UNIVERSITÀ DEGLI STUDI DI TRENTO

INTERNATIONAL DOCTORATE SCHOOL IN INFORMATION AND COMMUNICATION TECHNOLOGIES

XX CYCLE – 2009

Requirements Analysis and Risk Assessment for Critical Information Systems

Yudistira Asnar


UNIVERSITÀ DEGLI STUDI DI TRENTO

INTERNATIONAL DOCTORATE SCHOOL IN INFORMATION AND COMMUNICATION TECHNOLOGIES

XX CICLO - 2009

Yudistira Asnar

Requirements Analysis and Risk Assessment for Critical Information Systems

Prof. Paolo Giorgini (Advisor)
Prof. John Mylopoulos (Co-Advisor)

Thesis Committee

Prof. Fabio Massacci
Prof. Bashar Nuseibeh
Prof. Ketil Stølen

AUTHOR'S ADDRESS:
Yudistira Asnar
Dipartimento di Ingineeria e Scienza dell'Informazione
Università degli Studi di Trento
via Sommarive 14, I-38050 Povo di Trento, Italy
E-MAIL: yudis.asnar@disi.unitn.it
WWW: http://yudis.asnar.net

Abstract

Critical Information Systems (CISs) are a special class of information systems where their failures might produce catastrophic effects (e.g., life loss, economic loss, the environment destruction). A lot of efforts have been devoted in literature to improve the quality of CISs along the system development process. However, a major limitation of current approaches is that they consider the system only from the technical perspective and, very often, overlook the social aspects of the environment where the system will operate. Many incidents are indeed caused by factors beyond technical failures, such as abuses of permission or multi-actor (i.e., social) failures (e.g., mistrust, commitment repudiation). Considering interactions between humans and technology allows us to identify a wide range of risks in addition to those emerging from both aspects in isolation.

This dissertation focuses on risk assessment in requirements analysis. We propose the Tropos Goal-Risk (GR) framework comprised of a modelling framework, analysis techniques, a methodology, and a supporting tool. The GR modelling framework is built on the following basic concepts: *requirement* that the system must fulfil, *event* that can prevent the fulfilment of requirements, and *treatment* that protects the system from harms. The framework is equipped with a set of techniques to support the analysis and the evaluation of risks. These techniques are used to verify whether the risk level is acceptable and whether the cost is affordable. To develop a GR model, analysts are guided through a set of methodological steps where they capture stakeholders' needs, analyse risks, evaluate the impact of risks, and introduce necessary treatments. As a final result, analysts are able to elicit robust requirements to realise the stakeholders' needs that are feasible to be implemented in a given organisational-setting and have manageable risks. Moreover, such requirements incorporate necessary countermeasures to mitigate risks. All the methodological steps and analysis techniques are supported by a tool. This framework has been proved to be useful in assisting analysts to avoid eliciting risky requirements and also to reduce the number of requirement changes due to the revision of risky requirements or the introduction of countermeasures. Finally, to demonstrate the usability of our framework, we present a number of application of the framework (and its extensions) in several case studies.

Keywords

Risk Assessment, Requirement Analysis, Socio-Technical Systems, Critical Information System, Security and Dependability Analysis

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Software systems are increasingly essential to our lives. They play a central role in daily activities (e.g., online shopping, bank transfer) as well as in critical ones (e.g., air traffic management, nuclear power plant control). In critical contexts, system requirements need to comprise: 1) stakeholders' needs, 2) quality criteria to be satisfied, and 3) acceptable risks for systems' failures [17]. For instance, an Air Traffic Management (ATM) system must be able to control and to monitor air traffic and, besides this, it must fulfil the quality criteria of being 24/7 availability. Unfortunately, there are some situations where the systems fail in fulfilling such requirements. Hence, the stakeholders need to specify the acceptable risk, such as the system may fail twice per year with a maximum outage time of 10 minutes for each failure. Moreover, analysts should not only prevent failures, but also foresee the actions (e.g., fail-safe measures) to be taken in case of failures. In this way, both the system users and the environment would still be safe [179]. For example, in case of broken radars Air Traffic Controllers (ATCOs) need to manually maintain safety separation among aircraft.

In [192], Sommerville categorises critical systems into three classes: *safety-critical* systems where failures result in life loss, or environment damage (e.g., nuclear plant management system); *mission-critical* systems where failures obstruct goal-directed activities (e.g., spacecraft navigation system); and *business-critical* systems where failures cause economic loss (e.g., bank accounting system). For these systems, Security and Dependability (S&D) are considered as the most common desired quality attributes. Security is defined as the preservation of confidentiality, integrity, and availability of information under attacks [106], while dependability concerns about availability, reliability, integrity, safety, and maintainability of the system against failures [17]. Moreover, an S&D system should be able to deliver trusted services and prevent failures beyond an acceptable level.

Analysing requirements for CISs is a complex activity, since it must consider the entire a socio-technical setting where human agents are tightly coupled with technical systems. For instance, in [197] the author presents how elder cares are provided by means of pervasive computing technologies. In such a system, performances closely depend on the reliability of pervasive sensors and the perceptiveness of staffs in responding to changes in elderly people. Considering human activities in a CIS has the advantage of making the system more flexible and adaptable to external events (e.g., response on a cardiac-arrest of an elderly patient, mitigation to an incoming attack). However, they may also cause some failures as reported in [152] (e.g., staffs on strike, mis-entry the prescription). To assure better performances of a CIS, analysts should consider risk (i.e., uncertainty) from both the social and technical aspects of the system since early phases of the system development. In other words, analysing CISs as Socio-Technical Systems (STS) will results in a better quality of requirements because it considers not only technical issues but also the social aspects of the environment in which the system will operate. For example, an

employee strike, which is not a technical risk, may disrupt the availability of the system in its business context.

An STS is considered as a complex network of interrelationships between humans and technical systems that includes users, stakeholders, regulations, hardware, software, and data [170].[1] Considering both humans and technical aspects also allows analysts to realise the secure-by-design principle where the system is designed to be secure from the beginning and not treated as systems' patches. For instance, to reduce the risk of frauds analysts may design and introduce some procedures/processes (e.g., assessing loan, approving loan) to be performed by different actors (i.e., segregation of duty). Finally, considering the socio-technical dimension also prevents analysts from eliciting risky requirements, and consequently minimises possible requirements changes.

1.1 Problem Statement

As discussed above, in order to develop CISs from a socio-technical perspective, we should be able to assess risk during the analysis of stakeholders' needs/requirements. In addition, we have to detail how requirements can be operationalised; then identifying relevant risks and consequent countermeasures to be adopted in order to protect the fulfilment of requirements.

Traditionally, in software engineering, risk analysis relates to the risk associated to a software project and not to the software system [29], or at most risk is considered during the design phase of the software development lifecycle [214]. In other words, risks are identified and evaluated against the design and not against the earlier software development artefacts (e.g., business objectives, stakeholders' needs, software requirements). Such an approach can result in a situation where design modifications are required in order to adopt specific countermeasures that, in turn, might cause further problems, such as the inconsistency of the design or the contradiction of stakeholders' requirements. Considering risk analysis as an integral part of the requirements analysis can reduce the necessity of introducing design modifications and, possibly, the revision of stakeholders' requirements.

Recent works have tackled the problem of considering risk in the early phase of system development based on two most common approaches: i) focusing only on technical systems (e.g., UMLSec [116], Fault Tree Analysis (FTA) [100], Failure-Modes Effect and Criticality Analysis (FMECA) [205]); ii) general information system frameworks including the human components (e.g., ISO 27001 [105], CORAS [64], COSO-ERM [58], COBIT [200], ITIL [154], and some *i**-based frameworks [137, 141]). Moreover, most of risk analysis frameworks (e.g., CORAS, ISO 2700X, FTA, FMECA) concentrate in analysing risk from a single point of view. This approach is over-simplistic since an organisation is not a self-aware entity, which is subject to some risks. As illustrated by Holton in [94], an organisation has to be considered as a conduit pooling risks of all members of the organization itself. This implies that the organisation's reaction against risk should consider the subjective perception of each member and not limit the analysis, for example, to the business perspective of managers and leaders (e.g., CEO).

The problem of limiting to a single perspective is that some members in the organisation might have a different perception toward risk and then, as described in [220], act differently from what was defined in the organisation's strategies. For instance, Air Traffic Controllers (ATCOs) believe the system is at risk in guaranteeing safety operations of air traffic management, even if technical staffs argue that the countermeasures have significantly reduced risks. This situation might occur because the ATCOs are not aware with any countermeasures deployments, or in ATCOs' viewpoint the countermeasures are

[1] To avoid confusion, from now on, the term "system" refers to the notion of STS and for the technical aspects we refer as "technical system".

ineffective mitigation, or ATCOs simply do not trust technical staffs in operating the countermeasures.

Analysing risks from a multi-actor viewpoint allows analysts to assess the risk level perceived by each actor and, possibly, to assess risks originated from dependencies among different actors, especially at the absence of trust relations (e.g., delegation enforced by regulation/policy, the only way to fulfil the objective). For an example, we depend on a postal service for handling the shipment of our internet purchase as requested by the e-shop. If we trust the postal service, then we perceive the risk of mishandling is low, otherwise we might perceive a higher risk.

1.2 Research Contribution

The thesis presents an integrated framework, called the Tropos Goal-Risk (GR) Framework, that enables analysts to consider risks since the early phase (i.e., requirement analysis) of the system development. The framework is comprised of a conceptual modelling framework, analysis techniques, a methodology, and a supporting tool. Analysts are guided by a set of methodological steps towards the risk assessment during the requirements analysis phase. The analysis is based on GR models founded on the following basic concepts: *requirement* in which the system need to provide for the organisation, *event* that can prevent the fulfilment of the requirement, and *treatment* that protects the system from harms. Analysis techniques are used to verify properties of GR models (i.e., whether the risk level is acceptable or whether the cost is affordable). The tool is built to support the work of analysts in developing and analysing GR models. As final result, analysts elicit requirements that have a manageable risk and feasible to be implemented in a given organisational-setting. Note that the requirements also include countermeasures.

The GR modelling framework is developed to support risk analysis at the organisational level. It allows analysts to capture risks beyond the technical aspect of a software system. For instance, the risk of *an immediate increase of air traffic* cannot be considered as a technical failure, but it can however disrupt the business of an air traffic management service. Ideally, requirements should consider and possibly prevent such a risk. Another particularity of the framework is that it is able to support analysts in assessing risk as actual and perceived risk. In the actual risk assessment, analysts assess the risk level that an organisation is exposed to, while the perceived risk assessment aims at assessing the perception of each member of the organisation by considering its mental states (e.g., risk tolerance, expectations, etc.) and its trust relations towards other members. Notice that sometime there are discrepancies in the results from both assessments. For instance, from the actual risk assessment the ATM system is concluded to be very reliable, while ATCOs, however, still perceive a high risk. By comparing both results, analysts can identify such discrepancy and (if necessary) fix it so to prevent the degradation of the system's the performances. It is because such a situation can cause a stress to ATCOs since they may perceive a high risk, though it is not real; or in other settings ATCOs may behave carelessly because they believe the safeguards will take care of all their errors.

Note the fact that we consider risk analysis as part of the requirement analysis does not imply we don't perform risk analysis in later phases of system development (e.g., design, implementation, and deployment). As in [153], problems and solutions analysis are a weaving process because some problems can be only materialised after we commit to a particular solution. The same principle is applied here; one may discover new risks or refine identified risks into more precise forms as soon as the details about the system are specified.

1.3 Structure of the Thesis

This thesis is organised as follows:

Chapter 2 presents an overview of the State-of-the-Art (SoA) related to the risk analysis for critical information systems in a socio-technical context. It starts from revisiting the notion of system theory, and particularly in the area of Organisational Theory and Socio-Technical Systems (STS) Theory. Understanding of these theories is critical since it shifts the paradigm how we perceive and analyse IT systems. Moreover, the review of organisational theory is crucial to understand the behaviour of an organisation and its members in facing risk; the STS theory explains how social and technical aspects are interrelated to form a system and deliver its features. It then continues by reviewing the SoA in Software Engineering in dealing with risk during the engineering process, particularly requirement phase. It also examines the pre-eminent proposals in Security and Dependability Engineering since security and dependability are considered to be the main quality attributes for critical information systems. At last, the chapter presents a comprehensive review of the state-of-practices in IT Governance, which have been mainly used to develop IT systems (i.e., business-critical systems).

Chapter 3 details the problems and challenges addressed in this thesis. Though risk is considered as one of the most prominent problems that humans have faced [24], this concept is still yet unique and not well understood for the modern era. The complexity of information systems (i.e., socio-technical setting, information is intangible) makes this situation even harder. It also illustrates some particularities of risk in IT domain. It then continues to explain the approach taken by us to resolve the problems. Moreover, it explains the underlying works of this thesis, and finally details the research contribution of this thesis.

Chapter 4 introduces the Tropos Goal-Risk modelling framework. It starts from clarifying the notion of risk considered in this thesis. It then explains in detail the basic concepts of the modelling framework which are the adaptation from *i** [226] and SI* [230]. Essentially, those concepts are structured into three-layers conceptualisation (i.e., value, event, and treatment) following the idea of the Defect Detection and Prevention (DDP) [56]. Moreover, along with the explanation we present their graphical representation and their usages using the Air Traffic Management scenario. The chapter also highlights the benefits, in terms of expressiveness, that one can benefit from modelling requirements and risks using the Tropos Goal Risk modelling framework.

Chapter 5 details a set of analysis techniques for a Tropos Goal-Risk model. In fact, these techniques are important for analysts to verify some properties (e.g., risk level, cost) and to assist analysts in improving the model (i.e., so that the risks are acceptable). It starts from formalising the GR constructs in a formal framework. Essentially, the formalisation is developed adapting from the Dempster-Shaffer (DS) Theory of Evidence [182]. The DS theory is chosen because apparently it is better in dealing with subjective inputs even compared to the Bayesian Theory [188]. This feature is essential since often risk analysis in CISs development is conducted under incomplete knowledge and objective evidence. Basically, there are two main analysis techniques: *forward reasoning* adapted from [80], which computes the risk level based on a given inputs, and *backward reasoning* adapted from [181], which elicits a set of treatments to mitigate the risks for a given constraints (e.g., maximum cost). Both techniques are implemented in terms of automated reasoners.

Chapter 6 describes the Tropos Goal Risk methodology. It presents a set of methodological steps to assess risks along the requirement analysis. The process starts from identifying requirements to satisfy stakeholders' needs. It then continues by identifying and analysing risks that can obstruct the requirements. It then assesses the risk level, and if it is below the risk tolerance defined by the stakeholders then the modelling process is finished. Otherwise, analysts need to introduce some treatments to mitigate the risks. These steps continue until all risks are acceptable by the stakeholders. Moreover, the chapter also indicates some crucial points in each step of the methodology.

Chapter 7 - Chapter 9 present the application of the framework in several case studies from various domain applications. Chapter 7 demonstrates how to perform risk assessment in an organisation. Chapter 8 demonstrates the usage of proposed analysis methods in forming an organisational-setting. In this chapter, we assume that there is no such organisation-setting at the beginning. Analysts here are responsible to explore possible organisation-settings and come up with the one that is not risky. The chapter explains how the GR framework works incorporated with the planning technique [41] to elicit an organisation-setting that has the acceptable level of risk. Chapter 9 demonstrates the possibility of encoding the GR framework in the knowledge and deliberation process of an autonomous agent. This effort enables the agent to perform risk assessment and to (re)act avoiding/against unacceptable risks by selecting additional countermeasure.

Chapter 10 concludes the thesis, with a summary of the main contributions and a brief discussion of potential future research and extensions.

Chapter 2

State of the Art

This chapter reviews the state-of-art related to risk assessment in IT systems engineering. We start from revisiting the paradigm of socio-technical systems (STS) [202]. Essentially, this paradigm shifts the perspective from perceiving an organisation as a social aspect and a system as merely technological artefacts to analysing both aspects as a whole. Considering interaction between humans and technology in an organisation allows us to identify a wide range of risks in addition to those emerging from both aspects in isolation. We show how some classes of failures cannot be identified when social and technical aspects are considered separately (e.g., conflict of interests, repudiation of commitments) [34]. Then, we give an overview about risk assessment in Security and Dependability (S&D) engineering and software engineering. In addition, we review the state of practice on the area of IT governance, which is commonly used by practitioners to manage IT systems so that in-line with the business objectives of the organisation.

2.1 System Theory

In this section, we review the notion of socio-technical systems starting from its origin, System Theory [35, 216]. We then review in-depth the two specific developments of the system theory: organisation theory [169] and socio-technical theory [202]. These studies helped us to understand the nature of an organisation and how an organisation should perceive technical systems.

System theory is originated from the work of biologists in trying to understand how organisms function, live, and interrelated forming an ecosystem [18]. System theory, particularly General System Theory (GST), studies the phenomena in an organisation, such as independence among the components, type, and spatial-temporal scale of the existence of some entities [35, 217]. It also aims to identify the common principles of complex entities (e.g., enterprises, ecosystems, colonies), and to develop the models, usually mathematical ones, which can be used to describe them. In [217], von Bertalanffy argues that scientific observations are often subject to the isomorphic law where not only general aspects, structures, and viewpoints are alike, but some observations may share the same mathematical model. For instance, John Nash discovered Nash's Equilibrium theory that was inspired by the movement of birds. This theory later gained an acceptance in economy because it explains some phenomena in stocks trading. GST aims at investigating the formulation and deduction of basic principles, which are valid for systems in general. System, in GST, is defined as complex interacting component where each interaction can be seen as a particular type of relations among components. In other words, a component can have different behaviours depending on the relation in which it is involved.

Socio-Technical System is a paradigm for analysing a system as a whole [202]. This paradigm emerges due to the odd phenomena in the coal mining where the productivity failed to increase following the increase of mechanisation. Trist, and his postgraduate students, discovered a new paradigm for designing the works/responsibilities of each component in the system [202]. This paradigm investigates the requirements and the relations between social and technical aspects of the system. Moreover, to improve the design one should start from analysing the primary objectives of the system, and then designing the responsibilities for each individual with considering their respective motivations. One study [202] suggests us to organise individuals into several groups, and to assign them some responsibilities and let them to have a significant degree of control. Essentially, this principle is in-line with the principle of self-regulation in cybernetics [18]. Moreover, Trist, in [202], presents a set of new design principle which is best suited for designing an STS: i) joint optimisation instead of imperative based, ii) innovation instead of risk aversion, and iii) man as complementary to the machine instead of man as an extension of the machine. Finally, the development of a system should be brought at the level of macro-social (i.e., larger than a single organisation) since the system affects entities beyond the organisation. The initiatives to have community-based socio-technology endeavours (e.g., eco-green technologies) and networks (e.g., networks of excellence) are some of the examples of bringing the system in macro-social situations.

Organisational Theory/Behaviour emerges as a discipline that studies the impact of individuals, groups, and structures on behaviours within an organisation. In fact, an organisation is one form of a system (i.e., a coordinated social unit) that is composed of two or more individuals who intend to achieve common goals by means of relatively continuous interactions [169]. This discipline also aims at gathering knowledge and applying it towards the improvement of productivity (in terms of effectiveness and efficiency) in an organisation. As such, the organisation can achieve the goals by transforming inputs to outputs at the lowest cost. Essentially, organisational behaviours are shaped by the behaviours at group and individual level that interact one to another following defined constraints (e.g., policies). Though groups are composed by individuals, the group behaviour might be completely different from the behaviour of each constituent as illustrated in *the Wisdom of Crowds* [198]. In [169], Robbins presents the model depicting how the behaviour of an organisation is formed and influenced by its surroundings (e.g., individual behaviour, group behaviour, organisation design-structure, power & politics).

2.2 Security and Dependability Engineering

Security and Dependability are considered as the most important properties of critical systems (e.g., power plant, air traffic management, bank system) [192]. Dependability essentially is a system quality/ property concerning availability, reliability, safety, and maintainability [17], while (information) security is defined as a quality of being able to preserve the confidentiality, integrity and availability of the (information) systems under attacks. In addition, other properties, such as authenticity, accountability, non-repudiation, and reliability are often also considered [106]. In this section, we review prominent works that have been proposed in the area of security and dependability engineering. Particularly, how they address issues related to risks.

2.2.1 Failure Modes, Effects, and Criticality Analysis

Failure Modes, Effects, and Criticality Analysis (FMECA) (or often called FMEA) [205] was developed in the 50's by reliability engineers to determine problems that could arise from the failure of military

systems. It is also used to identify and assess potential failures, their effects to the system, and countermeasures to mitigate the failures. FMEA considers how each component fault (called also failure mode) can result in system failures and ensures that the appropriate countermeasures are in place. Essentially, FMECA is the extension of FMEA with an addition of defining criticality of failure modes in terms of quantitative measures instead of qualitative ones. Steps of FMECA can be considered as follow:

1. define the system - it defines goals, functionalities, and boundaries of the system. It helps to define the failure modes that can occur in the system;

2. define problem of interests of the system - it defines categories of problem that could arise in the system. For instance, the categories are reliability problems (e.g., getting wrong position of the aircraft), safety problems (e.g., getting incorrect air pressure). These categories limit the analyst from exploring irrelevant failure modes;

3. decompose the system into smaller elements - this task can be done following two approaches: component based or functional based. The component based is more suitable when the component of the system can be uniquely identified (i.e., usually it is applied to analyse the infrastructure of the system). The functional based approach is normally used when the components cannot be uniquely identified (e.g., organisational structure) and many components have overlapped functions in the system;

4. identify potential failure modes of each component of the system. As an example, radar systems can produce the wrong position of aircraft;

5. evaluate each potential failure mode against the problem of interest of the system. There are several aspects that are critical during the evaluation, such as effects, causes, likelihood and severity of the failure mode, and the probability of countermeasures fail in mitigating the failure mode.

In FMECA, each failure mode/risk has an attribute, namely Risk Priority Number (RPN) that is calculated on the basis of *probability* of risk, *severity/effect* of risk, and *detectability* of risk. RPN is used to prioritise which failures modes/risks are critical and need more treatments to mitigate them. Although FMECA is very effective to analyse various failure modes, this technique suffers from several limitations. FMECA is concentrated more on the failure modes that are originated from infrastructure faults. Human error analysis typically is performed as an extent when they can lead to the infrastructure faults. Deviations from normal operations, that do not produce infrastructure failures, are often overlooked. FMECA analyses component faults one by one, and sometimes overlooks in analysing the combination of components faults. Moreover, it could not model the system failure that is caused by external events (e.g., natural disaster). FMECA just supports one view of the service failures. However, in reality each service failure has different consequence depending on the operational mode. For instance, in Air Traffic Management (ATM) the failure of "Radar By Pass System" (RBP) would not lead to severe situation because the ATM typically runs in a redundant "Radar Data Processing" (RDP) mode. However, this assumption is not valid anymore when all RDPs have failed; the failure of RBP can result in catastrophic consequences.

2.2.2 Fault-Tree Analysis

Fault Tree Analysis (FTA) [100, 196] is one of the most commonly-used techniques in reliability engineering. It determines various combinations of hardware and software failures, and human errors that

could lead to the occurrence of undesired events/failures (called top events) at the system level. The modelling starts from identifying system failures. Each failure is analysed on its own fault tree and refined into more detail events and end up with un-develop/external events (called leaf events). FTA visually models logical relationships among events (e.g., infrastructure failures, human errors, and external events) that could lead to the system failure. The logical relationship is represented by several types of logical gates [100], such as AND, OR, XOR, NOT gates. Additionally, there are some relationships (e.g., redundancy and inhibit) that also govern the occurrence of an event that can affect the system.

Through this model, analysts can obtain minimal leaf events that could lead to the occurrence of system failures (called minimal cut-set) which is useful to ensure the safety of the system. Moreover, the model can show how to prevent the failures from happening by negating the minimal cut-set (called minimal path-set). Minimum path-set can be seen as a means to prevent the occurrence of failures by preventing the minimal leaf events to occur. The main goal of FTA is to assess the likelihood of the system failures/top events based on the likelihood of leaf events. The results can be used as an input to improve the system, in terms of safety and reliability. One method of doing it has been presented by Bedford and Cooke, in [21], using the probability calculus as underlying theory. Notice that this approach has also been adapted for analysing faults in software systems [69, 74, 91, 92], analysing threats [92] or attacks [178].

Although FTA has demonstrated its effectiveness to model how combinations of events/component failures/human errors can lead to the system failure. It has notable limitations: narrow focus (i.e., each FTA examines only one specific accident of interest, to analyse other types of accidents, other fault trees must be developed); and the detail level of the tree is closely related to the analysts (i.e., there is no strict guideline defining whether an event is already detailed enough or not).

2.2.3 Probabilistic/Quantitative Risk Analysis

There have been several frameworks to analyse risks quantitatively, for instance Probabilistic Risk Analysis (PRA) [21] from reliability engineering area, and Quantitative Risk Analysis (QRA) [218] in the economic area. Essentially, both works uses the Kolmogorov probability theory [171] as underlying mathematical theory.

