcomes some of Techne limitations:

- Techne is not able to express and extract different preference levels among decision criteria. Our method helps stakeholders distinguish different levels of preferences over decision criteria, without requiring the stakeholders to provide importance weights or relative rankings directly.

- Techne reasoning is based on propositional logic, i.e. softgoals are assumed either satisfied or denied. In many decision cases, satisfaction of non-functional requirements and stakeholders’ goals is not a binary value, and depending on the available data, analysts might be able to extract finer-grained evaluations. Our method enables decision analysis with a mixture of evaluations in different scales and different granularity levels.
Chapter 8

Tool Support for Decision Analysis

This chapter describes a decision aid tool that automates the decision analysis methods proposed in Chapter 6. The tool enables analyzing a decision problem when different criteria are quantitatively and/or qualitatively evaluated; or, instead of measurement, alternatives are compared over different criteria.

The tool adopts the Even Swaps multi-criteria decision analysis approach [64] to make trade-offs among requirements. This tool semi-automates the Even Swaps process, in the sense that the process is still interactive with stakeholders, but suggesting and applying next swap is automated by an algorithm. The tool suggests which requirements to select for the next swap, and at the same time, justifies each suggestion for the user.

In this chapter, we first describe the features of the tool. Then, the architecture design and interfaces of the tool are described. Finally, we discuss the benefits and limitations of the tool and weaknesses of the architecture design.

8.1 Features and Functionalities

The decision aid tool has two main modules:

1. A decision analysis module based on comparing alternatives

2. A semi-automated Even Swaps module

In this section, we discuss features and functionalities that each module provides.
8.1.1 Semi-Automated Even Swaps Module

This module aids decision analysts in making trade-offs among goals and requirements to select the best solution when several alternatives are available. This module is suitable for cases where decision criteria can be evaluated quantitatively and/or qualitatively. The sequence of functionalities in this module is as follows:

1. The tool first collects the list of goals and alternatives from the user. To apply the Even Swaps method users need to evaluate alternatives with respect to their decision criteria. They can choose to evaluate the criteria on a scale of their choice and aim to minimize or maximize the criteria. Figure 8.1 illustrates the graphical user interface for collecting initial input from decision stakeholders.

2. The user can select to analyze the decision problem with the Even Swaps module (the second tab in the interface in Figure 8.1). The Even Swaps cycle starts with selecting a pair of alternatives to be compared and analyzed (Figure 8.2 Steps, a and b).

3. After the tool selects the best pair of alternatives to be analyzed, it lists useful swaps and calculates the distances between criteria for each swap (Figure 8.2 Step c).
4. Based on the rules for identifying the best and easiest swap (Chapter 6, Section 6.2), the tool suggests a swap to stakeholders (Figure 8.2, Step d).

5. Swaps are asked from stakeholders in a separate Java Dialog. Stakeholders can reject the suggested swap, if they cannot express their value trade-offs through the suggested swap (Figure 8.2, Steps d and e).

   - If stakeholders are not able to swap suggested goals, the tool suggests the next best swap and stores the rejected swap in a swap “black list”, and in the rest of the process, swaps in the black list will not be suggested to the stakeholder.

![Image](image_url)

**Figure 8.2:** The Even Swaps method is semi-automated and the tool interacts with stakeholders for making suggested swaps

6. If stakeholders are able to make the swap, the tool applies the swap to the decision problem and updates the table. The tool also stores the swap in a reusable swaps list and these stored swaps can later be reused for other alternatives.
7. The dominated solution is removed from the list of alternatives (Figure 8.3, Step a), and a new cycle starts with comparing and analyzing a new pair of alternatives (Figure 8.3, Step b).

8. These cycles of analyzing a pair of alternatives and removing the dominated one continues, until one solution is left which is the overall best solution.

8.1.2 Comparison-Based Decision Analysis Module

This module aids decision analysts in making trade-offs among goals and requirements when decision criteria cannot be evaluated, but instead alternatives are compared. The sequence of functionalities in this module is as follows:

1. The tool first collects the list of goals and alternatives in the tab shown in Figure 8.1.

2. By going to the “Comparison” tab, the user can choose to analyze the decision problem
by comparing the alternatives (Figure 8.4, step a). In this tab, the tool interacts with users, and for every decision criteria, the user selects the alternative that satisfies the decision criterion better, out of the pair of solutions (Figure 8.4, step b).

3. The user also specifies the extent of difference between the alternatives. As discussed in Chapter 6, the choice of scale’s granularity is enforced by the humans’ cognitive limits (limited to nine distinct levels). By using nine levels, numerous possible consequences are generated, and thus, numerous swap queries would be asked from stakeholders. Thus, we use a less granular scale (0 to 5) for comparing alternatives to generate possible consequences (Figure 8.4, step c).