In both approaches, risks are defined as (negative) uncertain event (e.g., threats and failures) quantified with two attributes: *likelihood* and *severity*. Risks are prioritised using the notion of "expectancy loss" which is the product of its likelihood and severity. Priority here reflects the criticality of an event. When resources are limited, analysts may decide to adopt countermeasures for mitigating events according to their criticality. Alternatively, to rate the risks (i.e., threats), Microsoft refines both parameters into finer parameters: Damage potential, Reproducibility of threats, Exploitability of threats, Affected users, and Discoverability (DREAD) [145]. Moreover, Vose, in [218], argues it is critical to distinguish between variability (i.e., resulted from random variables) and uncertainty (i.e., resulted from the lack of knowledge) during risk analysis for a better assessment.

Both approaches suggest us the use of objective data and avoid as much as possible the subjective one. The presence of subjective data (e.g., experts' judgement) can result in the assessment being more imprecise and prone to subjectivity. However, to mitigate these issues we can adopt some techniques to gather "good" opinions from experts (e.g., uncertain judgements [155], Bayesian combinations [21], experts' probability distribution combination [53, 218]). Moreover, even when analysis is done using objective data, it still does not guarantee that the assessment is free from uncertainty. In fact, both works suggest several techniques to quantify the uncertainty in the assessment (e.g., statistical methods, Bayesian method, bootstraps).

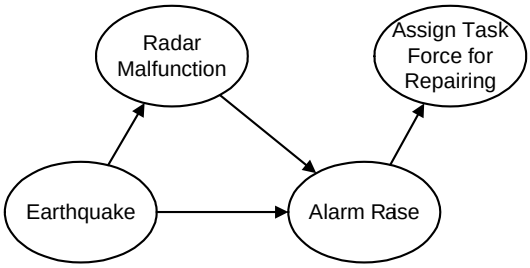


Figure 2.1: Bayesian Belief Network

2.2.4 Bayesian Belief Networks

Bayesian Belief Networks (BBN) [22, 160] actually, is not a model that is specifically built for risk/ reliability modelling. It is a mathematical model to depict interrelation of several events by defining the conditional probability between events. It has been well used in artificial intelligence area to reason under uncertainty. Basically, BBN is a Directed Acyclic Graph (DAG) where vertex represents an event that can occur, and edge represents the relation of conditional probability between events. This model helps to define the joint probability distribution of several events. For example, there may occur: radar malfunction, earthquake, raised alarm, assigned task force for repairing. Earthquake is the root event because there is no other event that influences the occurrence of earthquake. The earthquake occurrence probably will cause radar malfunction and raise the alarm. Radar malfunction can be caused by other circumstances besides earthquake, and it will also raise the alarm. Once the alarm is raised, a task force is assigned to repair the malfunction. All these relationships are depicted in Figure 2.1. For a given model, analysts can compute the probability of an event (depicted as vertex) as demonstrated in various frameworks [75, 76, 147].

An event that gives direct contribution to another event is called parent event (i.e., Radar Malfunction and Earthquake are considered as parent events for the event of alarm raised). Each event has complete values of the conditional probability with respect to the occurrence of its parents. For instance, probability of radar malfunctions when the earthquake occurs. Based on those data, it is possible to calculate the probability of any circumstance involving several events that are categorised as a failure (e.g., alarm is raised when there is no earthquake nor radar malfunction, there is no assigned task force when the alarm is raised). This model is very helpful to quantify join distribution of several events. However, some frequentists argue that the probability value should be obtained from long-observation of random process. They also deny the correctness of the Bayes theorem, because Bayes allows us to represent probability as a degree of believability (i.e., not based on the long-observation). As we notice BBN represents the model as a DAG which cannot capture the situation where there is a vicious circle among the events, such as event A triggers event B and event B triggers event C and event C triggers event A.

In the area of dependability engineering, this model is used to combine the probabilities gathered from experts' judgement [21]. Moreover, BBN can be used to assess the safety level of a safety case [224]. In [95], the authors demonstrate how benevolent a BBN is in assisting the development of a secure system, by computing risks, and finally how to derive the Return of Security Investment from it.

2.2.5 CORAS

CORAS [85] is a research and technological development project under Information Society Technologies (IST) Program that is aiming at developing a framework for risk analysis of security critical systems.

It is developed following standards (e.g., AS/NZS 4360:1999 [16] and ISO 17799:2000 [103]). The CORAS risk management process consists of the following steps:

1. context identification, by selecting usage scenarios of the system;
2. risk identification, which tries to identify the threats based on scenarios and the vulnerabilities of these assets;
3. risk analysis, which assigns the values (impact and likelihood of occurrence) to identified threats;
4. risk evaluation, which identifies the risk level of each threat;
5. risk treatment, which addresses the treatment of considered threats.

These steps are done in three different levels of organisations: decision maker (e.g., CEO, shareholders) where strategic objectives are set, entry level (e.g., general managers) where the analysis is done in high-level abstraction, and full level (e.g., plant manager) where risks are analysed in-depth. Along these steps, analysts must monitor, review, and communicate their findings across the organisation.

Moreover, CORAS is open to work together with related frameworks (e.g., FMECA, FTA, Hazard and Operability (HAZOP) [122]). The guidelines for choosing and integrating these frameworks into CORAS steps has been presented in [55, 85]. The methodology has already been tested in some security systems (E-Commerce and Telemedicine). CORAS also proposed a language to perform risk assessment and implemented as UML profile [157].

2.3 Software Engineering

We review the most relevant works in the area value-based software engineering (VBSE) [27] and agent-oriented software engineering (AOSE) [39]. These two areas of software engineering are particularly relevant for CISs and STSs in that they are able to capture critical and social aspects of a software system. VBSE proposes the notion of value as the main driver directing the overall process of engineering. AOSE studies how to develop a software system by modelling the system as an agent society where each agent interacts one to another providing the system features. We additionally review the two major approaches in requirement engineering (i.e., goal-oriented and domain-oriented), and their extensions for handling security concerns. Last, but not least, we conclude reviewing some extensions of UML [178], as a software modelling language, to deal with security.

2.3.1 Value-Based Software Engineering

Value-Based Software Engineering (VBSE) [27] emerges as a response to the fact that much research and practice in software engineering treats every requirement or use case equally important. In reality, some requirements are critical for organisations and the others are considered as secondary requirements. VBSE aims at adapting the value perspective into existing or emerging principles and techniques in software engineering. They span from requirement engineering, architecture design, design-development, risk management, quality management, and people management. This approach is build over seven basic principles [28, 30]:

- Benefit Realisation Analysis - determines and coordinates other initiatives, apart from IT development ones, required for the organisation to realise the potential benefits of IT systems;

- Stakeholder Value Proposition Elicitation and Reconciliation - identifies value propositions that are interested by stakeholders and, possibly, reconcile them in case they are in conflict or there are limited resources to realise them all;
- Business Case Analysis - determines the business value of IT systems in terms of costs, benefits, and their Return of Investment (ROI) across the organisation;
- Continuous Risk and Opportunity Management - manages Risks and Opportunity of IT systems across its lifecycle;
- Concurrent System and Software Engineering - defines a process model for engineering the system and software that enables the engineers to engineer software or services concurrently and continuously;
- Value-Based Monitoring and Control - ensures a software project results in a product that has earn-value according to the organisation expectation;
- Change as Opportunity - aims at engineering the change to create some opportunities. Often, organisations perceive changes as additional costs of rework due to the flaw in requirements. However, it is not completely valid because some changes create new opportunities that are impossible before. For instance, the emergence of web-service leads some organisations to change its IT architecture, and it creates an opportunity for organisations to assemble new services easily to meet the new trend in the marketplace.

In [27, 83], there are several software engineering practices that have been adapted to this paradigm (e.g., stakeholders' value elicitation, measurement and decision making technique, requirement analysis driven by risks, requirement selection, software testing) and also their applications in some developments of software system.

2.3.2 Agent-Oriented Software Engineering

Agent-Oriented Software Engineering (AOSE) aims at developing a software system, called Multi-Agent System (MAS), by modelling the system as an agent society. The system features manifest from the interaction among agents in the society. Several AOSE methodologies have been proposed in the literature. In this section, we will review some of them, namely GAIA [229] and OperA [66] that put emphasis in designing an agent society/organisation. However, one considers Tropos and KAOS in this category, but in this thesis we consider them as requirement engineering frameworks. Notice that most of the works in AOSE have not addresses the issues related to uncertainty and risks in particular. However, this field has developed several frameworks that model and analyse systems as societies of social and technical agents.

GAIA

GAIA [229] is considered as the first AOSE methodology that explicitly uses the analogy of human-based organisation. Organisations are modelled as a collection of interacting roles, which are defined in terms of responsibilities, permissions, activities, and protocols. Responsibilities captures expected behaviours of a role, and permissions are the entitlement to perform its responsibilities. Activities are units of action that a role can perform (i.e., without involving another role), and protocols define the interaction between roles or actions that involve other roles.

As in human organisation, an agent can play several roles at the same time. The notion of a role here gives an agent a well-defined position in the organisation, with an associated set of expected behaviours and their permission. To accomplish their roles, agents typically need to be able to perform some actions/activities or to interact with each other to exchange knowledge and to coordinate their activities. These interactions occur according to patterns and protocols imposed by the role itself. Additionally, a MAS is required to specify the environment where the agents interact to accomplish their roles. To specify the environment, one needs to determine all entities and resources that the MAS can exploit, control, or consume when all agents work achieving the organisational goals.

Though role and interaction models can be useful to describe an existing organisation, they lack in defining relationship between constructs (e.g., general relationships between roles, between protocols, or between roles and protocols). This motivates the introduction of the notions of organisational rules and organisational structures. In fact, the analysis of MAS should identify the constraints that an actual organisation needs to respect (i.e., norms). A role model implicitly defines the topology of the interaction patterns and the control regime of the organisation's activities. All of these are meant to define the overall architecture of the MAS organisation (i.e., organisational structure).

OperA

OperA methodology [66] aims at designing a model for organisations that support dynamic and autonomous interaction. The model is useful to understand agent societies in an organisation, since it uses organisational concepts, collectivist view of the environment, and uses the agent paradigm to provide a way to view and characterise intelligent systems. The main focus of this work is designing the environment where agents interact. Ideally, the environment has a balance that can be found between the autonomy of agents, their coordination needs, and the environment expectations.

OperA proposes of a 3-layers model to capture roles, goals, and interactions within the organisation:

- Organisational Model - describes a social system from the organisation perspective. It describes the aims and concerns of the organisation with respect to the social system. All of these are captured in terms of roles, interaction scripts, and social norms;
- Social Model - defines the activity of independent agents. In other words, it details the organisational model with specific agents mapped to roles through a social contract. Social contracts represent explicitly expectations on the behaviour of agents within the society;
- Interaction Model - describes activities of an agent society. They are described as interaction contracts, which are the agreements between role enacting agents (specified in the social model) concerning the enactment of interaction scenes (specified in the organisational model).

This framework is provided with a formal framework representing the contract, based on deontic temporal logic. It allows a realistic representation of the domain (in all models) formally and to verify them.

2.3.3 Goal-Oriented Requirement Engineering

In this area, we mainly consider the works that are originated in KAOS [61] and *i**/Tropos [38, 226]. However, the details of *i**/Tropos will be presented in Chapter 3 since it is one of the underlying work of this thesis.

KAOS

KAOS, a goal-oriented requirements engineering methodology, has been proposed aiming at modelling not only *what* and *how* aspects of requirements, but also *why*, *who*, and *when* [61]. KAOS has already been extended beyond goals, by defining *constraints* and *obstacles* that can be seen as boundaries in requirement analysis [210]. A constraint is formulated in terms of objects and actions that are available to the agent, and an obstacle is an undesirable behaviour. In addition, KAOS introduced the notion of anti-goals as goals associated with malicious stakeholders, such as an attacker [209]. In other words, obstacles can be seen as unintended risks, while anti-goals are threats or intended risks. These features make KAOS suitable for analysing requirements for secure and dependable systems, since security deals mainly with malicious intent (captured by anti-goals) and dependability addresses random events that can result in failures (captured by obstacles). In [210], the authors present a collection of techniques for deriving obstacles systematically from goals and domain properties, and also resolving the conflicts between obstacles and goals. The methodological steps of developing a KAOS model are described as follow:

1. acquire goal structure through AND/OR Graph of *goals* and identify the concerned *objects* of the goals;
2. identify potential *agents* and their *capabilities* by defining *actions*;
3. identify *obstacles* to the *goals*;
4. define the *obstacles* resolution;
5. operationalise *goals* into *constraints* and residual *obstacles*;
6. refine *objects* and *actions* for *goal* fulfilment;
7. derive strengthened *actions* and *objects* to ensure *constraints*;
8. identify alternative responsibilities;
9. assign *actions* to responsible *agents*.

KAOS uses temporal logic to specify concepts, like agents, actions, obstacles, and constraints. This facilitates formal reasoning to verify and validate models.

Obstacle and anti-goal can be considered as risk, since both are undesirable behaviours. In [210], they demonstrate how to derive the obstacles from the goal structure and to ensure the completeness of obstacles with respect to the goal structure and its domain. This technique assumes a complete knowledge to define the domain of the goal, which is hardly to meet in the reality. The methodology, also, introduces the guidance for defining obstacle resolution to mitigate and, possibly, eliminate obstacle. Though we can represent a risk as an obstacle/anti-goal, two mandatory properties of risk (i.e., likelihood and severity) are still missing. However, the obstacle concept and all related analysis (obstacle identification, obstacle resolution) are useful to enrich the notion of risk in our framework. Moreover, Boness et al., in [31], proposed a framework to assess risk a given requirement model (i.e., a goal graph). The risk of each goal is quantified in four factors: the environment assumptions, the achieve-ability of the implementation of the requirement, the integrity of the refinement, and the stakeholders' mandate. The framework results in the risk profile of each requirement and depicts which requirements that are safe or can be proceeded with caution or should not be proceeded. However, the correctness of this framework closely depends on

the integrity of the assessors, and moreover, it does not provide the guidance to resolve a situation where a requirement is too risky and should not be proceeded.

*i**/Tropos

*i**/Tropos [38, 226] is requirement engineering methodology that models and analyses requirements both the system-to-be and its organisational environment. *i**/Tropos uses the concepts of actor, goal, task, resource, and social relationships to capture stakeholders' intentions in an organisation. To address security issues, several approaches have been taken by several researchers. For instance, Mayer et al., in [141], extends the *i** modelling framework to analyse risk and security issues during requirement analysis. The framework models business assets (including business goals) of an organisation and assets of its IT systems (such as architectures and code). Countermeasures are then selected to mitigate risks, thereby ensuring that risks will not affect any assets severely. Liu et al., in [137], propose a methodological framework for security requirements analysis founded on *i**[226] and the Non-Functional Requirement (NFR) framework [52]. In particular, their analysis explores alternative designs and evaluates them on the basis of threats, vulnerabilities, and countermeasures.

Unlike both proposals, Giorgini et al., in [82], propose Secure Tropos introducing new additional concepts that are specific for security: permission and trust. In this way, a model can capture not only the objective and capability of actors, but it also the entitlement of an actor and the ownership of artefacts (e.g., goal, task, resource). Secure Tropos supports the notion of delegation to transfer objectives, entitlements, and responsibilities from an actor to another. Trust is a relation between two actors representing the expectation of one actor (the trustor) about the capabilities and behaviour of the other (the trustee). In fact, trust is the mental counterpart of delegation/dependency therefore trust is typically necessary for delegation. However, there could be the case where a delegation is done without trust, hence it may carry risk. Secure Tropos introduces the concept of monitoring as replacement of trust. In other word, an actor (the monitor) is appointed by the delegator to monitor whether the delegatee will not misuse the artefacts and fulfil assigned obligations. Delegation, trust, and monitoring have been refined by distinguishing between relations involving permission over the entitlement of artefacts and relations involving execution (obligations) to fulfil particular artefacts [81]. Moreover, the authors, in [148], propose a refinement the notion of dependency with some "security constraints" that should be fulfilled (i.e., either by the delegatee or the delegater) to ensure the security of the system.

2.3.4 Domain Oriented Requirement Analysis

In requirement engineering, domain oriented is another approach for eliciting requirements [110, 199]. Unlike goal oriented, this approach aims to understand problems in the domain (i.e., world) and structure the knowledge resulting from the analysis. The results are then used by analysts to develop the software system. The emphasis here is more to the analysis of the software system and not the system at large (i.e., a socio-technical system). However, there is a similar approach proposed by Lee et al., in [132], that attempts to analyse the system at-large by means of a problem domain ontology based on US-DoD standard on DITSCAP [204].

The Domain Theory

Sutcliffe and Sutcliffe, in their book [199], propose a framework, namely the domain theory, to organise knowledge resulted from software analysis so that it can be reused. The basic motivation of reuse is to

save time and efforts. The authors here argue that reusing the knowledge is better because it has been verified and validated by previous experiences.

The domain theory consists of three main components:

- meta-schema or modelling language, defines the semantics for generic models of application classes;
- generic models, state the existing problems in the world. They are organised in a class hierarchy;
- a computational matching process, retrieves best-suited generic models for a given set of facts about a new application.

To be able to cope with a wide-range of applications, generic models are organised into three ontology-based models: *grounded domains* as an abstraction of real-world problems that are modelled as interacting objects to achieve a goal, *metadomains* that capture human activities composed by generic tasks and operate on grounded domains and task, and finally *generic tasks* performed by agents that change the state of an object thereby it achieves a goal.

Such a structure allows analysts organising the results of analysis, in terms of domain knowledge, into modular models that, hopefully, can be reused in the future. For instance, there are many similarities in a hiring application despite the fact whether it is a library, a car rental, or a video rental. In such setting, analysts can reuse the (nearly) same grounded domains for those application domains. Unfortunately, in this framework uncertainty, risk to be precise, has not been addressed as important construct that one should organise for future reuse. In our view, since the current state of practice of risk identification is more an art then an engineering process, then we consider the needs to have means to capture such knowledge in the grounded domains for risks related to domain application, in the metadomains for risks related to organisational setting, and in generic tasks for risks related to tasks. In fact, risk management has adopted such a similar approach by employing risk register [159] to record all identified risks, but it is still too project specific.

Problem Frames

The Problem Frames framework analyses the system by assuming there is a problem in the world (depicted as several interacting domains) that needs to be solved by a machine [110]. Hence, analysts should elicit the requirements of the machine represented in terms of the interactions (i.e., shared phenomena) between the machine and other domains in the world and the controller of the shared phenomena.

In the Problem Frames, knowledge are organised into four concepts: (controlled) *domain* where the problems are originated, *required behaviour* that a machine should do, control *machine* that should be built to solve the problem, and *interface* indicating which phenomena are shared among domains and the machine. [110], Jackson believes most of the problems can be projected into five basics problem frames:

- required behaviour frame, machine performs a specific required behaviour to govern the controlled domain;
- commanded behaviour frame, machine acts following the command issued by operator domain and the action will govern the controlled domain;
- information display frame, machine presents the condition of real world domain into the display;
- simple workpiece frame, machine performs a modification to a designed domain commanded by the user domain;

- transformation frame; machine transforms the input to the output.

In further development, the problem frames propose some extensions that try to tackle some security issues. Starting from [60], the authors proposed the notion of anti-requirement (or security requirement). It captures the requirement of attackers. This proposal is refined into a set of abuse frame to the security problem [134]. Essentially, an abuse frame can be seen as a commanded behaviour frame that is performed by malicious users to compromise the system. The attack is materialised if the abuse frame can be composed in the system problem diagram. The purpose of abuse frames is to analyse security threats and vulnerabilities, and to assist the analysts to identify whether such threats can occur in the system (i.e., the problem world). Moreover, in [133] the authors propose three basic abuse frame - *interception*, *modification*, and *denial of service* abuse frame that can compromise confidentiality, integrity and availability respectively. Interception frame represents an information disclosure by attackers. Modification frame represents an information alteration where attackers want to change sensitive information. Denial of service frame models a situation where attackers intend to make unavailable or unusable a particular domain.

In [86], the authors use the problem frames to analyse the assumptions underlying security requirements. In fact, analysts *trust* the *assumptions* to be true in order to satisfy the security requirements. In recent security incidents, the incidents are made possible because the assumptions, taken by analysts in development, are not valid any more (i.e., the new technology makes such attack feasible). Representing explicitly trust assumptions allows analysts to develop satisfaction arguments [87] stating that a system is secure. Moreover, trust assumptions assume assertion acts that restrict the domain, and in some ways they contribute to the satisfaction of security requirements.

DITSCAP Automation Framework

Lee et al., in [132], propose a framework for modelling critical systems (including social aspects) which is based on a standard developed by US Department of Defense (US-DoD), called DoD Information Technology Security Certification and Accreditation Process (DITSCAP) [204]. The DITSCAP framework analyses the risk caused by vulnerabilities and threats of a system by evaluating the implementation security requirements. However, it does not quantify the risk level. This work aims at automating some steps of capturing, modelling, and analysis of security requirements. To this end, analysts are required to build *Problem Domain Ontology* (PDO). In [131], the authors demonstrate how to build a PDO from regulatory documents (e.g., DITSCAP [204]). PDO, which is machine readable, is composed of Requirement Domain Model (RDM), Goal Hierarchy, Viewpoints Hierarchy, and other domain specific taxonomies. In [131], RDM is built to capture the problem domain requirements from the DITSCAP document, and structure it hierarchically. Goal Hierarchy captures the objectives of the system that want to be developed. Requirements capture desires, ideas, and relationships among entities in the system; and their interpretation closely depends on the viewpoints that are modelled in the Viewpoints Hierarchy. Finally, PDO can be enriched by other related taxonomies. In DITSCAP PDO, it includes the Risk Taxonomy that captures broad spectrum types of risk that is related to the DITSCAP domain.

Moreover, there are other works that take similar approach (i.e., taxonomy/ontology based), such as ISO 27002 [105], Fault Taxonomy [17], Software Flaws [129], STRIDE [145]. These approaches are useful to assist analysts in identifying the sources of risks, but they require analysts to keep up-to-date with the current situation. It is because the ontology/taxonomy is prone to incompleteness, especially, when the domain application has not been well developed or there are many new technologies that enable new classes of attack.

2.3.5 UML-based Security Model

Several works aim to extend UML [157] for addressing security related issues, namely Abuse Case [144] and Misuse Case [186] that help in identifying use cases that can compromise the security of the system; SecureUML [138] that model access control policies in terms of UML diagram; and UMLSec [116] that extend UML with security features to capture some security requirements (e.g., confidentiality, integrity).

Abuse Case

Abuse Case [144], based on UML Use Case, aims to capture and analyse security requirements. An abuse case is a use case, representing an interaction between the system-to-be and one or more actors, where the result is harmful to the system or one/more actor(s) in the system. Actors here are the same with the one that participate in use cases of the system. In other words, Abuse Case focuses analysing how an internal malicious actor can cause harm to the system or other actors in the system. Unfortunately, abuse cases are not drawn together with the use cases of the system. Consequently, it does not allow analysts to investigate relations between use and abuse cases which can introduce conflict between functional (i.e., derived from use cases) and security requirements (i.e., derived abuse cases).

Misuse Case

Misuse Case [186], based on UML Use Case, captures security requirements using the notion of misuse cases as the inverse of use cases. Initially, a use case describes the functionalities that the system should provide for the users. The authors here extend the use case diagram with some additional concepts, most notably misuse cases and misusers to capture security threats to the system.

Unlike abuse case, misusers can be attackers external to the system. Moreover, misuse case is modelled together with use case in the UML use case diagram. It makes possible to represent which use cases that are threatened by particular misuse cases and actions that the system should take to prevent such misuse cases. In additional to the standard "includes" and "extends" relations, misuse case introduces some new relations in the diagram, namely "threaten" and "mitigate" to relate use cases and misuse cases.

SecureUML

SecureUML [138] focuses on modelling access control policies, Role-Based Access Control in particular, and integrating them into a model-driven software development process. SecureUML is realised in terms of a profile for UML [157]. SecureUML provides support to express RBAC concepts, like roles, permissions and user-role assignments. In particular, RBAC concepts are represented as metamodel types: User, Role, Permission, Actions, Resources as well as the relation among them. An access control configuration is an assignment of users and permissions to role. Permission captures the action that can be done by a role over particular resources. However, RBAC lacks, in general, in expressing access control conditions that refer to the state of a system (e.g., the state of a protected resource, date or time). To cover such cases, SecureUML proposes authorisation constraints, expressed in Object Constraint Language (OCL) [146].

UMLSec

UMLSec [116] is proposed as an extension of UML, in terms of a UML Profile [157], for modelling security related features (e.g., confidentiality, integrity, access control). Security requirements are captured

by means of an Abstract State Machine model and several UML stereotypes. Moreover, each concept is defined with a formal semantic that later allows us to perform formal verification on particular security properties (e.g., confidentiality, integrity). Associated constraints are used to evaluate specifications and indicate possible vulnerabilities. The main security requirements considered by this approach are:

- Fair Exchange - prevents frauds in an interaction among entities;
- Confidentiality - ensures that only authorised users know the information;
- Secure Information Flow - prevents information flow of sensitive information from trusted parties to untrusted parties;
- Secure Communication Link - ensures the security, for a given adversary model, of communication links between different parts of the system.

This framework extends several UML diagrams to model security requirements, such as:

- Class diagram - focuses on the exchange of data preserves security levels;
- Deployment Diagram - focuses on physical security requirements;
- Activity Diagram - focuses on secure control flow and coordination;
- Sequence Diagram - focuses on security-critical interactions;
- Statechart Diagram - focuses on security preserved within object;
- Package Diagram - focuses on functional relations between parts and the whole with respect to security.

However, UMLsec captures technical systems rather than socio-technical aspects. It focuses on analysing security as a service and often overlooks in analysing the requirements of why such a security service is necessary. This framework can used as a design framework to specify the detail design of the system, once the analysts have specified requirements that are incorporated means to realise stakeholders' needs and treatments to mitigate security risks.

2.4 IT Governance

Unlike previous sections that review the state of the art of related fields, this section reviews the state of practice for enterprises/organisations to ensure the governance of their IT assets. First, we consider Enterprise Risk Management (ERM) [58] that defines a framework to manage risks in an enterprise. This framework catches most attentions of the practitioners due to its comprehensiveness especially after the application of Sarbanes Oxley Act (SOX) [177] in US or BASEL II in Europe [19]. We then consider CoBIT [200] as a framework for defining control objectives of IT governance, IT Information Library [154] that is proposed by UK-Office of Government Commerce, and finally, we conclude with some related ISO standards (e.g., ISO 2700X on security and ISO 38500 on IT governance).

2.4.1 Enterprise Risk Management (ERM)

This framework has been developed by PricewaterhouseCoopers and Committee of Sponsoring Organisations of the Treadway Commission (COSO) [58] which is defined as a process that:

- is effected by every people at every layer of the enterprise;
- is applied in strategy setting and across the enterprise;
- is designed to identify potential events that may affect the enterprise;
- manages the risk to be within the enterprise risk appetite[1];
- provides reasonable assurance regarding the achievement of the enterprise objectives.

The framework has three dimensions: achievement of objectives, components of ERM, and entities. Entity is defined as an organisation that is purposed to provide some values to its stakeholders (e.g., business enterprise to run the business of shareholders). The framework categorises achievement of objectives as follows:

- Strategic - high-level goals, aligned with and supporting the mission of entity;
- Operation - effective and efficient use of the resources of entity;
- Reporting - reliability of reporting;
- Compliance - entity compliance to the existing laws, regulations, and standards.

The categorisation aims to address different entity interests and to identify the expectation of an entity to each category.

ERM is composed of eight interrelated components that can be viewed as steps to manage risk. The components are:

- *Internal Environment* is the foundation of all ERM components. It consists of how risks are viewed and addressed by individuals in the enterprise. Moreover, it includes risk management philosophy, risk tolerance, integrity and ethical values in entity;
- *Objective Setting* is a setting of high-level goals, which are aligned with, and supports entity visions/missions. Through this setting, the management can consider any alternative strategies to achieve strategic objectives considering the associated risks;
- *Event Identification*, an event is identified from internal or external sources of entity. It could affect the implementation of the strategy and the achievement of objectives. Events may have positive impact (called opportunity), negative impacts (called risk), or both. ERM considers the occurrence of an event as non-isolated occurrence because an event can trigger other events or they can occur concurrently.
- *Risk Assessment* assesses an event from two perspectives: likelihood and impact. It allows an entity to consider which event might have significant influence to the achievement of the objectives.

[1] the amount of risk that an entity is willing to accept in pursuit of its mission/vision

- *Risk Response* defines how entities will response to the risk. It includes risk avoidance, risk reduction, risk sharing, and risk acceptance. The management considers several factors in choosing the proper risk responses, such as:
  - Impact of a risk response on reducing risk likelihood or impact;
  - Cost and benefit of potential risk response;
  - Possible opportunities lose, once we apply risk response.
- *Control Activities*, policies and procedures are defined to ensure that risk responses are carried out once risks occur. In [58], COSO has defined types and several common control activities that are usually applied in risk management.
- *Information and Communication*, every enterprise identifies and captures relevant information. This information needs to communicate to the entire personnel, in a certain form and a certain time frame, to help them perform their responsibilities and roles in ERM.
- *Monitoring*, it is a process to assess the current presence and function of the ERM components. An entity in ERM might change over time; the entity objective can change. This fact could cause a risk, and its risk response, becoming irrelevant.

ERM covers nearly all steps of risk management process, but a precise definition of each step is still missing. Therefore, we adopt the COSO-ERM as the base line for our methodology, and we also adopt the more general concept of risk as an event that gives impacts (negative or positive) to the achievement of objectives. We will adopt the concept of event, which is more general than risk, to model risks and opportunities and eventually to reason about them.

2.4.2 CoBIT

COBIT [200] mission is to develop an IT governance control framework that is internationally accepted and implemented in enterprises. It provides a framework for business-process owners for organising IT processes to meet the organisation's objectives. COBIT is organised into 4 domains:

- Plan and Organise - provides the guidance for service delivery;
- Acquire and Implement - provides the solution by considering the some guidance/policies;
- Deliver and Support - makes the service available for the users;
- Monitor and Evaluate - monitors all the services and evaluates their effectiveness.

Those domains are organised into 34 IT processes that guarantee the governance of IT systems in supporting business objectives.

Essentially, controls aims to ensure the execution of IT processes in realising the organisation's objectives and they are not specifics to security aspects. COBIT provides a body-knowledge of the classes of IT process that exist in an organisation, and gives guidance how to control those processes. Moreover, COBIT has a strong notion on measurement that relates how business objectives are achieved (measured by KGIs), and how IT processes realised those objectives (measured by KPIs).

2.4.3 IT Information Library

ITIL [154] is developed on a common sense approach to service management - *do what works*. Moreover, they are adapting a common framework of practices and unite all areas of IT service provision into a single goal - *delivering value to business*. ITIL defines a service as *a means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks* [154]. Based on this definition, there are two main concepts of a service: delivering value and not caring how a service is implemented. ITIL organises the service management activity into 5 core stages of ITIL lifecycle:

Service Strategy provides guidance to perceive services as strategic assets more than just organisation capabilities. This stage is essential for developing policies, guidelines, and processes of service management across the ITIL lifecycle;

Service Design provides guidance to design and develop service, and also service management practices. It also includes design principles for converting business objectives into services. However, those principles are not only applied to new services, but include also maintaining or increasing services.

Service Transition provides guidance for developing and improving of the transition capability of new and changed services into operational ones. This stage ensures that the requirements of service strategy, encoded in service design, are well implemented in the service operation.

Service Operation includes the knowledge of day-to-day operation of service management. It provides guidance to achieve effective and efficient service delivery and to ensure the value retained by the customers and the service provider.

Continual Service Improvement provides instrumental guidance to maintain and create values for customers by means of better design, transition, and operation of services

Different from other frameworks, ITIL provides beyond WHAT-to-do to manage services by providing the details HOW they are done. However, the UK Office of Government Commerce (OGC), the author of ITIL, encourages an organisation to adapt each stage to best-suited practices depending on the culture of the organisation. Indeed, the framework emphases on managing value delivery of services of an organisation to its customers. However, it analyses an organisation as one single actor and does not capture which actors, in the organisation, generate the values to be delivered to the customers. This information is essential to assess (perceived) risks by each individual within the organisation, and also to allocate risk treatment efforts.

2.4.4 IT Standards and Best Practices

ISO 27001 on requirements for Information Security Management Systems (ISMS) [105]. It is an international standard that is prepared to provide a model for establishing, implementing, operating, monitoring, reviewing, maintaining, and improving an Information Security Management System (ISMS). It emphases on:

- Understand the security requirements of an organisation's information systems, and the need to establish policy and objective for information security
- Implement and operate controls to manage security risks

- Monitor and review the performance and effectiveness of the controls
- Improve the control based on objective measurement

Principally, ISMS adopts the Plan-Do-Check-Action model [62] for managing Information Security. This standard concentrates on security risks as the main drivers in defining security controls. However, sometimes an organisation might also be required to control some process because some regulations (e.g., SoX) prescribe to do so. This standard has specified WHAT-to-do to establish security requirements, to operate them in terms of controls, to monitor-review the controls, and finally to maintain-improve their performance. Moreover, the standard has also specified a format to document all of these processes.

ISO 27002 (initially ISO 17799) on code of practice for information security Management [106]. This standard provides general guidance on defining control objectives of information security management and also propose a complete taxonomy of security controls (i.e., in 11 categories) to achieve the objective. It also includes a guideline to implement the control in an organisation based on the best practices. However, all of these are too generic and may lead to the wrong implementation-operation of the security controls. Note that there are some efforts done by ISACA [2] to map the IT processes in COBIT with some relevant controls in ISO 27002 [200]. This mapping is useful to help analysts in identifying which controls are relevant for protecting their IT processes.

ISO 27005 on code of practice for information security management [106]. It provides a guideline for information security risk management. This standard expands from risk assessment where the risks are identified and estimated, risk treatment, risk acceptance, risk communication, and risk monitoring. Moreover, this approach expands across all phases of ISMS-ISO 27001 (i.e., Plan, Do, Check, Action) and it is suggested to be performed along ISMS. However, it does not model explicitly business objectives and how risks are developed in the business context. As most of the standards, it has proposed a standard baseline for risk management, but it is too generic and often there is no clear-cut how-to-do such step.

ISO 38500 on corporate governance of information technology [108]. The objective of this standard is to provide a framework of principles for Board of Director to use when evaluating, directing and monitoring the use of information technology (IT) in their organisations. In general, good IT governance should comply with the following principles:

- Responsibility - each individual/group understand and accept their responsibilities in respect to supply and demand for IT;
- Strategy - business objectives take into account the current and future IT capabilities, and IT strategic plans satisfy the current and ongoing business objectives;
- Acquisition - IT acquisitions are made for valid reasons (e.g., careful analysis, clear & transparent decision making, balance between cost-benefit & risk-opportunity);
- Performance - IT fits in supporting the organisation;
- Conformance - IT complies with all mandatory legislations and regulations;
- Human behaviour - IT policies, practices, and decisions show respect for human behaviour;

[2] http://www.isaca.org

However, this standard does not prescribe in detail how to manage the IT governance in an organisation. According to ISO 38500, a director should do three main tasks to govern IT:

- evaluate the current and future of IT;
- direct preparation and implementation of plans and policies to ensure the usage of IT in realising business objectives;
- monitor conformance of the policies and performance against the plan;

All these tasks should keep in mind the basics six principles of IT governance. ISO 38500 gives only basic principles and tasks to govern IT in an organisation. There are not enough details on WHAT things need to be done and HOW things should be done. Finally, governance can be seen as one reason why risk management must be established in an organisation. However, good governance does not imply that the organisation will success in pursuing its business objective because it only indicates that the organisation has addressed (most) relevant risks to its businesses, which is not necessarily sufficient.

2.5 Summary

A system is defined as a collection of organised entities where each entity interacts with at least one other entity, and they all serve a common objective [35]. The similar principle is also applied for CISs in the context of socio-technical systems. From this study, we reveal some gaps between the proposals coming from these research fields (i.e., software engineering, security-dependability engineering, and IT governments) in managing risks. In S&D engineering, the main emphasis is on the system, in terms of the technical one. Unfortunately, the S&D of a system, at large, cannot depend solely on the technical aspect. In other words, having S&D technical systems is required to guarantee the performance of systems, but it is not sufficient. One must consider other aspects, such as organisational-setting and business assets. Software engineering mainly addresses risk from a modelling perspective and does not propose much from the assessment point of view and also overlooks the uncertainty analysis that can affect the operational and maintenance of the software system. Actually, IT governance provides comprehensive methodological frameworks to analyse ISs and to ensure their operation being aligned with the strategic objectives of the stakeholders, despite the fact they are often too generic. Moreover, most of the steps are non-formal and therefore they require enormous human involvement even for tedious tasks (e.g., calculating risks). Last, but not least, most frameworks assess the risk in a single perspective, which is inappropriate. It is because an organisation is composed of several individuals that may have different perception towards the risk.

Chapter 3

Challenges and Research Approach

In ISO Guide 73:2002 [109], risk management is defined as *"coordinated activities to direct and control an organization with regard to risk"*. Identifying risk is only a beginning of activity on managing risk. Analysts should continue the process by analysing in detail and introducing some countermeasures until all risks have been addressed (i.e., mitigated or accepted). The most crucial step of risk management process is the part when analysts assess the risk. If the assessment is incorrect then it is likely that the organisation will adopt inappropriate countermeasures or treat the wrong risks.

- How much risk is the organisation exposed to? - *assessing Risk Level*
- How much risk are our Information Systems (ISs)? How much does it contribute to the risk of the organisation? - *assessing IS Risk Profile*
- How does each individual in the organisation perceive these risks? - *estimating Risk Perception*
- How much risk can the organisation sustain? - *deciding Risk Tolerance*
- What can we do against them? What are the effects to the risk level and the side-effects to the business? - *eliciting Risk Treatment*
- Which treatments should we adopt for a given limited resource? - *deciding Risk Treatment*

These are some basics questions that most stakeholders/analysts post regarding risk in the organisation related to the IT system. Though they seem to be simple questions, they demand very comprehensive and complex analysis to come up with appropriate answers.

In this chapter, we detail the challenges in assessing risk, particularly when it is done together with the requirement analysis in a socio-technical context. To better understanding the challenges, we start the chapter by detailing the notion of risk. We then explain our research approach, the Tropos Goal-Risk Framework, based on two important works in requirement engineering, namely Tropos [38] and Defect Detection and Prevention (DDP) [56]. Finally, we detail the research contribution of the thesis.

3.1 What is Risk

Risk, in ISO Guide 73:2002, is defined as *"combination of the* **probability** *of an event and its* ***consequence****"* [109]. In fact, the term "risk" is generally used only when an event delivers the possibility of

negative consequences. Risk can be originated from a deviation of the expected outcome (e.g., processing a loan takes longer) or from a normal operation (e.g., a user input that activates a dormant fault) [122]. Essentially, a risk is constituted upon three basic concepts: *event*, *likelihood*, and *severity*

Event We adopt the WordNet[1] definition for event:

- something that happens at a given place and time;
- a special set of circumstances;
- a phenomenon located at a single point in space-time;
- a consequence (i.e., a phenomenon that follows and is caused by some previous phenomena).

The notion of event used here is more general than some similar concepts, such as

- *threat*, in computer security [162], is defined as a set of circumstances that can cause loss or harm. *Attack* then is defined as the threat executed by an agent to exploit the system vulnerability;
- *hazard*, in safety engineering, is a situation that has the potential for causing damage to humans, systems, or to the environment [49];
- *failure*, in reliability engineering, is the state where the delivered service deviates from correct service [17]; Prior to this situation, a system may end up in the state, called an *error*, that is incorrect state but it does not manifest as a service deviation.
- *accident*, in Air Traffic Management (ATM), is an occurrence associated with the (aircraft) operation that can result in a person fatally or seriously injured, the aircraft sustaining damage/structural failure or missing or inaccessible [99]. However, an event can be just an *incident* (i.e., an occurrence other than accident) that can lead to the accident.

In other words, those concepts put more emphasise as a potential circumstance that could cause harm or loss without definition of any notion of likelihood.

Likelihood and Uncertainty *Likelihood* refers to the notion of how likely an event will occur in the future. One might be confused with the following notions: uncertainty, likelihood, frequency, probability, possibility, and fuzzy. In fact, there are several mathematical theories to model uncertainty, such as probability theory [171], possibility theory [67], fuzzy set [227], Dempster-Shaffer (DS) theory of evidence [182] etc. At glance, these theories are alike and can be interchangeably used. However, this assumption is misled, because these theories are developed for different purposes and represent different classes of uncertainty.

We intend to clarify all of these notions following existing literature [135, 155, 190, 212]. O'Hagan et al., in [155], argue that uncertainty could be divided into two classes: aleatory and epistemic uncertainty. The former is induced by randomness, and the latter is due to incomplete knowledge. Smithson, in [190], argues that uncertainty is also originated because of vagueness or ambiguity. In this dissertation, we categorise uncertainty into three classes: aleatory, epistemic, and imprecision.

Notice uncertainty emerges though one has a complete knowledge [151]. For instance, we know all about a dice including possible outcomes, but still we cannot decide for certain the outcome of rolling a

[1] `http://wordnet.princeton.edu/`

dice. In this example, we can categorise the event of rolling dice is exposed to an aleatory uncertainty induced from the nature of **aleatory** (i.e., variability and randomness).

Example 3.1. *The radar engineers have, almost, full knowledge how a radar work, but they still put another radar as a backup because they do not know in certain when the radar will fail in the future.*

This class of uncertainty has been heavily studied in the dependability community. Typically, this uncertainty arises due to spatial variation (e.g., the development site is different with the runtime site), temporal fluctuation, and development variability. This uncertainty may result in failures, hence analysts should investigate all possible uncertain events in this class and analyse them. The Probabilistic Risk Analysis (PRA) [21] and Fault Tree Analysis (FTA) [196] are useful in computing the level of uncertainty. Both frameworks are built using the Kolmogorov Probability Theory [124] as the underlying mathematical theory. However, this uncertainty can hardly be reduced. One can use some redundancy techniques [90] (e.g., primary-secondary, main-backup, multi-version programming) to deal with such uncertainty.

Epistemic uncertainty is caused by incomplete knowledge.[2]

Example 3.2. *Aircraft handover might fail though the ATM system operates normally. It is because analysts are hardly possible to anticipate all possible actions that a controller will do for an aircraft handover.*

Most of security problems have arisen because of the incompleteness. Security analysts are hardly aware of all possible threats that attackers might launch to the system. Essentially, there are two subtypes of epistemic uncertainty: 1) visible incompleteness where *we are aware about what we do not know*, and 2) blind incompleteness where *we are not aware about what we do not know*. The example 3.2 is an example of visible incompleteness. Essentially, it can be reduced by empirical efforts, such as interviews, collect more data, expert judgements, more test cases, etc.

Example 3.3. *In smart-card technology, attackers can extract secret keys and consequently compromise the integrity of smart-card by analysing the power consumption of a smart-card [123].*

It is an example of blind incompleteness, because analysts had never got an idea how such possibilities existed. Essentially, there is no such mechanism to reduce this subtype of uncertainty. Having a system with high augmentability is beneficial for this uncertainty because it should allow us to easily upgrade the system as soon as a new vulnerability is discovered. Typically, analysts use the probability theory (with Bayesian interpretation) or the DS theory of evidence to calculate the degree of uncertainty, such as Bayesian Belief-Network [95, 224], Belief-based model [115]. Aleatory and epistemic uncertainty are the ones that define the likelihood of an event and consequently the risk of the system. However, they require different treatments to reduce/to deal depending on the nature of their uncertainty. Note that often analysts mix both classes since the analysis lacks of hard evidence.

The last class is **imprecision** uncertainty which is introduced due to imprecision. The imprecision can be caused by unclear definition (e.g., ambiguity, vagueness) or the limitation of measurement/assessment techniques. In other words, this uncertainty may expose a state-of-affair together two previous uncertainties.

Example 3.4. *Statistically, under a normal operation Air Traffic Controllers (ATCOs), with their workstation, are always capable to ensure the safety of air traffic.*

[2] In [218], Vose calls epistemic uncertainty as "uncertainty"

Assuming the state-of-affair in example 3.4 is correct and sound, it still suffers from uncertainty (i.e., imprecision). What does it mean to have a normal operation? What are the criteria of an air traffic to be safe? Thus, analysts should fix this issue before assessing the likelihood of an event. In other situations, the imprecision is introduced due to the limitation of the techniques.

Example 3.5. *To reduce uncertainties of the event in example 3.2, one may conduct interviews to some controllers (defined by a statistical sampling technique).*

However, the outcome of the interviews still contains the uncertainty (called error) due to the limitation of the techniques. Essentially, the imprecise uncertainty does not directly drive the value of likelihood, but it can play a significant role in defining the extent of risks. For instance, the risk level of an event of *having immediate air traffic increase* depends on the likelihood (i.e., variability) of its occurrence and the how big the increase is, which is often is an imprecise value (e.g., 10-20 aircraft).

Severity expresses the extent of consequence that an event delivers. In fact, it is irrelevant to consider an event without any consequences to the system. COSO, in [58], defines risk as an uncertain event with negative impact. This definition allows us to analyse also opportunity (i.e., uncertain event with positive impact) instead of only risk analysis.

In risk management, often the severity of a risk is expressed in terms of the extent of *loss* if the risk occurs, and the likelihood is expressed as a probability (i.e., $[0, 1]$). Hence, the level risk is calculated using the notion *expected loss* (i.e., expected value [171]) which is the product of severity and likelihood ($\lambda(E)$), as follow:

$$ExpectedLoss(E) = \lambda(E) \times \sum_{isRisk(E,A) \wedge A \in asset} loss(E, A) \tag{3.1}$$

where the severity is the sum of all *losses* of assets (A) because of the risk (E).

3.2 Challenges

Though risk has been studied since long time back by several disciplines of knowledge, it seems that one may overlook or misjudge risks while developing a system. For instance, it is evident why some critical information systems (e.g., air traffic management, nuclear power plant control, internet shopping) have suffered from some breaches or failures during their operation. One may argue that they never think about these possible threats, or consider the failures as insignificant events. These situations often occur since risk analysis is more an "art" process, rather than an engineering one, hence understanding the nature of risk is essential for managing it. In this section, we present some characteristics of risk from a socio-technical perspective:

- **Risks are intangible** matters because they address the issues of something bad that can happen in the future. There have been some works in risk communities and psychology that study how an agent (re)acts against the risk [25, 179]. Unfortunately, they come only as an essay which aim to understand how risks are perceived, and not how to analyse and manage these phenomena in the system development;
- **Risks are ubiquitous** and affect our decision making process. Risk can come as a result of our activities therefore eliminating risks can result in limiting our activities and related opportunities. Moreover, a risk of an actor can affect throughout the organisation since in an organisation actors might depends on each other for particular tasks/goals;

- Some actors, in the same organisation, might **value an asset differently**. For instance, IT staffs argue that the firewall is the most valuable asset in the organisation because it prevents intrusion from attackers. However, marketing staffs argue differently because the firewall does not have any actual business values. This situation can result in mis-allocation of resources to protect the business. Some works (e.g., Delphi method [136], Expert Judgement [155]) propose a framework that helps actors to come up with an agreement over the value of an asset. However, this approach is not suitable if the organisation operates a critical system, because we can agree on a particular value, but the action, carried by each actor, is still depending on the perception of each actor;

- Often the result of risk assessment is **not well perceived** throughout the organisation. Consequently, different agents in the organisation perceive the same risk differently and it can introduce some issues. For instance, Alice, a CEO, perceives the risk of intrusion as catastrophic for the business trade secret. Since the trade secret is locked up in the vault, Bob, a security guard, thinks that he does not need to screen each guest. Actually, in economic and commerce area, many efforts [117, 128, 220] have studied the correlation between the level of risk with the reaction of the actors towards the risk. In those studies, there are situations where individuals prefer to stay away from a risky subject, but there are a situation that they behave otherwise;

- It is **hardly possible to obtain "good" inputs** to perform IT risk assessment. Compared to financial risk, IT risk assessment is often conducted based on poor data (i.e., qualitative-subjective) obtained by experts' judgement; In such setting, the outcome of risk assessment is prone to subjectivity of the assessors.

- At last, not least, most of analysts identify risks over existing assets that are valuable to the organisation. Readers can revisit the work of ISO 27002 [106], Fault Taxonomy [17], Security Flaws [129], and STRIDE [145], essentially they emphasise on the risks that are originated from the resources or infrastructures. Actually, **assets are beyond just the "resources"** (e.g., servers, production machines, telephones, trade secret documents, and communication links), because there are other things that are valuable to the organisation. For instance, the *procedure of aircraft handover* is a critical business process for the business of an air traffic service provider. If the business process is disrupted (e.g., the controllers is careless), then the business is at risk, though all underlying infrastructures/resources (e.g., radars, working stations) work correctly. Moreover, many recent regulations (e.g., Sarbanes-Oxley Act [177], Basel II [19]) and standards (e.g., ISO 27000 series [105, 106, 107], BSI 25999 [43]) require an organisation to be able to guarantee the continuity at the business level, which is far beyond just the availability of technical systems. Therefore, analysts should identify risks at all aspects of the organisation that can prevent the organisation from achieving its objectives (i.e., creating values), and treat them according to the organisation priority and resources.