4. Comparisons of alternatives over the decision criteria are used to generate all possible consequences of alternatives. These possible consequences are analyzed using the Even Swaps module (Figure 8.5). Similar to the process shown in Figure 8.3, the dominated solution is removed from the problem. This process continues, until one solution remains, which is the best available solution.
8.2 Tool Design and Architecture

This section overviews the tool architecture. We present a schematic class diagram that illustrates the main classes of each module. This tool is implemented in Java and the graphical user interface (GUI) is created by using Java Swing library. We explain dependencies and relationships among different components in the architecture. The tool consists of four main packages:

1. GUI package
2. Data structure package
3. Automated Even Swaps algorithm package
4. Comparison-based algorithm package

Figure 8.6 illustrates the relation between these and the primary classes in each package. In this diagram, only the main classes, and their important attributes and methods that carry out the main functionalities and features are illustrated. Utility classes are omitted for the sake of simplicity.
Figure 8.6: Main classes and packages of the Even Swaps module

**GUI Package:** This package separates the graphical user interface (GUI) from the implementation of the algorithms. The main class in this package, TabbedUI, extends a JFrame (from Swing library in Java). This class contains a JTabbedPane, with three different tabs. In the first tab (as shown in Figure 8.1), alternatives and goals are entered into the tool by the user. TabbedUI stores alternatives, goals, and the impact of the alternatives on the goals in a data structure class called ConsequenceTable in the DataStructure package. TabbedUI is responsible for reading, updating, and highlighting this table (both the data table and the graphical representation).

The GUI package contains two main Java classes that are invoked by the algorithms when needed. AskSwapSimpleDialog, extends JDialog (from Swing library in Java) and is responsible for asking the suggested swap from the user. Figure 8.2 shows this JDialog. The second
JDialog in the GUI package is compareJDialog which is responsible for interacting with users to compare alternatives over decision criteria. Figure 8.4 shows this JDialog.

**Data structure Package:** The data structure package contains classes that act as the fundamental data constructs in both algorithms. ConsequenceTable which is initially populated in TabbedUI contains alternatives, goals, and consequences of alternatives on the goals. Later the Automated Even Swaps algorithm gets a ConsequenceTable object and applies the Even Swaps to it. Alternatives is a class for storing alternatives and analyzing their properties. For example, this class provides methods for searching in the list of alternatives and finding the most distant pairs.

**Automated Even Swaps algorithm package:** The automated Even Swaps algorithm that handles both quantitative and qualitative data is mainly implemented in the AuthoEvenSwapAlgorithm package. The Algorithm class implements two loops: one loop over alternatives until the overall dominant solution remains, and an inner loop over criteria, until between a pair of alternatives, one solution dominates the other with respect to all remaining criteria. Algorithm receives a ConsequenceTable object from the tabbedUI JFrame.

The Algorithm depends on the EvenSwaps class to apply the Even Swaps method. The EvenSwaps class implements the rules we proposed for selecting reusable, easy to make, and useful swaps in Chapter 6. This class keeps a list of reusable and black list swaps by instantiating objects from Swap class. The swap objects are data structures for storing and managing even swaps.

**Comparison-based algorithm package:** This module implements the comparison-based decision aid algorithm. The compareJDialog in the GUI package is the user interface that gets the users’ comparisons. The comparisons are stored in the data structure provided in the Comparison class. The CaseGenerator takes these comparisons and generates all possible consequences of the alternatives. The ComparisonAlgorithm class implements the loops over the alternatives and possible consequences of alternatives that CaseGenerator created. The ComparisonAlgorithm class works tightly with ComparisonEvenSwaps class to apply the Even Swaps to possible consequences. ComparisonEvenSwaps has methods similar to the EvenSwaps class.
8.3 Tool Benefits and Limitations

**Benefits of the tool:** The main benefit of the automated Even Swaps module is to reduce the burden on the human analyst, by automatically finding the best swap for the next step. It also stores reusable decisions and applies them without the need for human analyst interference. The comparison-based decision analysis module makes decision analysis, in the absence of measurement, possible. Without an automated tool, generating and analyzing numerous possible consequences is manually impossible.

**Limitations of the tool:** Both modules of the tool have limitations that need to be improved in future. Currently, the tool does not detect criteria that have a high priority for stakeholders. If such knowledge is extracted from the swaps that stakeholders make, it can be used to suggest smarter swaps in the next steps. For example, if the priorities of stakeholders are known, the algorithm can avoid suggesting swaps that require a significant sacrifice on highly important goals.

In our Even Swaps implementation, the minimally satisfied goal is swapped with the maximally satisfied goal. We justified this strategy in Chapter 6, Section 6.2. In this way, sometimes the suggested improvement on the minimally satisfied goal is significant, and if it is a highly important criterion, stakeholders may not be able to find a goal to sacrifice that matches the significant improvement on the important criterion. An example of such a swap is given in the case studies in Chapter 7, under the analysis of Cross-Site Scripting solutions. To avoid rejected swaps the algorithm needs to consider whether the suggested swap would request extreme sacrifices on the goals. This feature is not implemented and tested in the current implementation.