Once analysts are able to understand those characteristics, we believe they can better analyse risks, and treat them according to their criticality. However, risks of IT systems, particularly security, have some particular characteristics that are unique comparing with other types of risk (e.g., financial risk, operational risk, legal risk):

- Typically, IT risks are categorised as operational risks since their occurrence can disturb the operational of internal (business) processes. Unfortunately, it is **not always operational risk** because IT risks can result in many forms of risks. For instance, the failure of Health ISs can result in legal obligation to the patients (i.e., legal risk) because one suffers from mistreatments. In the case

of loan origination process in a bank, an error in a finance information system can result in the disturbance in the bank's financial statement. Such situation might lead to the liquidation of the bank, because the bank could not meet the Credit Adequacy Rate (CAR), defined by the central authority. In fact, Rice, in [167], illustrates the similarities of cement for modern civilisation and IT systems for information era. We hardly realise how significant the cement in developing the modern civilisation. It plays a significant role, but most users do not realise its presence or role; as happens also in the IT. The strength of a building closely depends on the combination of cements in making concrete, which is similar how secure an IT system depends on the security of each component. The differences is cement, as a piece of technology, have been well developed (or even there is no new break through any more) while IT is still immature. Moreover, cement (i.e., portland cement) is getting stronger with time while IT is otherwise. The older an IT system, the more vulnerabilities are discovered. Therefore, analysing IT risks based on the goals/business objectives of the organisation allows us to have a comprehensive view what the assets, at large, that need to be protected.

- In [179], Schneier mentioned that security is invisible even from security administrators. In other words, users cannot perceive the effect of security while the system is secure, and they only then **realise when security is broken**. Therefore, it is so evident why people in business do not like IT security [59]. Since we argue security as a quality of attribute of software systems [45], then the challenge is how users can perceive such quality since its presence is hardly visible and its failure can be easily felt. In fact, some authors aim at demonstrating the value of security investment by means of computing Return on Security Investment [95]. However, we feel such metric is inappropriate since security, like insurance, is not an investment because it does not generate values. However, their function is more protecting the value generated by the business. In this work, we try to propose a framework that can assess the perceived risk of each actor in the organisation, besides the actual risk that threatens the organisation. Using this framework, we can demonstrate how security measures can affect the security-feeling of an actor captured as perceived risk. In other words, traditionally risk management aims at mitigating the risks in the organisation such that the risks are acceptable by the organisation (i.e., in the viewpoint of Board of Directors/Shareholders). In security risks, the risk management should also mind about the perception of each actor in the organisation. Notice the right level of risk perception depends closely on the domain application. For instance, in Loan Origination Scenario clerks should perceives that their tasks are at high risk, though the system has handled most of human errors, so that they perform the tasks in caution. Conversely, in safety critical (e.g., Air Traffic Management) controllers operate under stress since they realise that human lives are at stake. Hence, any security measures ideally should also lower the perceived risk of the controllers.

These are characteristics of risk that we consider in developing the Tropos Goal-Risk Framework, though we believe there are many others. All of these convince us of the necessity to capture risks especially IT related ones, at business and organisation level by considering socio-technical aspects

3.3 Research Approach

Traditionally, risk analysis is used in software development to identify situations or events that can cause project failures [29, 172]. Risk analysis has also shown its importance during software design in evaluating system criticality [214]; risks are analysed and countermeasures are introduced. Typically, countermeasures amount to design fine-tuning. However, after any change to a design, requirements need to be

re-evaluated to ensure that they still hold. Moreover, in some cases risk mitigation requires the revision of the entire design and possibly of initial requirements. In conclusion, there are obvious benefits in conducting risk analysis during requirements analysis. Not surprisingly, there have been several recent attempts to integrate risk analysis with requirement analysis [31, 33, 57].

However, these works concentrate mainly on risks of the late requirement analysis and not the early one [226]. This means that risks are analysed along technical system requirements, rather than stakeholders' needs. However, it is important to consider risks, and more generally events[3], since early requirement phase, where stakeholders' objectives are captured and analysed. This allows us to understand better the business and organisational setting before concentrating into the details of software system. Moreover, such approach will enable us to capture a wide range of risks that are related to the business and not necessarily related to software systems (e.g., employees on strike, conflict of interests).

Generally, requirement analysis is exposed to several prominent issues, such as elicit wrong requirements, elicit incomplete requirements, and suffer from requirements change. By analysing risk in requirement phase, one might claim that the elicited requirements are robust, because they have encompassed risks associated with the stakeholders' goals and necessary countermeasures to mitigate them besides the requirements from the stakeholders' needs. Hopefully, the number of design modifications and requirement revisions introduced by risk and countermeasure will be reduced. Considering risk at the early requirement analysis introduces novel and valuable criteria for evaluating alternative ways to fulfil requirements, and also allow analysts to anticipate risks at requirement level and not only at design level. For instance, countermeasures introduce additional cost to the project, hence alternative requirements must be evaluated not only on the base of the cost to realise the stakeholders' needs and the qualities of the system, but also considering the associated risks and countermeasures costs. Of course, this approach does not replace the traditional risk analysis that is done during the design step (or any further steps in the software development), but rather they are as complementary since there are always new problems as soon as we introduce a new solution. To realising this approach, we adapt *i**/Tropos [38, 226] (i.e., including its descendent SI*[230]) with the idea of three layers analysis: objective, risk, and mitigation from DDP [56]. In next subsections, we overview *i**/Tropos and DDP as research baselines.

3.3.1 Defect Detection and Prevention

Defect Detection and Prevention (DDP) [56] is a model that is developed and applied in JPL and NASA. DDP consists of three layers model (Objectives, Risk, Mitigations) as depicted in Figure 3.1. Objectives are the things that the system has to achieve. Risks are all the kinds of things that, once occur, lead to the failure of objectives achievement. Finally, mitigations are action that can be applied to reduce the risks. Despite these, there are two relations that relate risk to objective, and mitigation to risk, namely impact and effect. Impact is defined as the quantitative representation of objective lost once the risk occurs. Effect is defined as quantitative representation of risk reduction if the mitigation is applied.

In this model, each objective has a *weight* to represent its importance. A risk has a *likelihood* of occurrence, and mitigation has a *cost* for accomplishment (namely resource consumption). Moreover, the DDP framework specifies how to compute the level of objectives achievement and the cost of mitigation from a set of taken mitigations. This calculation allows us to evaluate the impact of the countermeasures and thus supports the decision making process. Ideally, analysts adopt the most beneficial solution where the benefit is computed over the summation over the weight of attained goals, and the total cost is computed over the summation of the cost of selected mitigation. Additionally, the DDP model also

[3]Risk is an event with a negative impact, whereas an opportunity is event with a positive impact

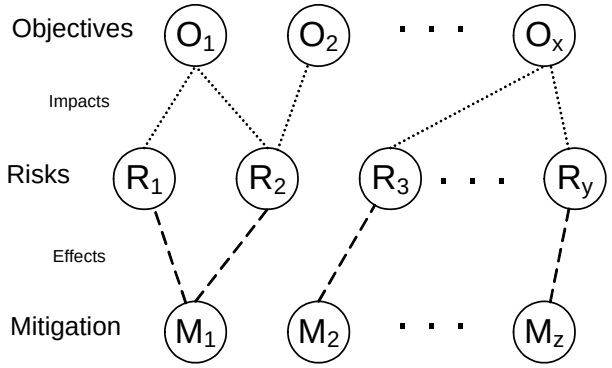


Figure 3.1: Topology of a DDP Model

allows us to work together with other quantitative tools (e.g., FMECA [205], FTA [196]) to model and assess risk/failure. In [74], the authors demonstrate how to integrate FMECA and FTA into a DPP model.

However, there are some differences between the concept of objective [74] and the concept of goal in goal-oriented requirement engineering [2, 38, 211, 226]. In goal-oriented, the goal has a structure (e.g., tree, graph). Conversely, the DDP depicts objective (also risk and mitigation) as a solitary object (i.e., there is no relation among objects in the same layer). This feature could not model the situation (e.g., domino effect [166]) in which the occurrence of a risk can increase/decrease the likelihood of other risks. This model allows us only to depict the relations between layers (i.e., risks to objectives, mitigations to risk) that are not enough to represent all possible situations. For example, there is a condition where mitigation can also obstruct the objective achievement besides mitigating the risk. Our framework will adopt and enhance the idea of three-layers model.

3.3.2 Tropos

Tropos is a methodology that uses agent and mental-state notions (e.g., goals and tasks) along all its software development phases [38]. The methodology covers all the phases, from early requirements analysis until implementation. Tropos, especially in early requirement phase, allows for advance understanding of the organisational setting where the system-to-be must operate. Tropos uses *i** [226] as a modelling language to capture and analyse early requirements of stakeholders.

In early requirement analysis, analysts identify relevant stakeholders, and model them as social actors, who depend on one another for goals to be fulfilled, tasks to be performed, and resources to be furnished. Through these actor dependencies, one can answer *why* questions, besides *what* and *how*, regarding system functionalities/requirements. Answers to *how* questions ultimately link system functionalities to stakeholder needs. Analysis then continues by refining goals where each goal is decomposed into subgoals, and positive/negative contributions are established among goals. This process ends when all goals/subgoals has been assigned to capable actors (e.g., itself, other actors, or new actors). The result of the analysis is a goal model that characterises alternative ways of fulfilling the top-level goals. In Tropos, the technical system is treated as an actor as well, with obligations to fulfil goals for external actors. Components of the technical system are also treated as actors and the system architecture consists of a collection of system actors with inter-dependencies among them.

Through goal models, the analyst can evaluate alternatives for the satisfaction of stakeholder goals and can also choose among them on the basis of criteria representing required qualities. However, Tropos

as well as most other goal-oriented approaches do not consider risks during requirements analysis. This means that the chosen alternative may also be the riskiest one. For instance, suppose that in order to satisfy receive loan application the bank can either receive electronic application or receive hard-copy application. Although, the electronic solution is clearly cheaper, it can introduce risks such as forgery of the application by external hackers or even by internal employees.

3.4 Research Contribution

As described in the previous section, literature did not propose solid solutions to the risk challenges for developing CISs. This thesis is one effort to answer some of these challenges proposing the following technical contributions:

- *A modelling language* for capturing and analysing risks in socio-technical systems. Our proposal is defined as an extension of the *i**[226], Tropos [38], and SI*[230]. The modelling concepts are organised into three-layers conceptual analysis: *value*, *event*, and *treatment*. In value layer, analysts capture value generators in an organisation at all levels. In this framework, we distinguish them into three levels: objectives, processes, and artefacts levels. An objective requires, at least, a process to realise it, while to execute a process, it can consume an artefact or can result in artefact. Moreover, this layer also captures social-dependencies among actors in the organisation. Each level has its own risks that are depicted in the event layer. They can be overlapped since the risks of a process can be seen, indirectly, as the risks of any objectives that require that process to achieve them. Risk is captured as an event with negative impact, while opportunity is an event with positive impact. This representation allows analysts to perform trade-off analysis in mitigating a risk that is also beneficial for other actors in the organisation. Finally, treatments are modelled in the treatments layer. In a GR model, treatments are captured using the same concept as process, but treatments aim at mitigating risks and not achieving business objectives. Since risks, in general, is characterised by likelihood and severity of loss [109], a treatment can operate in two modes: reducing the likelihood or alleviating the severity. Moreover, it is essential to distinguish between the actual risk and the perceived one because often actors, in an organisation, believe that they are at risk though the organisation actually have employed the state-of-art technology to control the risks or even vice versa. We can imagine what happens if the safeguard is delegated to the risk controller that perceive that the risk level as low, while the owner of the risk thinks otherwise. Therefore, the GR framework supports both types of risk assessment.

- *A risk management methodology* allowing a systematic process to manage risks in socio-technical systems. This methodology essentially is an adaptation over the methodology proposed by ISO 27005:2008 [107] with the basic concepts of the GR framework. The methodology defines a process that guides analysts in analysing assets (i.e., value generator) of an organisation, identifying related risks to those assets, and finally eliciting possible treatments that mitigate risks. The methodology is provided with automated reasoners that assist analysts in performing several analyses, such as assessment of risk level, evaluating possible combination of treatments, or synthesising the "best" solution that meets a given criteria (e.g., risk level, maximum cost). Moreover, the proposed methodology should be applicable for many purposes (as illustrated in Chapter 7 - Chapter 9). Finally, the methodology is resulted a set of requirements that constitute of necessary measures that are required to achieve business objectives and to protect them against significant risks.

- *A formal framework* for requirements verification and validation in terms of risk aspects. Essentially, two forms of reasoning have been developed based on previous works: forward reasoning [80] and backward reasoning [181]. Forward reasoning aims at computing risks analysis by assigning initial risk values; the reasoner computes and propagates the risk values across the organisation. The reasoner is built adopting the same intuition of the Dempster-Shafer (DS) theory of evidence [63, 182] This reasoning can be performed under qualitative or quantitative value of evidence. Moreover, in further development, this reasoner is extended to perform more complex analyses (e.g., business continuity analysis [7], design alternatives evaluation [10], business solutions evaluation [14], security and dependability solution assessment [13]). Backward reasoning works in the opposite direction. Starting from assigning desired values to goals (i.e., maximum risk level), the reasoner tries to find a possible solution that also satisfies a given constraint (e.g., maximum cost, maximum load each actor).

- *Computer-aided support* is built to assist analysts in developing a GR model and analyse them. Specifically, the tool supports analysts in capturing requirements and risks using the GR concepts and graphical models. All analysis techniques that are applicable for the formal framework have been included in the tool. Hence, the tool is able to guide the analysts in refining the model so that the desired properties of the model are satisfied.

Notice we put more emphasis in the coverage of risk, rather than the precision of risk assessment. It is hardly possible to obtain a precise risk level since the GR framework operates at requirements level. However, the identified risks here can be used and refined in the further phases of software development.

3.5 Summary

In this chapter, we have described the challenges for risk assessment and particularly for IT security risks. To answer these challenges, we propose to extend a requirement engineering methodology, namely Tropos, that is built upon *i** and SI* as the modelling language. The extension, namely Tropos Goal-Risk framework, adopts the idea of three layers model (value, event, treatment) from Defect Detection and Prevention (DDP). Moreover, the GR framework aims at capturing risks (negative events) at three level of organisation: objective, process, and artefact. In this way, analysts hopefully do not overlook any risks that are significant to the business and organisation. Moreover, the GR framework offers the capability in assessing perceived risks, besides the actual one, and a tool has been built to support analysts modelling risks and perform different analyses of the model.

Chapter 4

Tropos Goal-Risk Modelling Framework

Risk management is usually an activity done at the technical level. However, many classes of risk can be referred, anticipated, and avoided at the organisational or business level. We here advocate the necessity of considering risk since early phases of the software development, namely at the requirements analysis phase. Analysts can consider a broader range of risks, avoid eliciting risky requirements, and incorporate necessary safeguards in requirements.

Existing approaches lack of the ability to model risk-related concepts (e.g., events, treatments) during the requirement analysis (i.e., organisation and technical system aspects). On one hand there exist modelling frameworks that capture requirements (e.g., *i**, problem frames), but they do not consider any form of risk (or in general uncertainty). On the other hand, there are proposals for modelling risk (e.g., CORAS, DDP, FTA), but they operate at the lower level (i.e., technical artefacts). We propose a modelling framework, the Tropos Goal-Risk (GR) modelling framework, which adapts some basic concepts from *i** and SI* and extends with risk related concepts. The framework is intended to capture requirements of a social-technical system, relevant risks, and necessary treatments to mitigate them. Essentially, it is based on three layers conceptual analysis: value, event, and treatment, which an adaptation from the three layers of DDP.

This chapter is organised as follows. Section 4.1 describes the three layers model, while 4.2 introduces some additional concepts for capturing perceived risk, and then Section 4.3 concludes with a summary.

4.1 Three Layers Model

To enable analysts in assessing risks during requirement analysis, we propose a modelling language which is adapted from *i** [226] and SI* [230] and organised along the idea of three-layers model of the DDP (i.e., objective, risk, and mitigation) [56]. In general, a Goal-Risk (GR) model consists of three-layers representing values, events, and treatments as depicted in Figure 4.1. The value layer captures the concepts related to the actors' needs and the means to achieve them. We assume the system generates the value when it satisfies the stakeholders' needs. Uncertainties are modelled in the event layer. To reduce the risk of uncertainties, countermeasures are introduced into the organisation and modelled in the treatment layers. In the following sections, we detail all constructs constituted in each layer and relationships among constructs intra-/inter-layer. To simplify the explanation, we here assume all requirements and risks are in a single actor's rationale (i.e., an organisation can be seen as a "big" actor).

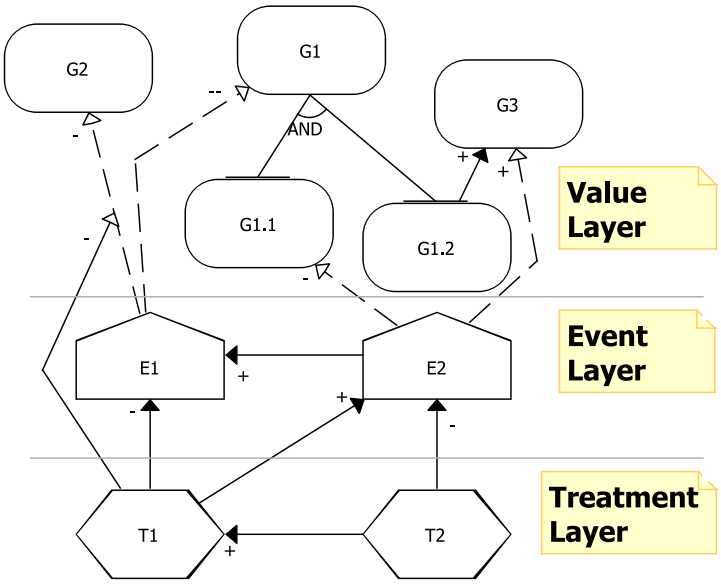


Figure 4.1: Goal-Risk Model

4.1.1 Value Layer

The value layer captures the valuable concepts for an actor that are modelled in terms of the strategic interests of the actor and some means to achieve them. We assume that the value is generated when the actor's interests are satisfied (i.e., either by itself or another means). To capture strategic interests, we adapt the notion of *goal* that have been well studied in Goal-Oriented Requirement Engineering (e.g., KAOS [61], *i** [226], the Tropos goal model [80]).

Example 4.1. *Executive controllers aim at managing the current air traffic, and in the same time are interested to have orderly incoming traffic.*

COBIT [200] categorises the strategic interests on an enterprise into: enterprise goals and IT goals. The former refers to the interests related to the business, and the latter refers to the goal that must be provided by IT systems. However, in the GR model we boil down both notions into a single concept of *goal*. This representation allows analysts seamlessly analysing business aspects down to IT related ones, and ensure all IT goals are driven by the business goals. Moreover, the value layer also captures the means to realise the strategic interests. We here distinguish the means into: 1) *process* level where a set of actions are required to achieve the goals; 2) *artefact* level where some resources are required to achieve the goals or to execute some actions.

The following is the basic concepts which constitutes the value layer.

Goal is a state of affairs which an actor intends to be achieved.

Softgoal is a "special" type of goal, but it does not have a clear-cut definition of its satisfaction. This concept is adopted from the work of *i** [226] and Non-Functional Requirement (NFR) [150]. For instance, one cannot give a precise definition about what the goal of *high usability* is. Moreover, there are various interpretations and usages of softgoal (e.g., security requirements [141], security constraints [149], NFR [72, 150]) that make it hard to grasp the basic essence. Softgoal, in this work, is

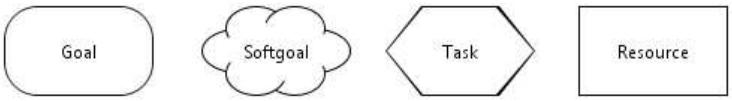


Figure 4.2: Goal, Softgoal, Task, Resource

treated as criteria or preference [174] that an actor desires to be achieved. In other words, an actor will do anything, in his/her power, to achieve the goal by adopting the means that will maximise the attainment of softgoal.

Task captures a course of actions that results in a desired state. A task can be executed to satisfy some goals or to produce some resources. It is similar with the concept of *plan* in Artificial Intelligence [173], *Task* in Work Breakdown Structure (WBS) [203], or *process/activity* in a business process [222].

Resource represents a physical or information artefact. Moreover, a resource is an entity without any intention. The concept of applications, information, and infrastructure, in COBIT, are considered as resources. A resource can be resulted from the execution of a task, and can be required for a task execution.

Essentially, the concept of goal (softgoal), task, and resource are used to capture the strategic interests of an actor that are organised into three levels: objective, process, and artefact level. Moreover, their graphical representations are depicted in Figure 4.2 and refer them as *business object*.[1]

Moreover, those concepts are also characterised with several properties: *evidence*, *value*, and *cost*.

Evidence represents the number of evidence that is refined into two dimensions: SAT and DEN. SAT captures the satisfaction evidence of a goal will be achieved, and DEN represents the denial evidence in which a goal will fail. The notion of evidence is adopted from the Dempster-Shafer Theory of Evidence [63, 182], which is similar with the notion of probability. The details about it will be explained in Chapter 5 along the details of the underlying formal framework. The similar intuitions are also applied for task and resource.

Value is something important/valuable resulted from the satisfaction of a goal. An actor might perform a task, but if the outcome of the task does not achieve the state-of-affair specified by the goal, then we can reasonably assume that the actor does not obtain the value that s/he needs. In [189], Smith distinguishes the value of an object into *value-in-exchange* and *value-in-use*. The former refers to the intrinsic value held by itself (e.g., the value of gold) and (relatively) independent from any subjectivity. The latter, often called as *utility*, is the value defined by some judgement how the usage of the object is important/valuable for an actor [23]. To avoid confusion, we refer value-in-exchange as value and value-in-use as utility. Note that we here assume the value is **only** resulted from the satisfaction of a goal. However, one can derive the value of some task or resource by a conjecture over the related goals since they might be essential to the achievement of some goals, but it is still bogus. It is because the execution of a task or the provision of a resource does not always end up in the satisfaction of a goal. We often derive the value of task or resource to analyse the criticality of a task or a resource to the interests of an actor

[1]Goal can be seen as the business objective of an actor. Task is the process performed by an actor to do the business. Resource is the artefact required to perform the task or to achieve the goal.

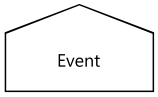


Figure 4.3: Event

Cost is the amount of resources (in broad sense) consumed to achieve *a goal*, to execute *a task*, or to provide *a resource*, Essentially, the cost of a goal is the summation of all costs of tasks and resources required to achieve the goal. Though cost is not only financial matters (e.g., time consumption, learning curve), to simplify the analysis, we urge analysts to quantify the cost in terms of financial units.

4.1.2 Event Layer

In the event layer, we analyse uncertainty that can affect the value layer either positively or negatively. It is the layer where risks are identified and analysed further. The GR framework adopts the view of COSO by modelling risk as an uncertainty with negative impact to the value layer, and an opportunity as the one with positive impact.

Event represents uncertain circumstance that affects the attainment of the value layer. An event is characterised by *likelihood* and *severity* as explained in Section 3.1. Likelihood is represented in terms of *evidence* (i.e., SAT and DEN) as the ones in the value layer, while severity is modelled as the extent of impact relation that relates the event layer and the value layer. The graphical representation of an event is depicted in Figure 4.3, while the impact relation is detailed in Section 4.1.4.

To avoid confusion in identifying events, we here organise events into three classes: failure, threat, and circumstance as depicted in Figure 4.4. *Failure* is defined as a state where the system delivers a services that is deviated from the requirement. It can occur because of a malicious intent - *an attack* or just simply *an accident*. An attack essentially is an executed *threat* that exploits a particular *vulnerability*/fault of the system, such as Denial-of-Service Attack where the attack agent has some motivations to bring the service being unavailable and has some method to realise the threat. An accident occurs without any malicious intent where vulnerability is activated by *an incident* originated from a random process (e.g., radar system down) or a human error (e.g., entering a wrong flight plan). Moreover, risks can also be originated from usual *circumstance* (e.g., air traffic increase) that is neither failure, threat, nor incident.

KAOS has introduced the notion obstacle (undesirable behaviour) [210] and anti-goal (malicious goal) [209]. Essentially, both concepts can be used as the initial point to identify the events, since obstacle is similar with the notion of incident, while anti-goal is similar with the motivation of a threat agent. In the GR framework, all these concepts are boiled down and captured as events that are structured using some relations. We will demonstrate how to do it following a methodological step in Chapter 6. Conventionally, risk management considers risks mainly in the level of artefact (or called assets) which is defined as "something" that is valuable for the organisation (e.g., the radar system is down or the controllers being unavailable). We here argue that there is a situation where the business is disrupted while all assets/artefacts are fully functioning. It might be because the business process is disrupted (e.g., order delivery is late); or there is a state-of-affair that prevents the business objective to be achieved (e.g., new competitors, new regulations). Thus, analysts must analyse all relevant risks for all three levels of the value layer: : objective, process, and artefact.