The main limitation of the comparison-based analysis module is the scalability problem. When consequences of alternatives are compared on the scale of Low to High, if all alternatives are different by a Low extent, for $m$ criteria, the tool generates $5^m$ possible consequences. For example, if the decision needs to be made with respect to 10 criteria, nearly 10 million possible consequences would be generated ($5^{10}$). The execution speed of the tool is currently slow when handling a large number of possible consequences. However, the even more critical problem is that the tool may require several even swaps queries from stakeholders, which practically makes
the method inapplicable. Thus, the proposed tool is only suitable when either a small number of criteria (6 or 7) needs to be considered, or if more criteria are considered, alternatives have distinguishable consequences on the criteria.

**Comparison with similar tools:** The need for automated tool support for suggesting even swaps to decision analysts has been recognized in [115, 116]. In [115], consequence changes made in an even swap are not assumed to depend on the satisfaction levels of the goals, which allows using the trade-off information given in one even swap to represent the stakeholders’ general preferences over the goals. The Smart Swaps tool [116] suggests next swaps to the user based on preference programming assumptions made in [115]. The Smart Swaps tool navigates the swap chains toward 1) making goals irrelevant and 2) making alternatives one step closer to the dominance situation.

In addition to those two goals, our tool introduces other criteria for suggesting the next swaps, such as reusability of swaps, swaps that are cognitively easy to make, preferring numerical scales to qualitative ones, etc. Our tool is tailored for requirements trade-off analysis, e.g., the tool suggests swaps that stakeholders are willing to make among requirements with minimal sacrifices to be made. Our tool is developed based on the opposite assumption made in [115], i.e. we assume that consequence changes made in an even swap do not represent stakeholders’ general preferences. A demo of the tool is available at [2].

**Architecture design weaknesses:** The architecture design of the tool can be significantly improved as well. ComparisonEvenSwaps and EvenSwaps classes share most of their methods and ComparisonAlgorithm and Algorithm are similar in terms of their data structures and main methods. To make the design more expandable and easy-to-maintain, the best approach is to define a Java “Interface” for the Even Swaps functionalities, where ComparisonEvenSwaps and EvenSwaps both implement the interface in their own specific way. This helps to add new implementations of the Even Swaps method with less effort. In addition, the implementation of the Even Swaps algorithm will be independent of the objects or modules that will interact with and invoke the Even Swaps implementation.

The best approach to implement ComparisonAlgorithm and Algorithm classes is to define a Java Abstract class that contains general methods and data structures required to implement a
decision analysis algorithm. ComparisonAlgorithm and Algorithm would extend the abstract algorithm class. This helps develop new algorithms more quickly by reusing the high level implementation in the abstract class.

Current architecture of the tool does not meet the desired design that we discussed above, because the current tool is the result of combining two separate tools into one interface. The two main modules of the current tool have been separately developed in different environments. In future work, further integration is needed to improve the expandability and maintainability of the architecture.
Chapter 9

Conclusions and Future Work

The objective of this thesis is to provide software analysts with a framework for identifying and making trade-offs between security and other requirements. We introduced a security requirements modeling notation for specifying possible attacks and vulnerabilities within goal-oriented requirements models of the software system. The resulting models are the basis of risk assessment, which involves evaluating the probability and damage of vulnerability exploitations within the system specified in the model. Vulnerabilities are re-evaluated in the presence of security solutions. We propose decision analysis methods to select good-enough security solutions among multiple alternatives.

The decision aid methods help make trade-offs among security and other goals. The basis of decision making is to evaluate and/or compare consequences of each alternative security solution on requirements and stakeholders’ goals. Each solution has positive and negative impacts on different requirements and goals, and to select the best “available” solution, stakeholders need to make trade-offs between their goals and objectives. We propose an algorithmic method that helps decision stakeholders apply their value trade-offs to find the solution that best satisfies their important goals.

In summary, this thesis provides three main components that assist analysts in eliciting security requirements, identifying security trade-offs with other goals, and selecting a good enough solution based on the risks and side-effects of security solutions. We worked out several
case studies to evaluate the utility of the proposed components. Based on the evaluations in the case studies, we have drawn some conclusions and identified limitations of the methods in the following sections. This chapter also discusses future plans to improve this work and expand its utility and usability.

9.1 Conclusions and Limitations

This section summarizes the main contributions of each component in this thesis as well as weaknesses and limitations of the methods.

9.1.1 Modeling Component

The proposed modeling notation in this thesis helps express security vulnerabilities and attacks within $i^*$ models, which provides the following benefits:

**Proactive Vulnerability Analysis:** The central idea of this thesis is analyzing security requirements based on potential vulnerabilities and their risks. This is a pro-active approach to build systems whose vulnerabilities have been anticipated and addressed. The vulnerability-centric analysis of security requirements distinguishes our work from previous goal-oriented security analysis modeling approaches such as [114, 51, 98, 113, 105].

**Model-Based Risk Assessment:** The main advantage of this approach compared to industrial threat analysis methods such as NIST SP 800-64 [59] and OCTAVE [4] is that our approach is model-based. Security requirements modeling bridges the requirements analysis activities with the security analysis. The modeling process structures the analysis and reasoning that system analysts undertake. The models serve as artifacts and reasoning tools, and wherever justifications are needed to support an evaluation, analysts can refer to certain views or layers of the model.

In section 7.2, we extensively discussed the advantages of our security requirements and threat modeling approach, compared to other model-based security analysis methods ([135, 140, 17, 113, 105, 142, 98]). Our agent- and goal-oriented approach helps analyze the system from the point of view of both the defenders and attackers. The goal and task breakdown facilitates
discovering vulnerabilities in the fine-grained resources and tasks. The \(i^*\) contribution links are highly expressive modeling elements to capture the impacts of vulnerability exploitations, as well as security solutions remediations and side-effects. Attack Trees, Misuse Cases and CORAS risk models [135, 140, 17] lack explicit constructs to capture consequences, and the structure of those models does not support tracing the impacts from one element to another. The modeling notations based on Secure Tropos [113, 105] provide an explicit element to model security threats; however, threats are high-level and abstract elements in the resulting models and their source actor or intentions are not expressed. Our approach deals with threat modeling in the same fashion that modeling the goals and dependencies of usual system actors is approached; thus the resulting threat models capture intentions of attackers, their task break-downs, and their alternative attack scenarios.

**Vulnerability Analysis within Goal-Oriented Dependency Networks:** When modeling a system by the \(i^*\) framework, analysts identify and analyze the main stakeholders of the software system as well as their goals and dependencies. Different components of the system can be expressed as autonomous actors on which other users or system components depend. This provides a suitable foundation to analyze security requirements. We can identify vulnerable entities that are brought to the system through actions taken and resources used by actors. This setting also enables analysts to trace the propagation of vulnerabilities and their impacts from one goal or actor to another in the network of actors’ dependencies. The goals hierarchy and the dependency network of actors help analysts reason about the effects of vulnerability exploitations on goals of different actors.

Anti-goal models [142] express vulnerabilities and link them to attack scenarios; however, our approach is more expressive, since it can express the relation between vulnerabilities and vulnerable elements. CORAS risk models [17] also explicitly model vulnerabilities; however, neither CORAS, nor Tropos-based models [105, 106] specify how vulnerabilities are brought to the system, by what actions, or actors. The structure of our models also help propagate vulnerabilities to other system elements or actors.

**Attack Analysis:** The modeling philosophy of \(i^*\) enables expressing attacks and attackers, in the same way the business domain and other actors are analyzed. Attack modeling ultimately
facilitates understanding attackers’ intentions, their means of attacks, exploitation scenarios they launch, and their dependencies on other actors and resources. Ultimately the models provide a strong basis for reasoning about the risk of vulnerabilities in the risk assessment component. Compared to well-known attack modeling approaches such as Attack Trees [135] and Misuse Cases [140], our approach is more expressive (at the cost of being more complex). Our modeling notation enables expressing the task hierarchy of attacks (similar to Attack Trees), but it also relates the attacker’s tasks to his/her goals. The attacker is analyzed as an intentional agent with dependencies and capabilities, while in the Misuse Case model, the focus is only on modeling the unexpected behavior of the attacker.

**Risk-Centered Countermeasure Analysis:** The proposed modeling process ends with adding a countermeasure layer to the security requirements model. The same i* modeling principles are reused to model and reason about the impacts of the countermeasures. The introduction of countermeasures involves inserting additional tasks, resources and even actors to the requirements model. Each of the newly added elements can be the source of new vulnerabilities in the setting; thus, the modeling process is iterative. Our proposed notation relates the countermeasures to vulnerabilities, while the semantics of the countermeasures’ impact in CORAS and Misuse Cases [29, 130] is not well defined. The CORAS model, although expressive and complex enough, cannot be used to evaluate the impact of countermeasures on the overall security of the system, higher level requirements, or business goals.

**Limitations:** The proposed security requirements modeling notation in Chapter 4 is an extension of the i* framework, and thus, it inherits limitations of the i* notation. As noted in [112], the i* notation has some visual and semantical weaknesses such as symbol redundancy and overload, complex perceptual discriminability, and excessive graphic complexity. Above all, the most general and immediate problem in requirements models including i* is the complexity management [23, 112]. i* lacks effective complexity management mechanisms such as views, filters, and sub-models. This thesis does not address the model management issues of i* models. We acknowledge that security extensions to the i* notations (vulnerabilities, attacks, and countermeasures) add to the complexity of the model even more. To manage the complexity of security requirements models, we deal with security-specific elements as different views
and layers of the same model. For example, vulnerabilities are added as a layer on top of the requirements view. This layer can be on taken off the model, which helps to understand and manage the resulting models. An immediate weakness of this proposal is that the modeling notation is not implemented in a tool to illustrate and test the effectiveness of the layers and views concept.