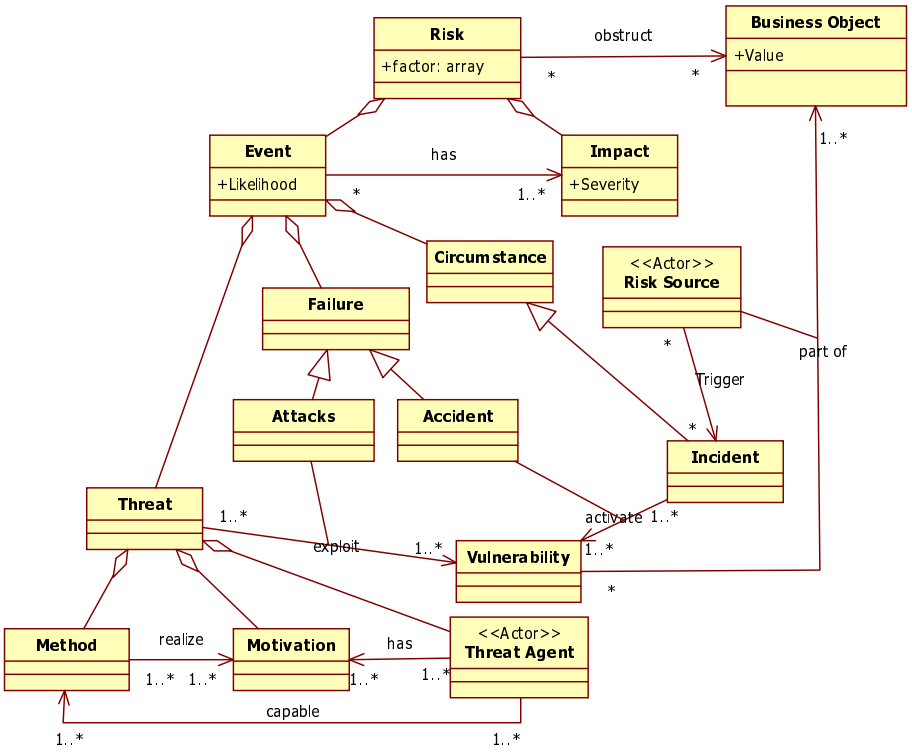


Figure 4.4: Taxonomy of Event

4.1.3 Treatment Layer

We next focus on treatments/countermeasures/mitigation intended to mitigate risks. A treatment may affect on a risk in two different ways: reducing its *likelihood* or attenuating its *severity*. To reduce the likelihood, a treatment is modelled using a contribution relation which introduces denial evidence to the event; and to reduce the severity, it is modelled using an alleviation relation. Both relations will be explained in detail in the following section.

Example 4.2. *To reduce the likelihood of the outage of radar coverage, each airspace needs to be covered with at least two radar systems. In case, both systems are down in the same time, then to alleviate the severity of such a risk ATCOs are ordered to expand the airspace separation among aircraft, and each pilot is obliged to maintain the assigned separation by means of the on-board TCAS (Traffic Collision Avoidance System).*

Essentially, to model treatments we use the same concepts (e.g., task, resource) introduced in the value layer. However, the aim of the execution of such task or the provision of such resource is not for achieving some goal, but mitigating risks that threaten the system and organisation businesses. In fact, this representation enables us to analysis second order risks (i.e., the risk of treatments) and to ensure the treatments are reliable enough.

4.1.4 Relationship among Constructs

Previously, we have explained basic concepts of the GR modelling language. In this section, we introduce several relations that are useful to analyse previous concepts into more details.

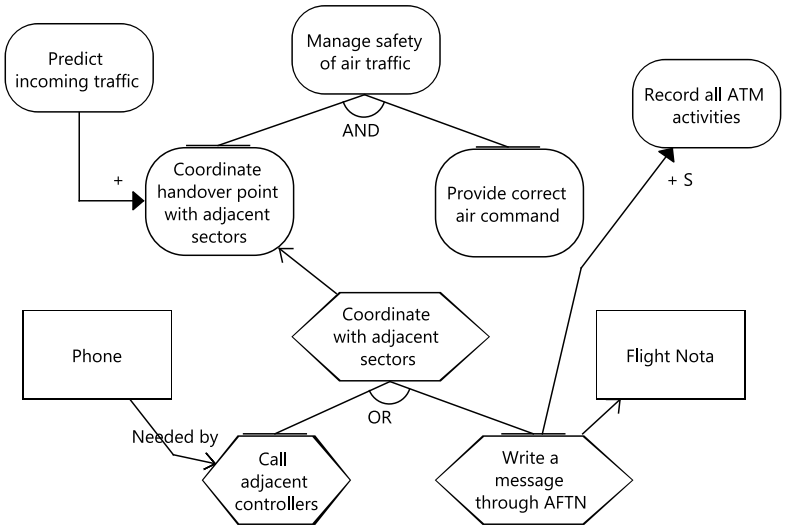


Figure 4.5: An Example of Decomposition, Means-End, Needed-by, and Contribution

AND Decomposition is used to refine a goal into several subgoals. To satisfy the up-level goal, it requires all subgoals to be satisfied.

Example 4.3. *As in Figure 4.5, an executive controller intends to achieve* manage the safety of air traffic. *To achieve this, he must achieve* Coordinate handover point with adjacent sectors ***and*** Provide correct air command.

OR Decomposition is used to capture alternatives to fulfil the up-level goal. To satisfy the up-level goal, it is sufficient by achieving only one of the subgoals. Moreover, both decompositions are also applicable for task, resource, and event to produce more detail structure or to capture alternatives.

Example 4.4. *In Figure 4.5, the task of* Coordinate with adjacent sectors, *controllers can be done by performing* call adjacent controllers ***or*** write a message through AFTN.

Means-End is used to identify which task or resource that provides a means to achieve a goal. It is also used for indicating a task that produces a resource.

Example 4.5. *In Figure 4.5, the task of* coordinate with adjacent sectors *provides a means to achieve the goal* coordinate handover point with adjacent sectors, *while* write a message through AFTN *provides a means to have* flight Nota *that is used to communicate the change on flight plan.*

Needed-by captures a necessary resource that is required to perform a particular task or required by another resource.

Example 4.6. *In Figure 4.5, to perform the task of* call adjacent controllers *it requires the provision of* phone.

Contribution is used to analyse the side-effect of the success/failure of business objects (i.e., goals, tasks, resources) on other business objects. Typically, contribution is used when the relation between concepts is not the consequence of a deliberative action, but rather results from side-effects. Note that the concept of contribution is different from *correlation* in a conditional probability [171], but rather *probable causation* [176]. The extent of a contribution relation can be positive or negative as indicated in the arrow with $+$ or $-$ sign respectively. Moreover, to distinguish whether the effect is delivered due the success of a business object, the failure of a business object, or in any cases we add the sign with "S"(atisfaction), "D"(enial), or none respectively.

Example 4.7. *In Figure 4.5, it indicates if the goal* predict incoming traffic *is fulfilled then it contributes positively by adding the satisfaction evidence of* coordinate handover point with adjacent sectors*, and if it fails then it will add the denial of evidence of the goal. In the case of the goal* record all ATM activities*, it receives supporting evidence if the task* write a message through AFTN *is successfully executed, otherwise the goal does not receive any side-effect from the task.*

Impact indicates of the effect of an event affecting business objects in the value layer. Essentially, an impact of an event is qualitatively distinguished into four types:

- Strong Positive ($++$) - the event (opportunity) occurrence supports *strongly* the fulfilment of a business object;
- Positive ($+$) - the event (opportunity) occurrence supports *fairly* the fulfilment of a business object;
- Negative ($-$) - the event (risk) occurrence prevents *fairly* the fulfilment of a business object;
- Strong Negative ($--$) - the event (risk) occurrence prevents *strongly* the fulfilment of a business object.

Essentially, an impact relation introduces new evidence – DEN for negative relations and SAT for positive relations – to the business object with considering the type of impact relations and the likelihood of events (i.e., computed over SAT and DEN of the events).

Example 4.8. *In Figure 4.6, both events* not up-to-date flight progress strip (FPS) *and* radio communication link disruption *can prevent the achievement of the goal of* provide correct air command*.*

Probable Causality captures the correlation among events in the event layer. Notice one event can trigger another event; or an even inhibits some other event events to occur. It has similar characteristic with the one for contribution relations.

Example 4.9. *In Figure 4.6, the event* bad weather *increases the likelihood of having* radio communication link disruption*, in which at the end increases the risk of the system.*

Effect models the mitigation where the treatment operates by reducing the likelihood of an event

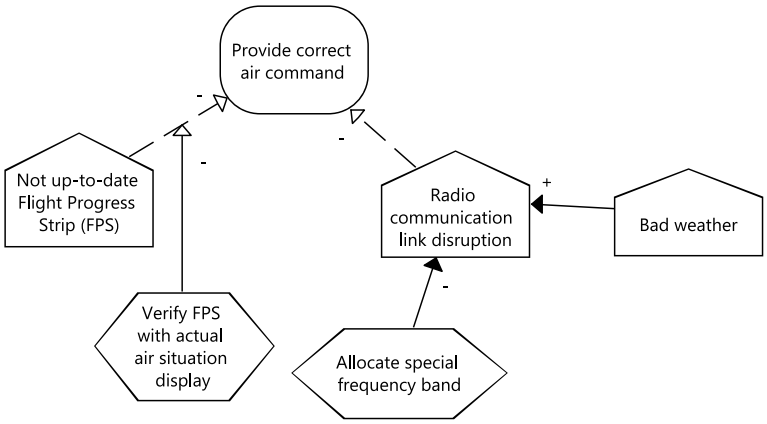


Figure 4.6: An Example of Impact, Probable Causality, Effect, and Alleviation

Alleviation models the mitigation where the treatment operates by reducing the extent of impact relation of an event in obstructing the business object.

Example 4.10. *In Figure 4.6, the treatment of* verify FPS with actual air situation display *aims at reducing the impact of the event* not up-to-date flight progress strip (FPS)*, and the treatment* allocate special frequency band *reduces the likelihood of having* radio communication link disruption*.*

Notice that there are significant similarity behaviours among the concepts of contribution, probable causality and effect. For instance, the behaviour of the effect relation can be mimicked with the "−" contribution, and in principle probable causality and contribution have similar behaviours. Therefore, we here reduce them into a concept of *contribution*.

4.2 Goal-Risk in Multi-Actor Setting

In the previous section, we have presented the core of GR modelling language. This section will extend it to be better suited in dealing with a multi-actor setting. Indeed one may just argue it is sufficient to assess the risk in an organisation by treating it as a single "big" actor. However, this approach fails to understand the domino-effect [166] of a risk to the performance of each unit in the organisation. Moreover, it is hardly possible to use it to perform analysis in outsourcing scenarios, since there is no "big" actor. At last, but not the least, it is critical to have a model that can assess perceived risk, besides actual risk, of each actor in the organisation. Moreover, some scholars prove that people reaction does not always reflect the actual risk [128, 179], but rather their perception. For instance, many studies show that people perceive driving safer than flying [179]. Conversely, the facts show that car crashes cause 42.642 deaths every year (only in US); this is equivalent to have a full-loaded Boeing 727 crashing every 1.5 day (or 225 crashes in a year) [206]. However, according to NTSB report [207] the annual accident-rate of aviation, from 1988 - 2007, is never beyond 0.7 accident per million hours of flight which is so remote comparing with the car crashes.

To enable perceived risk assessment, the GR modelling framework should be able to capture actors' mental-states, which are relevant to compute perceived risk, and the social relationships among actors in the organisation. In the following, we will illustrate the difference between perceived risk and actual risk, and we then introduce some concepts that enable us to modelling them.

4.2.1 Actual Risk vs Perceive Risk

Risk is the expected loss due to a negative event. Based on that value, users make decisions that attempt to minimise or to avoid it. However, some studies [128, 179] shows that on many occasions users behave differently. For instance, people perceive driving safer than flying though it is shown statistically otherwise. Those studies conclude that users behave according to their perception of risk rather than to the actual risk level. Several aspects can influence user perception toward risk, such as trust towards other actors, social norms, and risk/event characteristics (e.g., intentionality, recency, spectacularity) [179, 187]; in addition the way in which risk is communicated can affect the user perception [142].

In [128], the authors study the behaviours of Internet users (i.e., decision to buy via online). In that study, Lacoheea et al. [128] argued that there are strong relations among security, risk, and trust in affecting the behaviours of Internet users (i.e., decision to buy via online). Intuitively we assume that users will buy something if they believe that the risk is acceptable. In fact, the study indicates that most of users decide to order via Internet though the merchant does not have high security measures, but they believe that the merchant is trustworthy. Moreover, it shows also that having restitution policies is more effective than employing heavy security protection in gaining the users' trust. Thus, we can conclude that trust relation affects the perception of users about the risk of buying through Internet (i.e., deduced from their behaviours), and it does not necessarily follow the real value of risk (i.e., likelihood $\times$ severity). From these examples, we conclude the trust of users towards other users or the system is a fundamental aspect for assessing perceived risk.

Moreover, the perception of risk is also closely related with user expectations and preferences. User expectations represent the user feeling that something is about to happen, and user preferences capture the predisposition in favour of something. Different users can have different expectations for the same situation. Perceived risk can be seen as the distance between user expectations and preferences and the risks that users tolerate. For instance, if the expected losses of two events are the same, but Alice tolerates a risk more than user Bob; then Bob perceives the event is more risky than Alice's perception. Expectations also include the opportunities that users have when taking a risk. A concrete example is given by the number of people gambling, though they realise how remote the odd of winning the game is.

Moreover, perceived risk is also related with culture and norms of a particular society [179]. For instance, Indonesian citizens argue that e-Commerce is extremely risky, while EU citizens have a different perception of it. In any case, when a user is on the Internet, he is exposed to the same set of hackers. The different perception of risk between Indonesian and EU citizens is caused by the fact that e-Commerce transactions are not a common means in the Indonesian society. This aspect of perceived risk might also explain why driving is not perceived as dangerous; driving is considered an ordinary activity in almost every society.

Risk management aims to reduce the likelihood of events or their impacts within an acceptable level. This approach however is not suitable to manage perceived risk; users might behave carelessly because they know that they are exposed to risks remotely or because they seek opportunities. The aim of perceived risk management is to correct the user perception, advising users when their perception diverges from actual risk. To enable such modelling, in the next sections we introduce the concepts of actor, social relation among actors in the system, and mental-state of the actor.

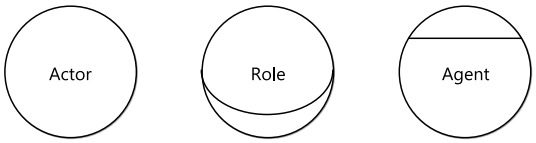


Figure 4.7: Actor, Agent, Role, and their relations

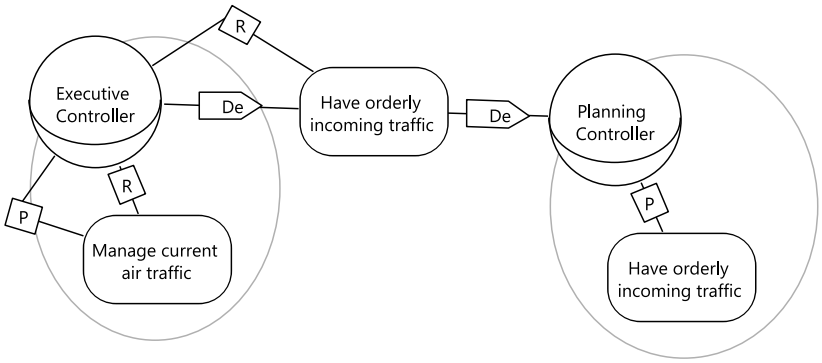


Figure 4.8: A Simple GR model in Multi-Actor Setting

4.2.2 Actor, Role, and Agent

Actor is an autonomous entity that has its own intentions (probably delegated from another actor). An actor can be a software agent [96] since it has an intention which is originated from some human agents (i.e., users, stakeholders). Actually, actor covers two basic concepts: **role** and **agent** as depicted in Figure 4.7.

Agent is an autonomous entity with a concrete manifestation in the system. It covers humans (e.g., stakeholders, users) as well as software agents and organisations.

Role is an abstract representation, in terms of a set of behaviours and functionalities, of an active entity within a particular setting.

In nutshell, role can be seen as a class and agent as its instantiation which plays a single role or more. Typically, a design is done at the role level, and in runtime agents operate to achieve their intentions. An agent (or a role) can be part of another agent (or another role) as an individual is a part of a group.

Previously, goal is defined as a state of affairs that an actor intends to be satisfied. In a multi-actor setting, we refine such definition following the notion of *goal* in SI* [230] where the actor, which is interested in the goal, can be different from the one that is capable to fulfil it.[2].

Example 4.11. *Based on the example 4.1, In the rationale of an executive controller* [3] *he is interested on the achievement of* have orderly incoming traffic *and* manage current air traffic*, while a planning controller is able to achieve the goal of* have orderly incoming traffic *even (assuming) it is not her interest as depicted in Figure 4.8.*

[2]The study about *entitlement* is out of the scope of this thesis

[3]To avoid a gender debate, this work assume the executive controller is a male, and the planning controller is a female

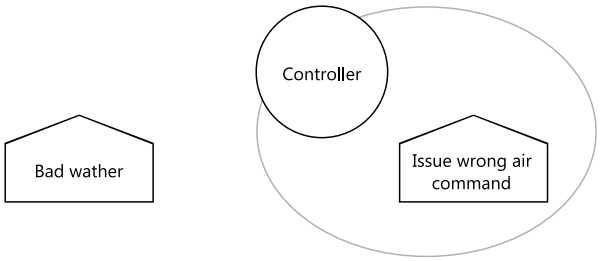


Figure 4.9: Local Event and Global Event

Adopting from SI*, an actor has desires and capabilities that are modelled as relations between the actor and a goal/a task/a resource.

Request indicates the desire of an actor for the achievement of a goal, the execution of a task, or the provision of a resource.

Provide indicates the capability of an actor in achieving a goal, executing a task, or provisioning a resource.

In graphical diagrams, both relations are represented as edges between an actor and a goal, a task, or a resource labelled with "R" and "P" respectively as in Figure 4.8. The distinction between desire and capability enables us in modelling which actors are at risk (i.e., risk owner) because they desire something that they are incapable of (e.g., Executive Controllers). In addition, actors can be called as risk source because they are potentially malicious (either have intentions or even means) or simply provide faulty/vulnerable business objects. An actor that is capable to mitigate risks are called risk controller.

In a GR model, each business objects should be felt on some actors' rationale either as their desires or capabilities. However, it is not always the true for events because there is an event which is not concerning particular actors. Therefore, in Figure 4.9 we distinguish between *global event* (e.g., bad weather), which is independent from any actors, and *local event* (e.g., issue wrong air command), which is related to particular actors because its likelihood might be vary depending on the actors/agents.

4.2.3 Social Relation

Dependency (or delegation on execution in SI*) is a relation between two actors, which denotes that an actor (the *depender*) depends on another actor (the *dependee*) for the attainment of a particular business object (the *dependum*). Intuitively, dependency relations bound the dependum evidence in the depender's rationale to the evidence from the dependee.

Example 4.12. *In Figure 4.8, an executive controller depends on a planning controller to satisfy the goal* have orderly incoming traffic*, because she has capability to achieve the goal.*

By delegating the fulfilment of a particular business objects, the delegator then becomes vulnerable because, if the delegatee fails to fulfil the assigned responsibilities, the delegator will not be able to achieve his/her objectives. Thereby, such a situation introduces risks that can decrease the dependability of the system.

Trust captures an actor's (the trustor) belief that another actor (the trustee) behaves according to an agreement (the trustum). The trustum here refers to the attainment of business objects. Together with the notion of trust, we adopt also the notion of *distrust* [82]. This relation is used to model the belief of an actor about the misbehaviour of another actor in fulfilling the trustum. Trust is characterised by a level which represents the degree of trust between two actors. According to [47, 183] trust can be seen as the mental counterpart of dependency; dependency is an action due to a decision, whereas trust is a mental state driving such a decision. Therefore, trust is normally necessary for delegation. However, there can be situations where an actor has to assign responsibilities to untrusted actors since s/he has no alternatives or an authority has prescribed to do so [73]. In fact, in [12] we underline the fact that actors are expose to risks because they depend on other actors for accomplish some task, and not because of trust. In other words, there is no harm in trusting someone.

Many researchers have investigated the relationships between trust and risk. Jøsang and Presti, in [114], notice that trust and risk drive the decision-making process. Risk is used as a measure of reliability trust (i.e., the probability of having a successful transaction); reliability trust together with the risk attitude of a user supports the user in making (trust) decisions. Solhaug et al., in [191], emphasise that risk cannot be inferred from trust because trust does not include aspects concerning the impact in case the trustee betrays. These proposals, however, attempt to assess actual risk. For instance, if Bob depends on Alice for some tasks, the actual risk is the same whether or not Bob trusts Alice. What changes is Bob's perception of risk: if Bob trusts Alice, he believes that Alice carries out the task regardless of the reality. In this work, we have identified two types of trust relevant to perceived risk: trust in execution and trust in evidence. *Trust in execution* represents the trustor belief of the capability and dependability of the trustee in fulfilling the trustum. *Trust in evidence* represents the trustor belief of the evidence provided by the trustee about the trustum.

Example 4.13. *If Luke (an execution controller) trusts (in execution) Paula (a planning controller) on* manage incoming traffic*, he perceives that incoming air traffic will be ordered properly by Paula. In the case of Robert (a sector supervisor) trusts in evidence provided by Paula about her performance in* manage incoming traffic*, then Robert's perception of risk depends on the evidence provided by Paula.*

In the first case, the perceived risk of Luke is deduced independently from the risks suffered by Paula. In the second case, if Paula indicates the traffic increase is beyond her capability (i.e., she feels at risk), then Robert's perceived risk increases; while Luke's perceived risk is remain the same.[4] The graphical representation of these social relations are presented in Figure 4.10, which dependency, trust on execution, and trust on evidence are depicted using relations with a label "De", "Te", and "Tev" respectively. In the case of distrust relations, we use "Se" to denote distrust on execution and "Sev" for distrust on evidence.

4.2.4 Mental State

Utility Function quantifies the value generated by business objects. Conceptually, the notions of utility and value are different [23]: the utility of business objects in an actor's perspective is assessed by summing up all the values generated for the actor, and it is not the case for the notion of value. In other words, a resource has the same value for two actors, but it might have different utility depending on the actors' rationale.

Example 4.14. *The application* airspace modeller *is valuable for the performance of ATM. The application costs (value-in-exchange) the same for any actors in the Air Traffic Control System. However,*

[4] We assume that the indication from Paula does not modify Luke's trusts toward Paula.

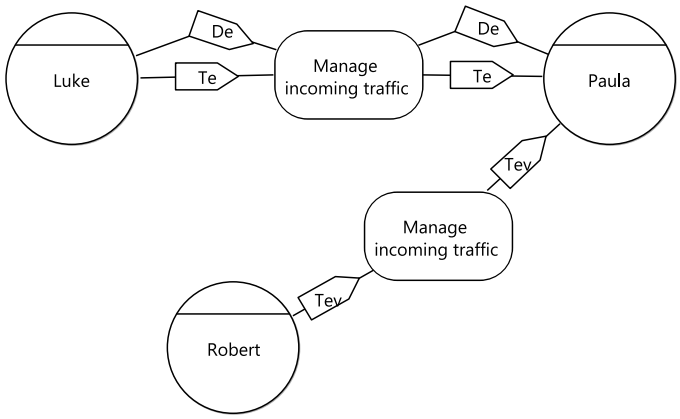


Figure 4.10: Social Relations: Dependency, Trust on Execution, and Trust on Evidence

it is considered being more valuable for sector supervisors compared with executive controllers because it is useful to assists the supervisors in defining the sector configuration (i.e., one of the critical task). In other words, the utility of the application is higher in the supervisors' rationale than in the executive controller's one.

We here model *loss* as the reduction of utility [15]. As consequence, for the same pair of event-business object an actor might perceive different expected loss from another actor. In fact, some works specify loss as a value function of an asset/a business object [65] or defined by experts' judgement [218]. However, they still do not solve such issues, because both approaches do not consider in which actor's rationale the loss is perceived.

Risk Tolerance is the expected loss acceptable by users. As mentioned before, perceived risk is not an absolute value; it is defined as the distance between the risk tolerance and the expected loss.

Example 4.15. *Two actors – Alice and Bob – expose to the risk of losing 2K on a transaction. The income of the two users is different, let say, Alice earns 100K/year and Bob 50K/year. Assuming that the income corresponds with the level of risk tolerance, Bob can argue that a transaction is risky, while Alice is more willing to accept the risk.*

Risk tolerance can be used to represent user expectations and preferences. Moreover, actors tend to be more risk tolerant for the matters that they prefer, but they are sensitive to the ones related to their expectation.

4.3 Summary

In this chapter, we have introduced the basic concepts of the Tropos Goal-Risk modelling language for modelling and analysing risks in an organisation/a single actor or a multi-actor setting. All of those concepts are structured in terms of a meta-model depicted in Figure 4.11 and in particular Figure 4.12 for social relations.