The other major issue with applicability of the proposed modeling notation is the knowledge-intensive process for developing vulnerability layers, attack views, and countermeasure layers. In contrast to the common belief that most requirements engineers are poorly trained to elicit, analyze, and specify security requirements [46], we have collected empirical evidence that only a small percentage of software professional have not received any security training [31]. Based on these results, we assume that requirements analysts have the minimum security background and knowledge to use vulnerability portals such as Common Weakness Enumeration (CWE) and find relevant vulnerabilities within the requirements model they have developed.

In Section 4.6.1, Chapter 4, we reported initial attempts to empirically evaluate the usefulness of \textit{i*} modeling for security requirements analysis. This study is limited to the comparison of goal-oriented models and natural language, but it is weakened due to varieties of biases such as subjective evaluation of participants’ performance, statistically-insignificant data, previous domain knowledge and modeling skills of subjects, that threaten the validity of such studies.

### 9.1.2 Risk Assessment Component

The second component of this thesis is a risk assessment method, that adopts and adapts Common Vulnerability Scoring System (CVSS) for evaluating the probability and damage of vulnerability exploitation scenarios within the requirements models. The contributions of this component are:

**Risk Assessment Based on Metrics:** The underlying philosophy of the risk assessment method in this thesis is to decompose risks into granular metrics that are measurable or at least more objective to evaluate than a single risk value. This helps reduce the subjectivity of the risk value.

Evaluation of risk metrics is based on reasoning on relevant elements of the model. This
is an advantage compared to other model-based risk assessment methods such as CORAS [30], Anti-models [142], and Secure Tropos [106].

**Qualitative CVSS Assessment:** The proposed risk assessment method targets the situations where measuring the probability and damage of vulnerability exploitations is not possible and analysts have to rely on a rough guess about the risk value. Compared to quantitative risk analysis methods such as [22], our method enables evaluating risks in the absence of numerical data about the damage and probability of attacks. To avoid assigning a subjective risk value to a vulnerability, we adopted and adapted the CVSS for evaluating the probability and damage of vulnerability exploitations. CVSS metrics take the properties of the vulnerability, environment, and exploitations of the vulnerability into consideration to calculate a normalized severity rating. This adaptation can extend the practicality of the original CVSS methods in the early requirements phases.

**Evaluating CVSS Metrics by Rationalizing on the Model:** A security requirements model, with a vulnerability layer and an attack view, provides analysts with the rationale to evaluate CVSS metrics. For example, to evaluate the Authentication metric, the attack view can be the reference on whether authentication is part of the exploitation scenario. Compared to the CORAS risk assessment method [30], our goal-oriented modeling approach helps to reason about the consequences of vulnerabilities and attacks, while CORAS models do not support consequence analysis. Compared to industrial risk and threat analysis methods such as OCTAVE [4] and Microsoft SDL [57], the main benefit of applying CVSS to security requirements models is supporting CVSS evaluation based on the model.

**Limitations:** The main justifying argument to support our claims is that quantitative risk measures provided by stakeholders can be inaccurate, subjective, and thus misleading. However, our method ultimately relies on collecting coarse qualitative evaluation of risk metrics such as exploitability and authentication effort. Although evaluating these metrics is supported by the models, the incomplete and imprecise models can result in inaccurate evaluations. Ultimately, our method inherits the original problem that we targeted to solve, since the analysts’ judgement of CVSS metrics can still be subjective and inaccurate.

The benefit of our method is that the probability and damage of vulnerability exploitations
are broken down into fine-grained vulnerability metrics. Thus, instead of directly asking stakeholders to come up with a high-level risk value, finer-grained variables help them reason about the severity of the risks. Thus, although inaccurate evaluations can threaten the validity of risk assessment results, our method reduces subjectivity during data collection.

The other major weakness of the proposed risk assessment method lies in the aggregation method we used to unify risk metrics into a probability and damage score. By aggregating several risk metrics that are evaluated in the scale of Low, Medium, and High to a unified score in the same scale, the resulting value is not as precise as the several individual metrics. This ultimately results in losing the impacts of one or more metrics in the final risk value. An example of this problem was shown in Table 7.7, in which, risk metrics are modified due to the presence of a security solution, but the final aggregated probability does not reflect the security solution effects (probability did not change). In this particular example, to reflect the impact of solutions, the probability and damage of Cross-Site Scripting were expressed in a more granular scale than Low, Medium, and High (See Table 7.7).

9.1.3 Decision Analysis Component

The third component in this thesis is a decision aid method that takes risk and security solutions, and extracts stakeholders’ value trade-offs to aid decision makers in selecting a solution. The decision analysis component tackles the problem of making trade-offs in the absence of numerical measurements and/or the unavailability of accurate evaluation of decision criteria and alternative solutions. We proposed two decision aid methods that are suited for different conditions and provide the following benefits:

**Decision Aid with a Mixture of Quantitative and Qualitative Values in Different Scales:** In Section 6.2, we proposed a decision aid method for situations where decision criteria can be measured or evaluated in different scales, qualitatively and/or quantitatively. To aid the decision analysts we developed an algorithm that automates the Even Swaps multi-criteria decision analysis method. Adopting Even Swaps is especially useful when some of the requirements and design objectives are measured quantitatively and some are evaluated using qualitative specifications. The major benefit of Even Swaps compared to other preference elicitation meth-
ods such as AHP [131] or SMART [146], is that swaps extract preferences of stakeholders in the context of actual solutions and with concrete impacts on decision criteria, instead of isolated from alternatives. The main argument to support extracting value trade-offs using even swaps is that preferences of stakeholders can strongly depend on the context of possible solutions and their impacts [126], and even swaps extract stakeholders’ value trade-offs in the context of solutions and their impacts.

The requirements decision analysis method in [44] is also based on a quantitative model elicited by the “coarse quantification” of relevant risk factors and their interactions. In the goal-oriented decision analysis approach in [95], impacts of alternative decisions on the degree of goal satisfaction are analyzed numerically. Our method, on the other hand, enables a systematic decision analysis process even if some of the criteria cannot be quantified. Unlike AORDD [70] and SVDT [68], our method does not rely on a fixed set of trade-off criteria, and analysts can define project-specific goals as the decision parameters. Unlike Techne reasoning [80] based on propositional logic, we can deal with satisfaction of non-functional requirements in a range of values in different scales of measurement (and not as a binary evaluation).

Decision Aid in the Absence of Alternative Evaluations: The second algorithm proposed in Chapter 6 addresses the situation where analysts cannot evaluate the consequences of different solutions on the goals and requirements. To address the lack of measurements, the proposed method relies on asking stakeholders and analysts to compare alternatives over different goals. The underlying idea of our method is to narrow down possible consequences of alternatives based on their pair-wise comparisons. The proposed method generates all possible consequences of alternatives, and then, applies the Even Swaps method to each one of those possibilities. This method is ultimately heuristic, i.e. if a solution is dominated for a minority of the possibilities, we have narrowed down the decision to verify whether those minor cases are invalid. We supported the decision aid methods with a prototype tool.

The immediate benefit of this method is that it enables making a decision when contributions of alternatives cannot be evaluated quantitatively or qualitatively. Requirements decision analysis methods such as [148, 44, 73, 101, 86] strongly depend on the availability of some initial alternative scores, while our method allows for decision analysis in the absence of any direct
evaluation of alternative actions. ATAM and CBAM [13, 88] do not heavily rely on the availability of numerical evaluation of alternative tactics; on the other hand, they are labor-intensive trade-off analysis methods that rely on experts’ negotiation and discussion. We proposed a systematic method supported by an algorithmic semi-automated tool for extracting and applying preferences over trade-off criteria to select a solution, instead of relying heavily on the opinion of experts in the subject matter.

**Limitations:** The engine for making trade-offs in our methods is the concept of even swaps. Thus, our work inherits limitations of the Even Swaps method. As we discussed in the final section of Chapter 6, making even swaps may require substantial cognitive abilities and domain knowledge. It might be cognitively demanding for stakeholders to hypothetically give up on one goal for better satisfaction of another. Diversity of evaluation scales in the consequence table can make the swaps even harder. Stakeholders may not be able to swap a goal measured in absolute values with a goal that is evaluated by qualitative labels.

Nevertheless, if stakeholders are not able to specify the maximal amount of a goal satisfaction level that they are willing to sacrifice for a unit of increase in another goal, then it is highly likely that they will not be able to numerically specify preferences over the goals either.

The other potential threat in applying the Even Swaps is the possibility of collecting inconsistent value trade-offs from human stakeholders. Inconsistent user opinion is a threat in any interactive decision aid method that relies on humans’ input. In the Future Work section of this chapter, we will discuss some possible ways to check the consistency of the users’ input.

The main limitation of the comparison-based analysis module is the scalability problem. We discussed in Chapter 8 that in the worst case, for \( m \) criteria, the tool generates \( 5^m \) possible consequences. For example, if the decision needs to be made with respect to 10 criteria, the tool generates nearly 10 million possible consequences \((5^{10} \approx 10^7)\). The main problem is that the tool requires stakeholders to make a large number of even swaps, which makes the method inapplicable in practice. Thus, this approach is only suitable when a small number of criteria (up to 6 or 7) are considered. For cases where analysts deal with more criteria, this method works if alternatives have highly distinguishable consequences on those criteria. This is a significant limitation to applying this method and reduces its application to small-size
problems that could be even manually analyzed.