In Figure 4.11, the model presents the basic constructs of the GR modelling language, such as: business objects (i.e., goals, tasks, and resources), events, and relations among concepts (e.g., decomposition, contribution, alleviation, means-end, needed-by). Moreover, impact is considered as a special relation

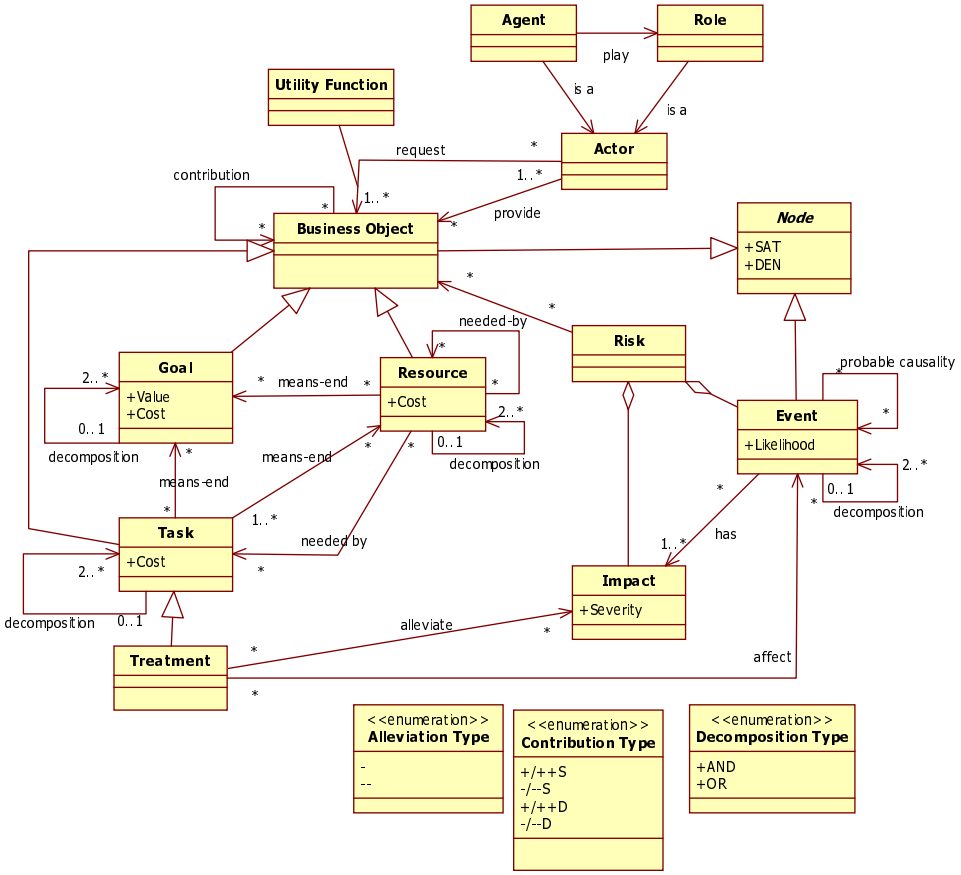


Figure 4.11: Metamodel of the Goal-Risk Modelling Language

because it can be a relation between an event and a business object, or it can act as a target node in the case of alleviation relation (as illustrated in Figure 4.5). In addition, those constructs have some properties characterising them. For instance, goal, task, resource, and event have evidence values (i.e. SAT and DEN). Some relations (e.g., contribution, alleviation, decomposition) have several subtypes as indicated in Figure 4.11.

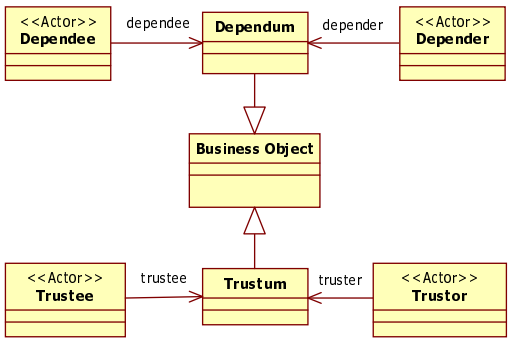


Figure 4.12: Metamodel of Social Relations in GR

To capture social relation among actors, the GR modelling framework adopts the concept of *dependency* from *i** [226] and *trust* from SI* [230]. In Figure 4.12, we present the metamodel of both relations. However, the entitlement/permission aspect in SI*[230] is out of the scope of the thesis. Indeed giving permission (i.e., delegation of permission) may introduce a new source of risk, but it is not always the case. For instance, Alice gives permission to Bob to use her PC to finish the group assignment, then the conclusion whether Alice is at risk or not is too premature. Unlike the dependency (delegation of execution), the fact of Alice delegates the permission to Bob is insufficient to decide whether it will put Alice at risk or not. Though Bob misuses the permission by using the PC not according to the purpose in which Alice agrees upon (i.e., finish the group assignment), it is still not necessarily a risk for Alice. Bob may use the PC for checking his mail account which does nothing harms for Alice. Therefore, we consider the need to study in depth the characteristics of entitlement.

Chapter 5

Risk Analysis

Several approaches have been proposed in the literature to analyse and assess risk against others CORAS [64], COSO-ERM [58], and *i**-based [141]). Models are used to capture and represent risk-related concepts that affect the system's performances. Unfortunately, the lack of a formal representation makes hard to perform automated analyses even for the tedious ones (e.g., compute the risk level, sum the total cost).

In this chapter, we present the underlying formal model of our approach aiming at enabling automated analysis techniques. *Forward Analysis* (adapted from [80]) is proposed to compute the risk level of an organisation (Section 5.3). To make the analysis is generic (i.e., able to assess actual risk and perceived risk), a GR model must be pre-processed before performing forward analysis (Section 5.2). Another analysis technique, namely *Backward Analysis* (adapted from [181]) intends to elicit possible treatments that can mitigate the risk up to an acceptable level (Section 5.4). Finally, we conclude with a summary (Section 5.5).

5.1 Tropos Goal-Risk Model as a Formal Model

We define a GR model mathematically as a quadruplet $\langle \mathcal{N}, \mathcal{R}, \mathcal{I}, \mathcal{S} \rangle$ where $\mathcal{N}$ is a set of nodes of business object, $\mathcal{R}$ is a set of relations relating between nodes, $\mathcal{I}$ is a set of special relations relating the value layer and the event layer, and finally $\mathcal{S}$ captures social relationship between actors in the organisation.[1]

5.1.1 Nodes - $\mathcal{N}$

Definition 5.1. $\mathcal{N}$ is a set of nodes which is the union of *global events* $\mathcal{E}_G$[2] and actors' rationale or perspective $\mathcal{P}$, which is defined a set of tuples of (a, o, p), where:

- $a \in \mathcal{A}$, where $\mathcal{A}$ is the set of *actors* participating in the organisation/system;
- $o \in \mathcal{O} \cup \mathcal{E}_L$, where $\mathcal{O}$ is a union of business objects (i.e., *goals* $\mathcal{G}$, *tasks* $\mathcal{T}$, *resource* $\mathcal{M}$) and *local events* $\mathcal{E}_L$.
- $p \subset \{request,\ provide\}$ is the relation stating whether o is the actor's desire, capability, or none (i.e., in the case of $o \in \mathcal{E}_L$).

[1] One can consider a GR model as $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$ if the analysis is done for a single actor

[2] Formally, $\mathcal{E}_G$ is denoted as $\{(a, e, p) | e \in \mathcal{E}_G \wedge a = nil \wedge p = \emptyset\}$ which is not part of any actors' perspective

Each node has two attributes - SAT- $Sat(N)$ and DEN- $Den(N)$ - representing the value of evidence that such node N will be satisfied or denied. It is based on the idea of the Dempster-Shafer (DS) theory of evidence [182] where the evidence of a goal (or other constructs) being denied (DEN) cannot be inferred from evidence on the satisfaction of the goal (SAT), and vice versa. Unlike in probability theory, if $Prob(A) = 0.1$ then we can infer that probability of $\neg A$ is 0.9 (i.e., $P(\neg A) = 1 - P(A)$). For instance, Air Traffic Management (ATM) system has strong evidence (let say 0.9) that the goal manage air traffic safely will be satisfied because it has qualified controllers and a reliable Air Traffic Control (ATC) system. However, we cannot infer how much the DEN of this goal because there is no other information which says so.

We here define qualitatively the value in the range of $\{(F)ull, (P)artial, (N)one\}$. Suppose N - (a, o, p) - is a goal ($o \in \mathcal{G}$), the value attribute is indicated as a node label and is represented by 6 different satisfaction predicates:[3]

- $FS(a, o)$ $(FD(a, o))$ - Actor a believes there is (at least) *full*/sufficient evidence to support the claim that goal o will be satisfied (or denied);
- $PS(a, o)$ $(PD(a, o))$ - Actor a believes there is (at least) *partial*/some evidence to support the claim that goal o will be satisfied (or denied);
- $NS(a, o)$ $(ND(a, o))$ - Actor a believes there is *no* evidence to support the claim that goal o will be satisfied (or denied). They are the same with T predicate in [80]. It is not mandatory to write these predicates in formalisation. They could be left implicitly.

The predicates state that there is *at least* a given level of evidence that the goal is satisfied (or denied), and we assume that $full \geq partial \geq less$, with the intended meaning $x \geq y \leftrightarrow x \rightarrow y$. Hence, we can write the following invariants:

Node	Invariant Axioms		
(a, o, p) :	$FS(a, o) \rightarrow PS(a, o) \rightarrow NS(a, o)$	where $o \notin \{\mathcal{E}_L \cup \mathcal{E}_G\}$	(5.1)
	$FD(a, o) \rightarrow PD(a, o) \rightarrow ND(a, o)$		(5.2)

Axiom (5.1)-(5.2) formalise the invariant of the value of SAT and DEN. If a node has (at least) *full* evidence of satisfaction (or denial), then it consequently has also (at least) *partial* evidence of satisfaction (or denial). Two actors can have two different level of evidence for the same goal.

5.1.2 Impact - $\mathcal{I}$

Definition 5.2. $\mathcal{I} \subseteq \mathcal{E} \times \mathcal{N}_2$ where

- $\mathcal{E} = \{(a_1, o_1, p_1) \in \mathcal{N} | o_1 \in \mathcal{E}_L \cup \mathcal{E}_G\}$;
- $\mathcal{N}_2 = \{(a_2, o_2, p_2) \in \mathcal{N} | o_2 \in \mathcal{O}\}$.

`Impact`(e.g., $E \overset{x_i}{\longmapsto} N$) indicates the severity of events E ($\mathcal{E}_L$ and $\mathcal{E}_G$) impacting the attainment of a business object - N . The relations are categorised into several types $x_i \in \{++_i, +_i, --_i, -_i\}$.

[3] The similar principle is also applied for a task to be executed, a resource to be provided, or an event to occur

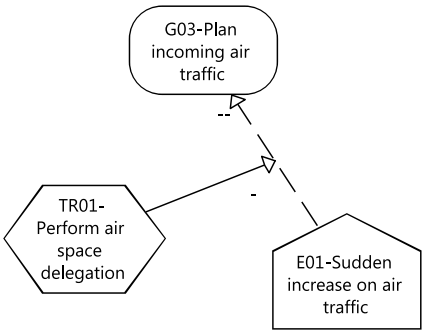


Figure 5.1: Impact and Alleviation in ATM scenario

$Sat(E) \wedge Den(E) \longmapsto \lambda(E)$		
$Sat(E)$	$Den(E)$	$\lambda(E)$
F	N	L
F	P	O
P	N	O
F	F	R
P	F	R
P	P	R
N	F/P/N	U

Table 5.1: Likelihood Calculation based on Evidence Values

Essentially, the negative relations add DEN evidence of N (i.e., risk), and the positive ones increase SAT evidence of N (i.e., opportunity). Impact relations are considered special because they act as relations between events and business objects, but it also behaves like a target node for alleviation relations. Since the risk level is defined by its likelihood ($\lambda(E)$) and its severity (x_i), we need to consider x_i and SAT-DEN (to calculate $\lambda(E)$) before determining the value of evidence propagated to N. The likelihood of E are qualitatively categorised into 4 levels: *(L)ikely, (O)ccasional, (R)are*, and *(U)nlikely*, with intended meaning $L > O > R > U$. They are represented in terms of four predicates: $Unlikely(E)$, $Occasional(E)$, $Rare(E)$, $Unlikely(E)$, respectively. Table 5.1 defines the calculation rules of likelihood from SAT and DEN values; and given the likelihood and the extent of impact, N receives additional evidence based on the propagation rules in Axiom (5.3)-(5.18).

Relation	Relation	Axioms	
$E \overset{++_i}{\longmapsto} N:$	$Likely(a,e)$	$\rightarrow FS(a,o)$	(5.3)
	$Occasional(a,e)$	$\rightarrow PS(a,o)$	(5.4)
	$Rare(a,e)$	$\rightarrow PS(a,o)$	(5.5)
	$Unlikely(a,e)$	$\rightarrow NS(a,o)$	(5.6)
$E \overset{+_i}{\longmapsto} N:$	$Likely(a,e)$	$\rightarrow PS(a,o)$	(5.7)
	$Occasional(a,e)$	$\rightarrow PS(a,o)$	(5.8)
	$Rare(a,e)$	$\rightarrow NS(a,o)$	(5.9)
	$Unlikely(a,e)$	$\rightarrow NS(a,o)$	(5.10)
$E \overset{--_i}{\longmapsto} N:$	$Likely(a,e)$	$\rightarrow FD(a,o)$	(5.11)
	$Occasional(a,e)$	$\rightarrow PD(a,o)$	(5.12)
	$Rare(a,e)$	$\rightarrow PD(a,o)$	(5.13)
	$Unlikely(a,e)$	$\rightarrow ND(a,o)$	(5.14)
$E \overset{-_i}{\longmapsto} N:$	$Likely(a,e)$	$\rightarrow PD(a,o)$	(5.15)
	$Occasional(a,e)$	$\rightarrow PD(a,o)$	(5.16)
	$Rare(a,e)$	$\rightarrow ND(a,o)$	(5.17)
	$Unlikely(a,e)$	$\rightarrow ND(a,o)$	(5.18)

Let's assume, in Figure 5.1 (i.e., $E_{01} \overset{--_i}{\longmapsto} G_{03}$), there is strong (i.e., full) evidence for sudden increase on air traffic *(E_{01})* due to the summer holiday, and *no* evidence of denial. Based on the Table 5.1, E_{01} is considered as a *likely* event, and consequently E_{01} will deliver *full* evidence of denial to the goal plan incoming air traffic *(G_{03})*. In addition, an event without any evidence of satisfaction is considered as an *unlikely* event regardless from the value of denial evidence. Such events do not introduce any new evidence to the business object regardless of the severity of the impact relations.

5.1.3 Relations - $\mathcal{R}$

Definition 5.3. $\mathcal{R}$ is the set of relationship among nodes in a GR model represented as $(N_1, \ldots, N_n) \overset{r}{\longmapsto} N$, where $N_1, \ldots, N_n$ are called *source nodes* and N is the *target node* in which $\{N_1, \ldots, N_n, N\} \in \mathcal{N}$which are related with the r relation (i.e., $\texttt{Dec}_{and}$, $\texttt{Dec}_{or}$, `Contr`, `Means-end`, and `Needed-by`).

In general, the evidence is propagated from source nodes to the target node depending on the type of relation (r) and the evidence value of the source nodes.

Definition 5.4. $\texttt{Dec} \subseteq 2^{\mathcal{N}} \times \mathcal{N}$ is the set of *decomposition* relations. There are two types of decomposition relation: AND and OR decomposition. Notice that upper level (target) nodes must have the same type of subnodes (source). Moreover, decomposition relations are *intra-actor*, that is, the target and source nodes must be in the rationale of the same actor.[4]

Definition 5.5. $\texttt{Dec}_{and}$ (e.g., $(N_2, N_3) \overset{and}{\longmapsto} N_1$) indicates the refinement of N_1 into several subnodes- N_2 and N_3 . To attain the upper level node, one must be able to satisfy all the subnodes, and consequently, the failure of one of the subnode results in the failure of the upper level node.

[4]If a target node is a global event, then all source events should be also global events.

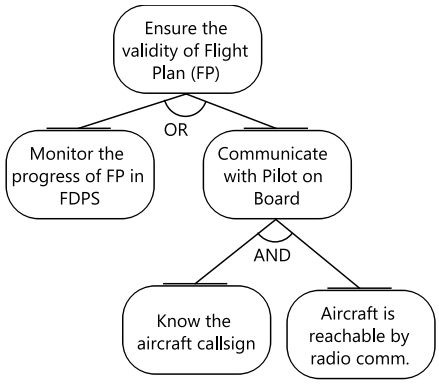


Figure 5.2: Decomposition in ATM scenario

Axiom (5.19)-(5.24) define how SAT and DEN evidence of nodes are calculated on the basis of the evidence of their AND-subnodes. Since the upper level will be satisfied if and only if all subnodes are satisfied, we assume that the SAT evidence of an upper level node is driven by the weakest subnode (i.e., the subnode with the lowest SAT value) as defined in Axiom (5.19)-(5.21). Conversely, the DEN evidence of the upper level node follows the worst subnode (i.e., the subnode with the highest DEN value) as formalised in Axiom (5.22)-(5.24), because the failure of a subnode is sufficient to result in the failure of the upper level node.

Relation	Relation Axioms	
$(N_2, N_3) \overset{and}{\longmapsto} N_1$:	$FS(a, o_2) \wedge FS(a, o_3) \rightarrow FS(a, o_1)$	(5.19)
	$PS(a, o_2) \wedge PS(a, o_3) \rightarrow PS(a, o_1)$	(5.20)
	$NS(a, o_2) \wedge NS(a, o_3) \rightarrow NS(a, o_1)$	(5.21)
	$FD(a, o_2) \vee FD(a, o_3) \rightarrow FD(a, o_1)$	(5.22)
	$PD(a, o_2) \vee PD(a, o_3) \rightarrow PD(a, o_1)$	(5.23)
	$ND(a, o_2) \vee ND(a, o_3) \rightarrow ND(a, o_1)$	(5.24)

For instance, in Figure 5.2 communicate with pilot on-board is AND-decomposed into subgoals Know the aircraft call sign and Aircraft is reachable by radio comm . To satisfy the upper level goal, one must fulfil all these subgoals.

Definition 5.6. $\mathtt{Dec}_{or}$ (e.g., $(N_2, N_3) \overset{or}{\longmapsto} N_1$) refines N_1 into several subnodes- N_2 and N_3 . It is enough to satisfy one of the subnodes for the achievement of N_1 , and consequently the failure of N_1 only occurs if all of the subgoals fail.

Axiom (5.25)-(5.30) define how SAT and DEN evidence of nodes are calculated on the basis of the evidence of their OR-subnodes. On the contrary with AND-decomposition, the SAT evidence of an upper level node here follows the highest SAT evidence of its subnodes (Axiom (5.25)-(5.27)), whereas the DEN evidence follows the lowest DEN values (Axiom (5.28)-(5.30)).

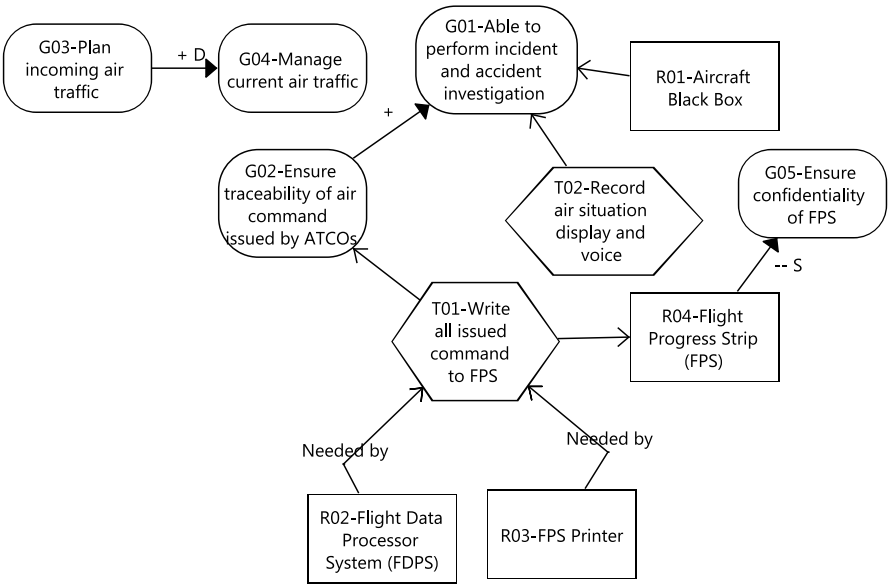


Figure 5.3: Contribution, Means-End, and Needed-by in ATM scenario

Relation	Relation Axioms	
$(N_2, N_3) \overset{or}{\longmapsto} N_1$:	$FS(a, o_2) \vee FS(a, o_3) \rightarrow FS(a, o_1)$	(5.25)
	$PS(a, o_2) \vee PS(a, o_3) \rightarrow PS(a, o_1)$	(5.26)
	$NS(a, o_2) \vee NS(a, o_3) \rightarrow NS(a, o_1)$	(5.27)
	$FD(a, o_2) \wedge FD(a, o_3) \rightarrow FD(a, o_1)$	(5.28)
	$PD(a, o_2) \wedge PD(a, o_3) \rightarrow PD(a, o_1)$	(5.29)
	$ND(a, o_2) \wedge ND(a, o_3) \rightarrow ND(a, o_1)$	(5.30)

For instance, in Figure 5.2 ensure the validity of Flight Plan (FP) is OR-decomposed into subgoals monitor the progress of FP in FDPS or Communicate with Pilot on Board . To satisfy the upper level goal, it is sufficient by fulfilling one of the subgoals.

Definition 5.7. `Means-end`$\subseteq \mathcal{N}_1 \times \mathcal{N}_2$ where

- $\mathcal{N}_1 = \{(a_1, o_1, p_1) \in \mathcal{N} | o_1 \in \mathcal{G} \cup \mathcal{M}\}$;
- $\mathcal{N}_2 = \{(a_2, o_2, p_2) \in \mathcal{N} | o_2 \in \mathcal{T} \cup \mathcal{M}\}$;
- **not** $o_1, o_2 \in \mathcal{M}$;
- $a_1 = a_2$.

`Means-end`(e.g., $N_2 \longmapsto N_1$) indicates N_2 is a means to attain N_1 . It captures the task or the resource used to satisfy goals; or to indicate the task that produces a resource. Moreover, means-end relation is an intra-actor relation.

This relation behaves similarly with a causation relation [176] or an implication relation. If there are more than one means (i.e., N_2 , N_3) to achieve the end (N_1), then their behaviour is similar to $\mathtt{Dec}_{or}$ (i.e., $N_2 \longmapsto N_1$,$N_3 \longmapsto N_1 \leftrightarrow (N_2, N_3) \overset{or}{\longmapsto} N_1$)

For instance, in Figure 5.3 the execution of write all issued command to FPS *(T_{04})* is a means for providing Flight Progress Strip (FPS) *(R_{02})*, hence the evidence of R_{02} follows the evidence of T_{04} (the same as axiom (5.33)-(5.36)). However, in the case of there are two ways to achieve G_{01} (i.e., $T_{02} \longmapsto G_{01}$,$R_{01} \longmapsto G_{01}$), this setting is equivalent with $(T_{02}, R_{01}) \overset{or}{\longmapsto} G_{01}$ and the propagation rules are governed by axiom (5.25)-(5.30).

Definition 5.8. `Needed-by`$\subseteq \mathcal{N}_1 \times \mathcal{N}_2$ where

- $\mathcal{N}_1 = \{(a_1, o_1, p_1) \in \mathcal{N} | o_1 \in \mathcal{M}\}$;
- $\mathcal{N}_2 = \{(a_2, o_2, p_2) \in \mathcal{N} | o_2 \in \mathcal{T} \cup \mathcal{M}\}$;
- $a_1 = a_2$.

`Needed-by`(e.g., $N_2 \overset{n}{\longmapsto} N_1$) indicates N_2 is needed to attain N_1 . It is used to indicate the resource that is required for executing a task or required by another resource, and it is an intra-actor relation.

This relation seems to be the inverse of `Means-end`, though it has some particularities. For instance, if there are more than one resources (i.e., N_2 , N_3) required to execute/to provide (N_1), then it behaves similarly to $\texttt{Dec}_{and}$ (i.e., $N_2 \overset{n}{\longmapsto} N_1$,$N_3 \overset{n}{\longmapsto} N_1 \leftrightarrow (N_2, N_3) \overset{and}{\longmapsto} N_1$)

For instance, in Figure 5.3 the execution of write all issued command to FPS *(T_{01})* requires Flight Progress Strip (FPS) *(R_{02})* and FPS printer *(R_{03})* (i.e., $R_{02} \overset{n}{\longmapsto} T_{01}$,$R_{03} \overset{n}{\longmapsto} T_{01}$) which is equivalent with $(R_{02}, R_{03}) \overset{and}{\longmapsto} T_{01}$ and the propagation rules are governed by Axiom (5.19)-(5.24).