We explained in Chapter 6, in the Discussion section, that the Marginal Rate Substitution (MRS) can dynamically change. We extensively discussed that MRSs are not static because even swaps heavily depend on the context of the value trade-off. The context is availability of some alternatives and their contributions on the satisfaction of goals [126]. This is the main root of our tool limitation in analyzing millions of possible consequences, because it does not reuse the swaps for many of the cases, and requires stakeholders to make a large number of swaps. In the Future Work section of this chapter, we will discuss potential ways to address this limitation.

**Framework Limitations:** A major limitation of this thesis is lack of a comprehensive evaluation of the entire framework (three components) through a large-scale industrial case study. In general, evaluating the utility and usefulness of the entire framework is challenging. Variety of uncontrolled factors can affect the security trade-off analysis process and its results in an industrial project; for example, the security knowledge of analysts, history of the project, knowledge of previous attacks in the target domain, criticality of the target domain, budget and time dedicated to security requirements analysis, etc. To avoid challenges and potential biases in the evaluation of the entire framework (all three components and the process) through empirical methods and industrial case studies, we focused on analytical study of top web application vulnerabilities (Chapter 7).

### 9.2 Future Work

**Modeling Component:** The immediate next step is to support the proposed modeling notation with a tool for graphically creating security requirements models. The tool can extend existing $i^*$ modeling tools such as OpenOME [120] with security concepts grouped into new layers and views. This requires extending the meta-model of the existing modeling tools, adding security-specific concepts to the visual modeling pallet, and re-organizing the model elements’ containers to support the concept of a requirements view, vulnerabilities layer, attacks view, and countermeasure layer.
The tool ultimately can support adopting the proposed framework in real-world scenarios, but in addition, it facilitates evaluating the usefulness and usability of the modeling approach in empirical studies. We demonstrated the expressiveness and reasoning power of our modeling method in various case studies, benchmarks, and in a limited exploratory study; however, further user studies are needed to provide more solid evidence that modeling a software system, in terms of autonomous actors and their dependencies, provides a suitable foundation for discovering vulnerabilities. Additional empirical experiments are essential to show the usability and applicability of the notation and whether analysts with a minimum security background can use it as the basis of search for vulnerabilities. In such studies, human subjects can be asked to use the proposed framework for modeling and analyzing a number of case studies. The modelers then need to be interviewed or quizzed and models need to be critically analyzed to draw conclusions about the practical usefulness and expressiveness of the modeling approach and modelers’ perception and problem solving performance.

The empirical experiment we conducted in Section 4.6.1 was limited to evaluating the impacts of model perception on security analysis results, but it is even more important to evaluate whether the process of modeling would affect the problem solving performance. We intend to repeat the study using the models developed in the $i^*$ notation with security extensions. It would also be interesting to measure the level of security knowledge among subjects to investigate how well experts and novices perform with, and without, referring to the models. In order to generalize the conclusions of this kind of work, the scale and complexity of the scenario can be increased to observe whether the visualization of large-scale scenarios helps or confuses analysts. Finally, this study is limited to the comparison of goal-oriented models and natural language. Complementary studies are needed to compare the cognitive process and impacts of adopting other types of enterprise modeling methods.

In the current framework, identifying relevant vulnerabilities on the basis of a requirements view is carried out by the human analysts. In order to make the tool more practical, it is helpful to automate the search for vulnerabilities in the CWE and/or OWASP repositories, in such a way that the tool searches vulnerability web portals for relevant vulnerabilities, based on the natural language phrases used in the requirements view of the $i^*$ models. This search feature
should to be integrated with the modeling tool.

**Risk Assessment Component:** In future, we plan to provide a tool for collecting analysts’ evaluations of vulnerability metrics to automatically feed them to the decision analysis tool. In addition to streamlining the risk assessment and decision analysis process, current CVSS metrics can be expanded and refined as well. To this end, we need to analyze more scenario examples using the CVSS, aiming to identify vulnerabilities whose criticality cannot be specified and evaluated by using the current CVSS metrics. We have identified vulnerabilities related to un-trusted computing environments as an example of security vulnerabilities for which CVSS metrics are irrelevant. We substituted some of the CVSS metrics with new metrics suited for un-trusted computing environments. In future, we need to expand and refine these new metrics as well.

To evaluate the utility of the proposed risk assessment method, we analyzed two of the top OWASP vulnerabilities and we showed that our risk assessment method also evaluates the risk of those vulnerabilities as high damage and high probability. In future work, we need to test more varieties of vulnerabilities with the proposed risk assessment method. To provide further validation security experts can also be involved, to study their reasoning strategy for assessing the risk of vulnerabilities. Ultimately, it is essential to compare the reasoning strategies of experts with the metrics being used in CVSS to evaluate whether our method reflects the way experts analyze risks and vulnerabilities.

**Decision Analysis Component:** In the previous section, we discussed several limitations of the current comparison-based decision analysis algorithm and the roots of those weaknesses. To tackle those problems, in future work, we aim to reduce the number of possible consequences that the algorithm generates by introducing the concept of thresholds for critical criteria. Thresholds are the minimum satisfaction level for important criteria that alternatives must meet, from the stakeholders’ point of view. These thresholds can be used to trim the number of cases that the algorithm needs to analyze. In this way, for cases where an alternative is under the specified threshold, the alternative is automatically dominated. We are also exploring ways to reduce the number of possible consequence cases by adding numerical data for criteria or alternatives whenever such data is available. This approach can reduce the number
of exceptional cases (and patterns) that we need to provide to stakeholders and experts for further evaluation.

In the limitations of the method, we discussed that our notion of swap reusability limits the performance of our tool. To improve the algorithm, in future work, we will analyze and add two new rules for suggesting swaps: first, improving our tool to identify relative rankings of criteria according to the swaps that stakeholders make. The knowledge of criteria ranking can direct the tool to remove unimportant criteria earlier in the process. The ranking of goals’ preferences can be identified from the swaps between two goals such as \( g_x \) and \( g_y \) where the consequence of a solution such as \( A \) on these goals are \( a_x \) and \( a_y \) respectively:

For a swap like \((g_x : a_x \rightarrow a'_x \iff g_y : a_y \rightarrow a'_y)\),

- if \( |a_y - a'_y| < |a'_x - a_x| \) then \( g_y \) is preferred to \( g_x \) (\( g_y > g_x \))

- if \( |a_y - a'_y| > |a'_x - a_x| \) then \( g_x \) is preferred to \( g_y \) (\( g_x > g_y \))

When the units of increase on \( g_x \) are more than the units of decrease on \( g_y \), the stakeholder is indicating to prefer \( g_x \) to \( g_y \). Based on the knowledge of relative rankings, the tool can target removing the least important goals earlier; the result of which is, the ultimate decision is made based on the goals that are of the most importance for the stakeholders.

Secondly, we aim to expand the swap reusability by swap generalization. To explain swap generalization, consider a swap such as \((g_x : a_x \rightarrow a'_x \iff g_y : a_y \rightarrow a'_y)\), where, \( a'_x > a_x \), \( d_x = |a'_x - a_x| \) and \( d_y = |a'_y - a_y| \). This swap indicates that the stakeholder is willing to change the consequence of solution \( A \) on \( g_x \) from \( a_x \) to \( a'_x \) (\( d_x \) level), and compensate it by changing the consequence of \( A \) on \( g_y \) from \( a_y \) to \( a'_y \) (\( d_y \) level). If we are given another solution such as \( B \), where \( b_x = a_x \), and we ask stakeholders to increase \( b_x \) to \( b'_x \) where \( b'_x = a'_x \), \( a_y > b_y \), then the stakeholder may not be willing to reduce \( b_y \) by \( d_y \) levels, because the satisfaction level of \( g_y \) is lower than the previous swap. This is a case where the swap is not reusable. On the other hand, if \( b_y > a_y \), the stakeholder has previously agreed to sacrifice on \( g_y \) by \( d_y \) levels, so it is rational to assume the stakeholder agrees to reduce \( g_y \) from \( b_y \) at least by \( d_y \) levels. This is called swap generalization. In future work, we aim to empirically test whether stakeholders agree with generalizing swaps in such a way. Swap generalization can potentially reuse swaps for half of the possible consequences and is a significant improvement of the algorithm. A
A key problem of the Even Swaps method is the potentially-inconsistent swaps made by decision stakeholders. This is a common problem that preference elicitation methods encounter [92]. The Smart-Swaps tool [116] provides a mechanism to identify inconsistencies in swap statements made by the stakeholders by comparing them with previously made swaps. The consistency validation approach in Smart-Swaps is based on eliciting weight ratios between the decision criteria according to the swaps stakeholders make. However, we discussed that weight ratios between the criteria are not static (discussions under the notion of swap reusability in Chapter 6). Thus, even if stakeholders make swaps that bear inconsistencies among weight ratios between criteria, it is not considered an inconsistency in our method. To identify relevant inconsistencies, we leverage on the concept of relative ranking of criteria, which we discussed earlier as the basis of a new rule for suggesting swaps. By extracting relative rankings of criteria, we can also detect inconsistent swaps, where the swap does not match the ranking that was elicited previously.

In summary, the decision analysis methods proposed in thesis can be further improved by conducting more case studies and applying the methods in action research and experiments. User studies would be helpful to improve the features and interface of the tool.
Bibliography


[121] OWASP. http://www.owasp.org/.