Definition 5.9. `Contr`$\subseteq \mathcal{N}_1 \times \mathcal{N}_2$ where

- $\mathcal{N}_1 = \{(a_1, o_1, p_1) \in \mathcal{N} | o_1 \notin \mathcal{E}_L \cup \mathcal{E}_G\}$;
- $\mathcal{N}_2 = \{(a_2, o_2, p_2) \in \mathcal{N}\}$.

$\texttt{Contr}_x$ (e.g., $N_2 \overset{x}{\longmapsto} N_1$) models the side-effect of the attainment/failure of a node- N_2 to another node- N_1 where $x \in \{+_S, ++_S, -_S, --_S, +_D, ++_D, -_D, --_D\}$ [5].

The type x represents the influence of a source node on the target node, such as $++_S$ propagates only SAT evidence, or $++_D$ propagates only DEN evidence. Moreover, the sign $x = ++$ is the shortcut for $++_S$ and $++_D$. `Contr` can be either an intra-actor or inter-actor relation since the fulfilment of an actor's goal can affect positively/negatively the fulfilment of a goal (or the execution of a task, or the provision of a resource, or the occurrence of an event) in another actor's rationale.

These relations are also used to model the effect of treatments (depicted as tasks) on the likelihood of events. These relations are not applicable only if the source node is an event and the target node is a business object. For the last situation, we model such a situation using *impact* relations because the evidence propagation is governed by the likelihood, instead of evidence values. Moreover, the relation from events to business object must be able to act as a target node to capture the severity reduction of risk.

As an intra-actor relation, both the source and the target node are laid in the same actor, whereas as an inter-actor relation, both nodes are in different actors' rationale. Axiom (5.31)-(5.44) defines the formalisation of contribution relations from the source node to the target node. The axioms specify that the evidence that an actor (a_2) has on the satisfaction or denial of a node (o_2) affects the evidence that

[5] The sign $+$ means the relation propagates both evidence (i.e., $+_S$ and $+_D$)

another actor (a_1) has on the satisfaction or denial of its node (o_1). In the case of an intra-actor relation, a_2 and a_1 are the same actor. In particular, Axiom (5.31)-(5.32) define that a node that has *no* evidence does not affect the SAT or DEN evidence of another node. Axiom (5.33)-(5.44) propagate SAT or DEN evidence from the source node to the target node according to the type of contribution the evidence of the source node.

Relation	Relation Axioms	
$N_2 \overset{x}{\longmapsto} N_1$[6] :	$NS(a_2, o_2) \rightarrow NS(a_1, o_1)$	(5.31)
	$ND(a_2, o_2) \rightarrow ND(a_1, o_1)$	(5.32)
$N_2 \overset{++_S}{\longmapsto} N_1$:	$FS(a_2, o_2) \rightarrow FS(a_1, o_1)$	(5.33)
	$PS(a_2, o_2) \rightarrow PS(a_1, o_1)$	(5.34)
$N_2 \overset{++_D}{\longmapsto} N_1$:	$FD(a_2, o_2) \rightarrow FD(a_1, o_1)$	(5.35)
	$PD(a_2, o_2) \rightarrow PD(a_1, o_1)$	(5.36)
$N_2 \overset{+_S}{\longmapsto} N_1$:	$PS(a_2, o_2) \rightarrow PS(a_1, o_1)$	(5.37)
$N_2 \overset{+_D}{\longmapsto} N_1$:	$PD(a_2, o_2) \rightarrow PD(a_1, o_1)$	(5.38)
$N_2 \overset{--_S}{\longmapsto} N_1$:	$FS(a_2, o_2) \rightarrow FD(a_1, o_1)$	(5.39)
	$PS(a_2, o_2) \rightarrow PD(a_1, o_1)$	(5.40)
$N_2 \overset{--_D}{\longmapsto} N_1$:	$FD(a_2, o_2) \rightarrow FS(a_1, o_1)$	(5.41)
	$PD(a_2, o_2) \rightarrow PS(a_1, o_1)$	(5.42)
$N_2 \overset{-_S}{\longmapsto} N_1$:	$PS(a_2, o_2) \rightarrow PD(a_1, o_1)$	(5.43)
$N_2 \overset{-_D}{\longmapsto} N_1$:	$PD(a_2, o_2) \rightarrow PS(a_1, o_1)$	(5.44)

In Figure 5.3, the achievement of the goal ensure traceability of air command issued by ATCOs *(G_{02})* contributes positively to the goal able to perform incident and accident investigation *(G_{01})*. It indicates that (at most) *partial* evidence of SAT of G_{02} is propagated to the SAT of G_{01} , and the same for DEN. In the case of "$--_S$", the SAT evidence of flight Progress Strip (FPS) *(R_{04})* is counted as DEN evidence to ensure confidentiality of FPS *(G_{05})*, but the relation does propagate the DEN of R_{04} . It is because the existence of FPS might compromise the confidentiality of FPS, but the non-existence of FPS does not imply that the confidentiality of FPS is preserved. In other words, the provision of R_{01} adds the denial evidence of G_{05} , but the denial of R_{01} does not affect G_{05} . Another example, the evidence of plan incoming air traffic *(G_{03})* being denial supports the DEN of manage current air traffic *(G_{04})* up to *partial* level. However, the SAT evidence of G_{03} could not be counted as supporting evidence for the SAT of G_{04} , because the satisfaction of G_{03} closely depends on the performance of executive controllers (ECs) and their supporting artefacts.

Definition 5.10. `Alleviate`$\subseteq \mathcal{N}_1 \times \mathcal{I}^-$ where

- $\mathcal{N}_1 = \{(a_1, o_1, p_1) \in \mathcal{N} | o_1 \in \mathcal{T}\}$;
- $\mathcal{I}^- \subseteq \mathcal{I}$.

`Alleviate`(i.e., $TR_1 \overset{x_a}{\longmapsto} [I]$) denotes a treatment - TR_1 mitigating the (negative) impact ($\mathcal{I}^-$) of events (i.e., $I = E \overset{x_i}{\longmapsto} N$). The extent of mitigation (x_a) is categorised into two levels: $--$ and $-$.

[6] $x \in \{++_S, +_S, --_S, -_S, ++_D, +_D, --_D, -_D\}$; A_1 and A_2 might be the same actor or two different actors

Alleviation \ [I]mpact	Initial	Rewrite
$TR \overset{--}{\longmapsto} [I]$	$E \overset{--}{\longmapsto} N$	$E \overset{\emptyset}{\longmapsto} N$
	$E \overset{-}{\longmapsto} N$	$E \overset{\emptyset}{\longmapsto} N$
$TR \overset{-}{\longmapsto} [I]$	$E \overset{--}{\longmapsto} N$	$E \overset{-}{\longmapsto} N$
	$E \overset{-}{\longmapsto} N$	$E \overset{\emptyset}{\longmapsto} N$

Table 5.2: Rewriting Rules for Alleviation Relation

Notice here impact relations are treated as the target nodes for alleviation relations. This allows us to model a treatment that mitigates the risk by reducing the severity of risk (e.g., $-- \longmapsto -$). The rules for alleviation relations are presented in Table 5.2. For simplicity, we only consider whether the treatment is selected or not without taking into account the treatment evidence. Moreover, alleviation relations reduce only negative impact relations. The sign "$\emptyset$" indicates that there is no further impact between the treated event and the business object.

In Figure 5.1 (i.e., $TR_{01} \overset{-a}{\longmapsto} [E_{01} \overset{--i}{\longmapsto} G_{03}]$), the event sudden increase on air traffic *(E_{01})* impacts severely on the satisfaction of plan incoming air traffic *(G_{03})*. However, analysts believe the application of perform air space delegation *(TR_{01})* will alleviate the severity of E_{01} significantly (i.e., $-- \longmapsto -$) so that E_{01} introduces lesser DEN evidence to G_{03} .

5.1.4 Social Relations - $\mathcal{S}$

Definition 5.11. $\mathcal{S} \subseteq \mathcal{A}_1 \times \mathcal{O} \times \mathcal{A}_2$ where $a_1 \neq a_2$
Social relations are categorised into three types of relation: dependency, trust on execution, and trust on evidence.

Definition 5.12. `Dep`(i.e., $A_1 \overset{D(O)}{\longmapsto} A_2$) captures a strategic dependency of an actor (the depender-A_1) on another actor (the dependee-A_2) about the attainment some business object (O).

Consequently, the evidence of the business object (O) in the depender's perspective (A_1) follows the evidence in the dependee's one (A_2). We here assume that the evidence values in the depender's are equal with the evidence values in the dependee's rationale as specified in Axiom(5.45)-(5.50).[7]

Dependency	Dependency Axioms	
$A_1 \overset{D(O)}{\longmapsto} A_2$:	$FS(a_2, o) \rightarrow FS(a_1, o)$	(5.45)
	$PS(a_2, o) \rightarrow PS(a_1, o)$	(5.46)
	$NS(a_2, o) \rightarrow NS(a_1, o)$	(5.47)
	$FD(a_2, o) \rightarrow FD(a_1, o)$	(5.48)
	$PD(a_2, o) \rightarrow PD(a_1, o)$	(5.49)
	$ND(a_2, o) \rightarrow ND(a_1, o)$	(5.50)

[7]Unlike previous relations, the source node of a dependency relation is the destination of the relation (i.e., the dependum in the delegatee's rationale) because it is where the evidence is originated.

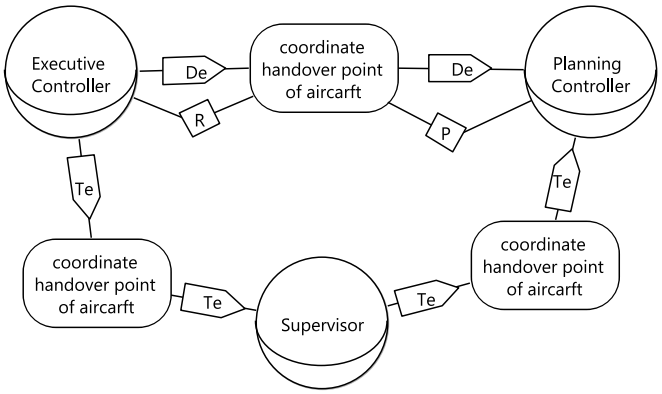


Figure 5.4: Social Relations in ATM scenario

For instance, in Figure 5.4 an executive controller (EC) depends on a planning controller (PC) (denoted by "De") to coordinate handover point of aircraft. Thus, the evidence gathered by EC is equal with the one in PC's perspective. By means of dependency relation, analysts can perform risk assessment or to understand how risks are affecting the overall organisation as we have presented in [15]. We call such risk assessment as actual risk assessment since the mental states of a single actor are not considered.

However, this formalisation is insufficient to understand how much risk perceived by an actor because of a dependency. To enable such analysis, we consider several additional concepts: trust relations and mental states (i.e., utility function, risk tolerance). In [12], we presented how analysts deduce the evidence perceived by the depender by considering the level of trust relation between the depender and the dependee. Back to the example in Figure 5.4, the absence of trust (i.e., neither trust nor distrust) between EC and PC about the goal might erode the evidence values provided by PC. In other words, EC perceives might perceive lesser satisfaction evidence than the one in PC's perspective or exaggerate the denial evidence. However, such perception surely will change where EC trusts or distrusts PC. In fact, in Figure 5.4, EC trusts PC (denoted by "Te") to coordinate handover point of aircraft through the supervisor.

Definition 5.13. Trust relations are distinguished into two classes:

- *Trust on execution* relation ($\texttt{Trust}_e$) represents an agent (the trustor) belief of the capability and dependability of the trustee in fulfilling a goal/business object (the trustum).
- *Trust on evidence* relation ($\texttt{Trust}_{ev}$) represents the trustor's belief of the evidence provided by the trustee about the trustum.

Actually, trust relations are not binary, but rather a range value (i.e.,$[0, 1]$ or fuzzy set). However, for simplification, the framework only distinguishes both trusts into two values: *trust* or *distrust* following [230]. Distrust relation models situations where an actor explicitly distrusts another actor. We here explicitly distinguish between distrust and "no trust" (called lack of trust) where an actor does not have knowledge about the behaviour of the trustee. Thus, in total there are four types of trust relation in GR modelling language: trust on execution, distrust on execution, trust on evidence, and distrust on evidence, which are represented in terms of predicates: $T_e(A_1, A_2, O)$, $S_e(A_1, A_2, O)$, $T_{ev}(A_1, A_2, O)$, $S_e(A_1, A_2, O)$ respectively. The first parameter represents the trustor, the second the trustee, and the last the trustum/the business object intended to be achieved. For example, in Figure 5.5 Robert trusts (denoted as "Te") Paula and Mary on the execution of define partial-delegation schema *(G_{12})*, but

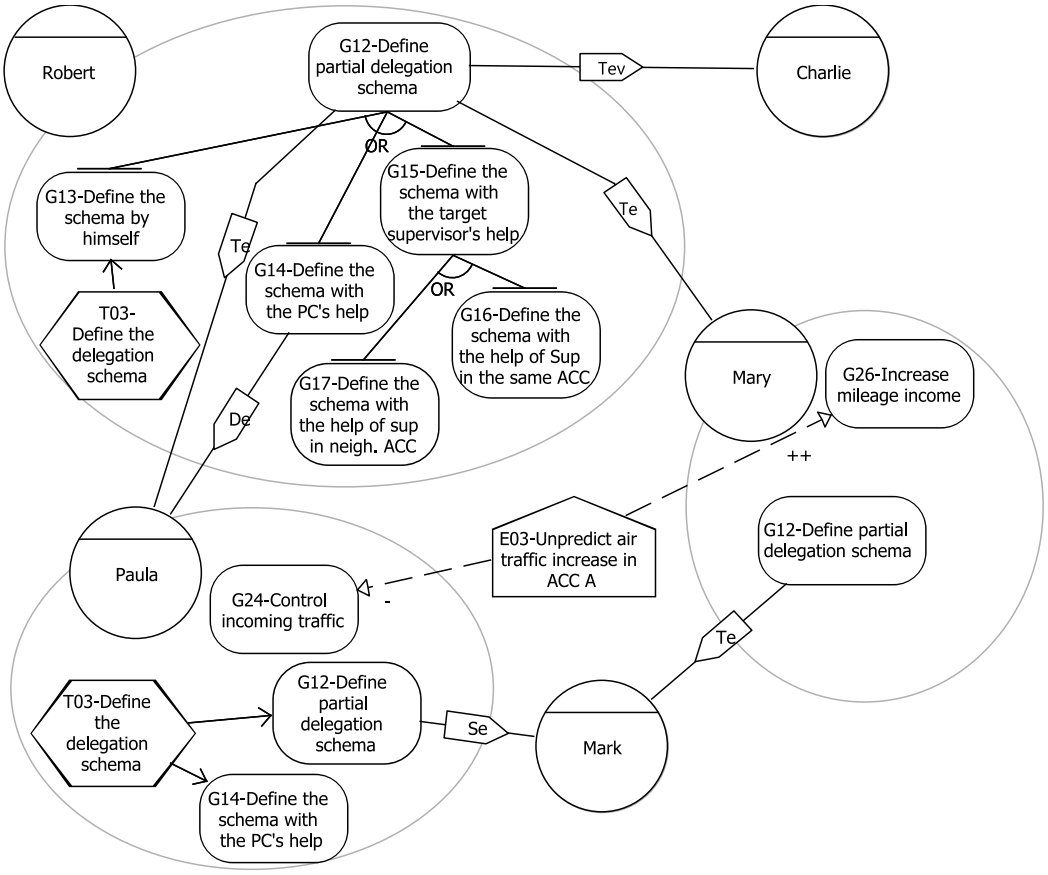


Figure 5.5: Dependency, (Dis)Trust on Execution and Evidence

Paula distrusts (denoted as "Se") Mark on the execution of G_{12} . In addition, Robert trusts (denoted as "Tev") on the evidence provided by Charlie about G_{12} .

(Dis)Trust Relations[8]

$$T_x(A_1, A_2, O) \wedge subnode(O_1, O) \rightarrow \quad T_x(A_1, A_2, O_1) \tag{5.51}$$

$$T_x(A_1, A_2, O) \wedge T_x(A_2, A_3, O) \rightarrow \quad T_x(A_1, A_3, O) \tag{5.52}$$

$$S_x(A_1, A_2, O) \wedge subnode(O_1, O) \rightarrow \quad S_x(A_1, A_2, O_1) \tag{5.53}$$

$$T_x(A_1, A_2, O) \wedge S_x(A_2, A_3, O) \rightarrow \quad S_x(A_1, A_3, O) \tag{5.54}$$

Trusts relations can be established directly or transitively as following Axiom (5.51)-(5.54). For instance, Alice trusts Bob to do X and Bob trusts Charlie to do X, then we can assume Alice also trusts Charlie to do X. Axiom (5.51) & (5.53) specify the transitivity between the upper level node to the subnode. The idea is that if the trustor believes that the trustee actor will (not) achieve a goal, then the trustor also believes that the trustor will (not) fulfil the subgoals. Axiom (5.52) computes the transitive closure of trust relations.[9] It infers indirect relations of trust between two actors. Axiom (5.54) deduces

[8] $x = e, ev$ and $subnode(O_1, O)$ refers to the situation where O_1 is the subnode of O in a AND/OR decomposition.

[9] For the sake of simplicity, we assume that trust is transitive. This choice mainly depends on the qualitative approach adopted in this work. More complex trust metrics can be adopted in a quantitative approach.

the indirect distrust relations between actors. The idea underlying such an axiom is that, if the trustor trusts the trustee, we can reasonable assume that the trustor will distrust the actor that is distrusted by the trustee. As an example in Figure 5.5, Robert trusts Paula on the execution of G_{12} , but Paula distrusts (denoted as "Se") Mark on the execution of G_{12} . Following those axioms, one may assume there is a distrust relation from Robert to Mark (through Paula) about the execution of G_{12} .

Trust Level[10]

$$S_x(A_1, A_2, O) \rightarrow DisTrust_x(A_1, A_2, O) \quad (5.55)$$

$$\neg S_x(A_1, A_2, O) \wedge T_x(A_1, A_2, O) \rightarrow Trust_x(A_1, A_2, O) \quad (5.56)$$

$$\neg S_x(A_1, A_2, O) \wedge \neg T_x(A_1, A_2, O) \rightarrow NTrust_x(A_1, A_2, O) \quad (5.57)$$

By means of trust relations (trust or distrust), analysts may reason about the trust level between actors as specified in Axiom (5.55)-(5.57). In particular, we have considered three trust levels: *Trust*, *Distrust*, and *NTrust* (i.e., neither trust nor distrust) with assuming the following order of precedence: $Distrust > Trust > NTrust$. This choice can be regarded as a particular instantiation of the denial-takes-precedence principle [111]. This corresponds to a conservative approach which discredits all trust relations in the presence of a distrust relation. In other words, the trust level of Robert towards Mark about the fulfilment G_{12} is "distrust". If there is a distrust relation between two actors, then the framework concludes that the trust level between them is $Distrust$; if there are only trust relations, then the trust level is $Trust$. Finally, if neither trust nor distrust relation are identified, the trust level is $NTrust$. In fact, $NTrust$ is necessary since sometime the requirements specification may not define any trust nor distrust relation between two specific actors. Moreover, Axiom (5.55)-(5.57) are also used to define the trust level when there are multi-paths of trust relation between actors. In Figure 5.5, Robert is connected to Mark by a distrust relation on execution for G_{12} through Paula; however, there is also a trust relation between them for the same goal through Mary. In such setting, the framework concludes that the trust-level between Robert and Mark about G_{12} is *Distrust* (based on Axiom (5.55)-(5.57)).

Notice trusting/distrusting another actor will not change the perception of an actor towards risks as far as there is no dependency between the actors. In other words, there is nothing harm in trusting someone. In assessing perceived risk, we need to revisit the dependency Axiom (5.45)-(5.50), because in perceived risk SAT and DEN evidence are propagated along dependency relations with considering the trust level between actors. Axiom (5.58)-(5.60) specify the evidence propagation when the dependency is covered by the trust level of *trust on execution*. In fact, in such settings the evidence gathered by the trustor is independent from the evidence in the trustee's perspective. In Figure 5.5, Robert delegates G_{12} (by delegating G_{14}) to Paula. By following Axiom (5.58), since Robert trusts Paula on fulfilling G_{12} , he might perceive to get *full* evidence of satisfaction on G_{12} from Paula. Conversely, Paula perceives that she will get *full* evidence of denial of G_{12} from Mark (Axiom (5.60)), if she depends on Mart to fulfil G_{12} . It is because Paula distrusts Mark in executing the goal G_{12} .

[10] $x = e, ev$

Dependency $A_1 \overset{D(O)}{\longmapsto} A_2$:	**Dependency-Trust Axiom**	
$Trust_e(A_1, A_2, O)$	$\rightarrow FS(A_1, O)$	(5.58)
$DisTrust_e(A_1, A_2, O)$	$\rightarrow FD(A_1, O)$	(5.59)
$NTrust_e(A_1, A_2, O)$	$\rightarrow NS(A_1, O)$	(5.60)
$Trust_{ev}(A_1, A_2, O) \wedge FS(A_2, O)$	$\rightarrow FS(A_1, O)$	(5.61)
$Trust_{ev}(A_1, A_2, O) \wedge PS(A_2, O)$	$\rightarrow PS(A_1, O)$	(5.62)
$Trust_{ev}(A_1, A_2, O) \wedge NS(A_2, O)$	$\rightarrow NS(A_1, O)$	(5.63)
$Trust_{ev}(A_1, A_2, O) \wedge FD(A_2, O)$	$\rightarrow FD(A_1, O)$	(5.64)
$Trust_{ev}(A_1, A_2, O) \wedge PD(A_2, O)$	$\rightarrow PD(A_1, O)$	(5.65)
$Trust_{ev}(A_1, A_2, O) \wedge ND(A_2, O)$	$\rightarrow ND(A_1, O)$	(5.66)
$DisTrust_{ev}(A_1, A_2, O)$	$\rightarrow NS(A_1, O)$	(5.67)
$NTrust_{ev}(A_1, A_2, O) \wedge FS(A_2, O)$	$\rightarrow PS(A_1, O)$	(5.68)
$NTrust_{ev}(A_1, A_2, O) \wedge PS(A_2, O)$	$\rightarrow NS(A_1, O)$	(5.69)
$NTrust_{ev}(A_1, A_2, O) \wedge NS(A_2, O)$	$\rightarrow NS(A_1, O)$	(5.70)
$NTrust_{ev}(A_1, A_2, O) \wedge FD(A_2, O)$	$\rightarrow FD(A_1, O)$	(5.71)
$NTrust_{ev}(A_1, A_2, O) \wedge PD(A_2, O)$	$\rightarrow FD(A_1, O)$	(5.72)
$NTrust_{ev}(A_1, A_2, O) \wedge ND(A_2, O)$	$\rightarrow PD(A_1, O)$	(5.73)

In the case of trust on evidence, the rule propagations are defined as in Axiom (5.61)-(5.72). We here assume the lack of trust-level will result in worse perception/situation (i.e., reduce the satisfaction evidence or increase the denial evidence). For instance, in Figure 5.5 Robert trusts on the evidence provided by Charlie about define partial delegation schema *(G_{12})* (by means of G_{16}). Suppose Charlie claims there is *full* evidence that he will satisfy G_{16} . By following Axiom (5.61), Robert perceives that the satisfaction evidence of G_{16} is also *full* following Charlie's claim. If the trust-level is *NTrust*, then Robert surely perceives a lesser degree of the satisfaction evidence. However, if the trustor distrusts on evidence given by the trustee, then the trustor assumes *no* evidence regardless the evidence in the trustee's perspective (Axiom (5.67)).

According these axioms, we might reasonably conclude that a dependency in the presence of distrust relation is considered as risky decision for the depender. Ideally, the depender depends to the trusted dependee, though it is hardly possible in every case. These trust-levels, inferred from trust relations, are considered as some factor to calculate the perception of an actor toward risk [20, 73]. Notice the perceived risk is not necessarily real, an actor might over-react or under-react [179].

5.2 GR Model Pre-Processing

A GR model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I}, \mathcal{S} \rangle$, as a mathematical model, has four sets of relations. $\mathcal{I}$ is a set or *impact* relations which act as a relation relating events and business objects and as a target node for the *alleviation* relations. $\mathcal{S}$ captures social relations among actors either about dependencies or trusts. $\mathcal{R}$ captures the remaining relations (e.g., decomposition, means-end, contribution, needed-by). All relations (except trust relations) govern how the evidence from source nodes - $Sat(\langle A_s, G_s \rangle)$, $Den(\langle A_s, G_s \rangle)$ - is propagated to the target node - $Sat(\langle A_t, G_t \rangle)$, $Den(\langle A_t, G_t \rangle)$. In perceive risk assessment, the extent of evidence propagation of dependencies depends also on the trust level between the dependee and the depender.

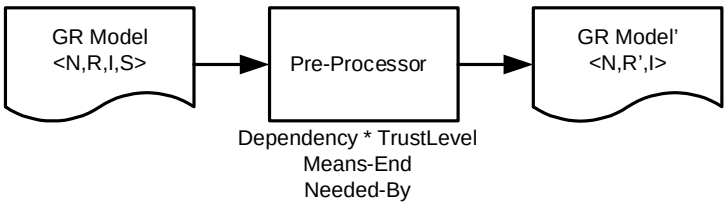


Figure 5.6: Pre-Processing a GR Model

Trust Level	$d \in \mathcal{D}$
$Trust_e(A_1, A_2, O)$	$A_1 \overset{D_{fs}(O)}{\longmapsto} A_2$
$DisTrust_e(A_1, A_2, O)$	$A_1 \overset{D_{fd}(O)}{\longmapsto} A_2$
$NTrust_e(A_1, A_2, O)$	$A_1 \overset{D_{none}(O)}{\longmapsto} A_2$
$Trust_{ev}(A_1, A_2, O)$	$A_1 \overset{D(O)}{\longmapsto} A_2$
$DisTrust_{ev}(A_1, A_2, O)$	$A_1 \overset{D_{none}(O)}{\longmapsto} A_2$
$NTrust_{ev}(A_1, A_2, O)$	$A_1 \overset{D_{\gg}(O)}{\longmapsto} A_2$

Table 5.3: Rewriting Rules for Dependency Relations in Perceive Risk Assessment

Conversely, the assessment of actual risk does not consider the trust level. To make the reasoning is generic, we perform pre-processing to rewrite $\mathcal{R}$ becoming $\mathcal{R}$' as illustrated in Figure 5.6. Essentially, this process rewrites: 1) the product of dependency and trust level relations, 2) means-end relations, and 3) needed-by relations.

The pre-processing works as follows:

- rewrites *means-end* relations. There is a situation where a target node has several means to attain it. It implies there is an option to attain the target node, which is similar with OR decomposition. Hence, these means-end are overwritten into a $\texttt{Dec}_{or}$;

$$(N_1, N_2) \overset{or}{\longmapsto} N \Leftarrow \{N_1 \longmapsto N, N_2 \longmapsto N\}$$

- It rewrites *needed-by* relations. There is a situation where a target node requires several source nodes to achieve the target node. It implies the provision of all source nodes is mandatory for achieving the target node which is similar with AND-decomposition. Hence, these needed-by are overwritten into a $\texttt{Dec}_{and}$.

$$(N_1, N_2) \overset{and}{\longmapsto} N \Leftarrow \{N_1 \overset{n}{\longmapsto} N, N_2 \overset{n}{\longmapsto} N\}$$

- In perceived risk assessment, a dependency relation propagates the evidence depending on the trust level. Hence, we model this relation as a dependency relation with some coefficients indicating the extent of propagation (i.e., called "coef. dependency" $D_x(O)$ where x is the coefficient and O is the dependum). In this framework, we define five types of coef. dependency relations (e.g., $D(O)$, $D_{fs}(O)$, $D_{fd}(O)$, $D_{none}(O)$, and $D_{\gg}x(O)$). The rules of rewriting are defined in Table 5.3.

In fact, dependency relations, in actual risk assessment, are considered as coef. dependency with a basic coefficient (i.e., $D(O)$) where it propagates the same evidence from the dependee to the depender

as defined in Axiom (5.45)-(5.50). D_{fs} captures a situation where the evidence gained by the depender is always *full satisfaction* regardless the evidence of O in the dependee's perspective. Conversely, D_{fd} implies the depender always gains *full denial* regardless the evidence in the dependee's perspective. D_{null} captures a situation where there is no evidence propagated through a dependency relation, while $D_{\gg}$ captures the propagation of evidence when the dependency is lack of trust relation (i.e., neither trust nor distrust). In this work, we assume the depender is a risk averse agent where it tends to downplay the satisfaction evidence coming from the dependee, and to exaggerate the denial evidence.

Coef. Dependency	**Coef. Dependency Axioms**	
$A_1 \overset{D_{fs}(O)}{\longmapsto} A_2:$	$FS(\langle A_1, O\rangle)$	(5.74)
$A_1 \overset{D_{fd}(O)}{\longmapsto} A_2:$	$FD(\langle A_1, O\rangle)$	(5.75)
$A_1 \overset{D_{none}(O)}{\longmapsto} A_2:$	$NS(\langle A_1, O\rangle) \wedge ND(\langle A_1, O\rangle)$	(5.76)
$A_1 \overset{D_{\gg}(O)}{\longmapsto} A_2:$	$FS(\langle A_2, O\rangle) \rightarrow PS(\langle A_1, O\rangle)$	(5.77)
	$PS(\langle A_2, O\rangle) \rightarrow NS(\langle A_1, O\rangle)$	(5.78)
	$PD(\langle A_2, O\rangle) \rightarrow FD(\langle A_1, O\rangle)$	(5.79)
	$ND(\langle A_2, O\rangle) \rightarrow PD(\langle A_1, O\rangle)$	(5.80)

In fact, the rewriting of dependency relations allows us to simplify the Dependency-Trust Axioms - Axiom (5.58)-(5.73) into Axiom (5.74)-(5.80).

5.3 Forward Analysis

Forward analysis aims at assessing the risk level of an organisation. Essentially, it is an adaptation of the one proposed in [80]. It takes the post-processed GR model - $\langle \mathcal{N}, \mathcal{R}, \mathcal{I}\rangle$ and $initial$ containing the evidence value of some input goals (i.e., often they are leaf goals). Notice the distinction of between assessing actual risk and perceived risk is located on the way the $\langle \mathcal{N}, \mathcal{R}, \mathcal{I}, \mathcal{S}\rangle$ are pre-processed into the post-processes one ($\langle \mathcal{N}, \mathcal{R}, \mathcal{I}\rangle$) as described in the previous section. We represent the evidence value as goal labels (SAT and DEN) in the range of $\{F, P, N\}$. All the axioms (i.e., Axiom (5.19)-(5.50), Axiom (5.74)-(5.80)) are encoded in terms propagation rules presented in Table 5.4. For impact relations, the propagation rules are depicted in Table 5.5. Note that the likelihood of event (E) is computed based from the evidence values of E depicted in Table 5.1.

Essentially, $Forward\_Reasoning$ (Algorithm 5.1), the implementation of forward analysis, consists of two main loops. The first loop propagates the input evidence throughout the GR model updating the nodes' labels without considering alleviation relations (line 5-8). Afterwards, given final labels ($current$) it applies alleviation relations mitigating the severity of impact relations. In other words, it rewrites the GR model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I}\rangle$ with less severe $\mathcal{I}^-$. It again reruns in the second loop where the final values of evidence of all nodes are computed. Here, the final evidence values of all nodes in the GR model are computed including their risk level (i.e., DEN).

$Update\_Label$ (Algorithm 5.2) updates SAT and DEN depending on the relations type and the evidence values of the source node. $Apply\_Rule\_Sat(Den)$ (line 6-7) propagates the evidence over decomposition, contribution, means-end, needed-by, or dependency relation relations following the rules defined in Table 5.4. In line 3-4, $Apply\_Imp_Sat(Den)$ computes the evidence propagated by an event over an impact relation following the rules presented in Table 5.5.

[11] $\ll$ and $\gg$ are left and right shift operators, respectively. Assuming $Sat(\langle A, G\rangle)$, $Den(\langle A, G\rangle) \in \{full, partial, none\}$ If $Sat(\langle A, G\rangle) = P$ then $Sat(\langle A, G\rangle) \ll 1 = N$. If $Den(\langle A, G\rangle) = P$ then $Den(\langle A, G\rangle) \gg 1 = F$.

Relation	$Sat(N_1)$	$Den(N_1)$
$(N_2, N_3) \overset{and}{\longmapsto} N_1$	$min\left\{\begin{array}{l} Sat(N_2), \\ Sat(N_3) \end{array}\right\}$	$max\left\{\begin{array}{l} Den(N_2), \\ Den(N_3) \end{array}\right\}$
$(N_2, N_3) \overset{or}{\longmapsto} N_1$	$max\left\{\begin{array}{l} Sat(N_2), \\ Sat(N_3) \end{array}\right\}$	$min\left\{\begin{array}{l} Den(N_2), \\ Den(N_3) \end{array}\right\}$
$N_2 \overset{+_S}{\longmapsto} N_1$	$min\left\{\begin{array}{l} Sat(N_2), \\ P \end{array}\right\}$	N
$N_2 \overset{++_S}{\longmapsto} N_1$	$Sat(N_2)$	N
$N_2 \overset{+_D}{\longmapsto} N_1$	N	$min\left\{\begin{array}{l} Den(N_2), \\ P \end{array}\right\}$
$N_2 \overset{++_D}{\longmapsto} N_1$	N	$Den(N_2)$
$N_2 \overset{-_S}{\longmapsto} N_1$	N	$min\left\{\begin{array}{l} Sat(N_2), \\ P \end{array}\right\}$
$N_2 \overset{--_S}{\longmapsto} N_1$	N	$Sat(N_2)$
$N_2 \overset{-_D}{\longmapsto} N_1$	$min\left\{\begin{array}{l} Den(N_2), \\ P \end{array}\right\}$	N
$N_2 \overset{--_D}{\longmapsto} N_1$	$Den(N_2)$	N
$N_2 \longmapsto N_1$	$Sat(N_2)$	$Den(N_2)$
$N_2 \overset{n}{\longmapsto} N_1$	$Sat(N_2)$	$Den(N_2)$
$A_1 \overset{D(O)}{\longmapsto} A_2$	$Sat((A_2, O))$	$Den((A_2, O))$
$A_1 \overset{D_{fs}(O)}{\longmapsto} A_2$	F	N
$A_1 \overset{D_{fd}(O)}{\longmapsto} A_2$	N	F
$A_1 \overset{D_{none}(O)}{\longmapsto} A_2$	N	N
$A_1 \overset{D_{\gg}(O)}{\longmapsto} A_2$ [11]	$Sat(\langle A_2, O \rangle) \ll 1$	$Den(\langle A_2, O \rangle) \gg 1$

Table 5.4: The Rules of Evidence Propagation for relation $r \in \mathcal{R}'$. Note: $N_1 = \langle A_1, O \rangle$ and $N_2 = \langle A_2, O \rangle$

In this reasoning, since the value of evidence indicates the (at least) number of evidence that N has, we assume the final evidence value of a node is the maximum (i.e., max) of evidence values gathered from all incoming relations (line 8). Let's suppose, we have$(N_1, N_2) \overset{and}{\longmapsto} N$ and $N_3 \overset{++_S}{\longmapsto} N$, and the evidence of each node is $Sat(N_1) = P$, $Sat(N_2) = F$, and $Sat(N_3) = F$. From $\texttt{Dec}_{and}$, it has $Sat(N) = P$, and from $\texttt{Contr}_{++_S}$ $Sat(N) = F$. Since we do not have any information stating how much evidence are overlapped, we then assume the final evidence of N is (at least) the maximum one (i.e., $Sat(N) = F$).

Based on $current$ evidence values coming from the first loop, the new impact relations are defined (i.e., it considers all alleviation relations). Thus, using $Apply\_Alleviation$ (Algorithm 5.3) we rewrite the sign (i.e., severity) of impact relations that are alleviated by treatments by means of $Update\_Sign$ (line 3). In fact, $Apply\_Alleviation$ is an implementation of the rules presented in Table 5.2). Afterwards, we recompute the labels using the rewritten GR model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$ (i.e., with less severe $\mathcal{I}^-$). The evidence values of all nodes in the GR model are final, and consequently the risk levels of top goals are estimated (i.e., DEN of top goals)

Require: gr_model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$, label_array initial
1: label_array current, old
2: current ←initial
3: i ←0
4: **repeat**
5: **while** $old \neq current$ **do**
6: old ←current
7: **for all** $N_j \in \mathcal{N}$ **do**
8: current$_j$ ←$Update\_Label(j, \langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle, old, initial)$
9: **if** i=0 **then**
10: $Apply\_Alleviation(\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle, current)$
11: old ←**nil**
12: $i++$
13: **until** i=2
14: **return** current

Algorithm 5.1: Forward_Reasoning

Require: int i, gr_model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$, label_array old,initial
1: **for all** $R_j \in \mathcal{R} \cup \mathcal{I}$ s.t. $target(R_j) = N_i \wedge \neg is\_Alleviation(Rj)$ **do**
2: **if** $is\_Impact(R_j)$ **then**
3: sat$_{ij}$ = $Apply\_Imp\_Sat(N_i, R_j, old)$
4: den$_{ij}$ = $Apply\_Imp\_Den(N_i, R_j, old)$
5: **else**
6: sat$_{ij}$ = $Apply\_Rules\_Sat(N_i, R_j, old)$
7: den$_{ij}$ = $Apply\_Rules\_Den(N_i, R_j, old)$
8: **return** $\{max(max\_array(sat_{ij}), initial[i].sat)$,
$max(max\_array(den_{ij}), initial[i].den)\}$

Algorithm 5.2: Update_Label

Termination and Complexity

Theorem 5.1. $Forward\_Reasoning(\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle, Initial)$ terminates after at most $2 \times (4|\mathcal{N}| + 1)$ loops.

Proof. Consider line 8 in Algorithm 5.1, we have that, for every node N_j in $\mathcal{N}$.

$$\begin{aligned} Current_j.sat &= max(..., initial[j].sat) \\ Current_j.den &= max(..., initial[j].den) \end{aligned}$$

so that SAT and DEN values are monotonically non-decreasing. In order not to terminate, at least one evidence value of N_j per step should increase. Each node has two variables $Sat(N)$ and $Den(N)$ where in combination they can change their values at most 4 times - from the lowest value (no) of evidence at SAT and DEN to the highest value ($full$) ones (see lattice in Figure 5.7).

Moreover, the algorithm runs twice: 1) computing the evidence values of treatment acting as alleviator, 2) computing the final evidence values after considering alleviation relations.
Thus, $Forward\_Reasoning$ must terminate after at most $2 \times (4|\mathcal{N}| + 1)$ loops. □

Require: gr_model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$, label_array current
1: **for all** $I_k \in \mathcal{I}$ **do**
2: **for all** $R_l \in \mathcal{R}$ s.t. $target(R_l) = I_k \wedge is\_Alleviation(R_l)$ **do**
3: I$_k$ ←$Update\_Sign(R_l, current)$

Algorithm 5.3: Apply_Alleviation

Impact Relation \ $\lambda(E)$	L	O	R	U
	$Sat(N)$			
$E \overset{++}{\longmapsto} N$	F	P	P	N
$E \overset{+}{\longmapsto} N$	P	P	N	N
	$Den(N)$			
$E \overset{--}{\longmapsto} N$	F	P	P	N
$E \overset{-}{\longmapsto} N$	P	P	N	N

Table 5.5: The Rules of Evidence Propagation for Impact Relation

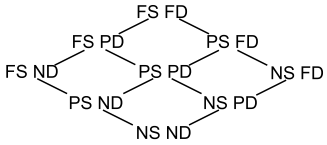


Figure 5.7: Lattice of SAT and DEN values of a node in GR model

Notice that it is the complexity under the worst case scenario, since many values might be updated in parallel at one loop.

Performance Evaluation of Implementation

We have also tested the performance of the implementation of the forward reasoning algorithm with several much bigger test cases that are generated randomly. We conducted the experiment using Python 2.5.1 and a machine with Intel(R) Core(TM)2 Duo CPU 1.6 GHz and 3GB RAM. The chart (in Figure 5.8) shows that processing time grows linearly with the size of the GR model. Moreover, increase of relations adversely affects the performance of the reasoner, more so than an increase in the number of nodes.

5.4 Backward Analysis

In previous analysis, we focus propagating the inputs to assess the risk of important goals (typically top goals) in a given GR model. In this analysis, called backward analysis, we instead aim to seek the possible inputs that can lead us the desired values (i.e., the acceptable level of risk) of important goals considering some constraints (e.g., the goal G can only have at most *partial* evidence of satisfaction). This analysis amounts to the top-down reasoning proposed in [181], where a GR model is encoded into satisfiability formulas and then a SAT formula (e.g., ZChaff[12], MiniSAT[13]) is used to elicit a possible assignment.[14]

[12] `http://www.princeton.edu/~chaff`

[13] `http://minisat.se/`

[14] some SAT solvers can behave as enumerator that enumerate all possible assignments that satisfy the formulas.

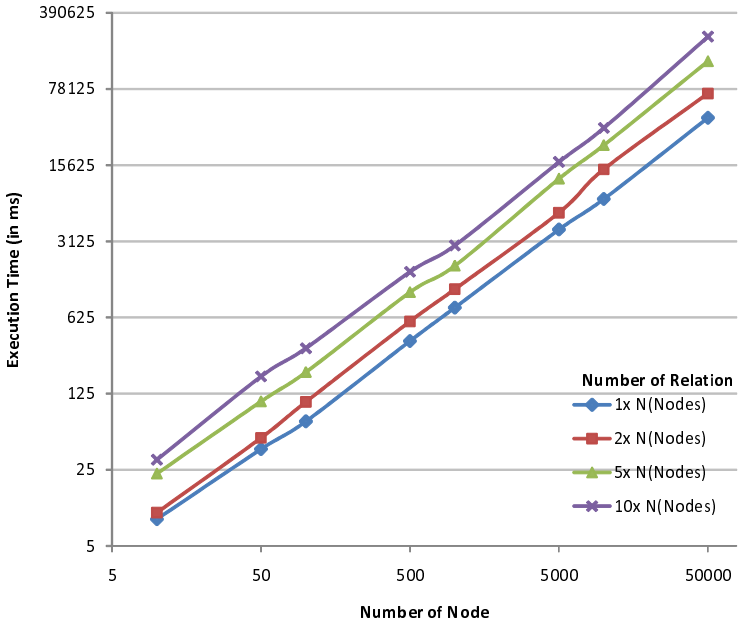


Figure 5.8: Execution Time of Forward Reasoning Algorithm

5.4.1 Encoding to SAT formulas

We need to reduce the problem of backward analysis for seeking the input values into the satisfiability (SAT) of a boolean formula Φ. As proposed in [181], the boolean variables of Φ are all the values $FS(N)$, $PS(N)$, $FD(N)$, $PD(N)$ for every node $N \in \mathcal{N}$, and Φ is written in the form:

$$\Phi := \Phi_{graph} \wedge \Phi_{outval} \wedge \Phi_{backward} \left[\wedge \Phi_{constraints} \wedge \Phi_{conflict} \right] \tag{5.81}$$

where the conjunction of Φ_{graph} is the representation of a GR model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$, Φ_{outval} captures the desired values of the important goals, $\Phi_{backward}$ encodes the backward search axioms which are the inverse of the propagation rules in Table 5.4. Moreover, $\Phi_{constraints}$ specifies a set of constraints or desiderata of node values. $\Phi_{conflict}$ allows the user for looking for possible input values which do not involve conflict of evidence (i.e., $N \in \mathcal{N}$ where N has some values in both polarities of evidence). Though this analysis is based on the top-down reasoning proposed in [181], there are some adaptations required to make the reasoner is best suited for risk analysis.

The main difference is located in the encoding of the backward rules (i.e., $\Phi_{backward}$). First, in the "original" top-down reasoning when the user defines the desired values as $FS(N)$ and $PD(N)$, it is correct if the proposed input values result in $FS(N)$ and $FD(N)$ (i.e., $F > P$). It is because based on Axiom (5.2) $FS(N) \rightarrow PD(N)$, hence the input values are considered to fulfil the desired values. In other words, defining $FS(N)$ and $PD(N)$ as desired values refers to the query to seek any possible input values that can result in (at least) *full* evidence of satisfaction and (at least) *partial* evidence of denial. However, this definition is inappropriate for seeking input values in the context risk analysis. When users define the desired values, they intend that it is the smallest evidence for satisfaction that they expect, and the highest evidence of denial that they can sustain. Therefore, we revise the backward axioms as presented in Axiom (5.89)-(5.91). Essentially, the revision is done by adding "guard" predicates, in terms of negative literal, to prevent some node having evidence which is

higher than expected. Second, we add new axioms because of the introduction of new relations (e.g., dependencies).

To make this chapter self-contained, we present how the encoding is done which is **similar** with the original top-down reasoning[181]:

Encoding the GR model

Φ_{graph} represents the GR model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$ given in the form:

$$\Phi_{graph} := \bigwedge_{N \in \mathcal{N}} Invar\_Ax(N) \wedge \bigwedge_{r \in \mathcal{R}/\texttt{Alleviate}} Rel\_Ax(r), \tag{5.82}$$

$Invar\_Ax(N)$ is the conjunction of the invariant Axiom (5.1) and (5.2) for each node in $\mathcal{N}$ of a GR model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$ and $Rel\_Ax(r)$ is the conjunction of the relation axioms in Axiom (5.19)-(5.50) and Axiom (5.74)-(5.80) for each r. These axioms encode the forward propagation of evidence values through the edge of GR model.

Representing the Desired Values

Φ_{outval} captures the desired values that the users intend to have for the important goals (i.e., typically top goals). It is composed in the form:

$$\Phi_{outval} := \bigwedge_{N \in Important(\mathcal{N})} vs(N) \wedge \bigwedge_{N \in Important(\mathcal{N})} vd(N) \wedge \bigwedge_{N \in Important(\mathcal{N})} \neg nvd(N) \tag{5.83}$$

$Important(\mathcal{N})$ is the set of important goals in $\mathcal{N}$ and $vs(N) \in \{NS(N), PS(N), FS(N)\}$, $vd(N) \in \{ND(N), PD(N), FD(N)\}$ are the evidence values that the users intend to achieve. Notice that the evidence predicate refers to the "at least" value some evidence supporting satisfaction/denial, hence based on Axiom (5.1)-(5.1) we might conclude $FD(N) \rightarrow PD(N)$. To prevent the users getting the denial evidence higher than their prescribed, we need to add the negation of $nvd(N) \in PD(N), FD(N)$ where the value of $nvd(N)$ depends on the value of $vd(N)$. For instance, if the users require to have $PD(N)$, then the $nvd(N)$ is $FD(N)$; in the case of $vd(N) = ND(N)$, then the $nvd(N)$ is $PD(N)$.

Encoding Backward Reasoning Axioms

$\Phi_{backward}$ encodes the axioms of backward search which is written in the form:

$$\Phi_{backward} := \bigwedge_{N \in \mathcal{N}/Input(\mathcal{N})} \Big(\bigwedge_{v(N)} Backward\_Axiom(v(N)) \Big) \tag{5.84}$$

$$Backward\_Axiom(v(N)) := v(N) \rightarrow \bigvee_{r \in Incoming(N)} Prereqs(v(N), r) \tag{5.85}$$

$Input(\mathcal{N})$ is set of input goals in G. $Incoming(N)$ is a set of relation in $\mathcal{R}$ where the target node is N and $v(N) \in \{NS(N), PS(N), FS(N), ND(N), PD(N), FD(N)\}$. $Prereqs(v(N), r)$ is a formula which is true if and only if the prerequisites of $v(N)$ through the relation r hold. The list of possible backward relation axioms $Backward\_Ax(v(G))$ is specified in Axiom (5.86)-(5.91).

Goal **Backward Relation Axioms**

$$N^{15}: \quad FS(N) \rightarrow \begin{cases} \bigwedge_i FS(N_i) & \vee & \text{If } (N_1,\ldots,N_i) \overset{and}{\longmapsto} N \\ \bigvee_i FS(N_i) & \vee & \text{If } (N_1,\ldots,N_i) \overset{or}{\longmapsto} N \\ FS(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{++_S}{\longmapsto} N \\ FD(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{--_D}{\longmapsto} N \\ FS(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D(O)}{\longmapsto} A_2 \\ NS(\langle A_2, O\rangle) \vee ND(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D_{fs}(O)}{\longmapsto} A_2 \\ likely(E) & & \forall I \in \mathcal{I}\text{: } E \overset{++_i}{\longmapsto} N \end{cases} \tag{5.86}$$

$$PS(N) \rightarrow \begin{cases} \bigwedge_i PS(N_i) & \vee & \text{If } (N_1,\ldots,N_i) \overset{and}{\longmapsto} N \\ \bigvee_i PS(N_i) & \vee & \text{If } (N_1,\ldots,N_i) \overset{or}{\longmapsto} N \\ PS(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{++_S}{\longmapsto} N \\ PD(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{--_D}{\longmapsto} N \\ PS(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{+_S}{\longmapsto} N \\ PD(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{-_D}{\longmapsto} N \\ PS(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D(O)}{\longmapsto} A_2 \\ FS(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D_{\gg}(O)}{\longmapsto} A_2 \\ occasionally(E) & \vee & \forall I \in \mathcal{I}\text{: } E \overset{++_i}{\longmapsto} N \\ likely(E) & & \forall I \in \mathcal{I}\text{: } E \overset{+_i}{\longmapsto} N \end{cases} \tag{5.87}$$

$$NS(N) \rightarrow \begin{cases} \bigwedge_i NS(N_i) & \vee & \text{If } (N_1,\ldots,N_i) \overset{and}{\longmapsto} N \\ \bigvee_i NS(N_i) & \vee & \text{If } (N_1,\ldots,N_i) \overset{or}{\longmapsto} N \\ NS(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{++_S}{\longmapsto} N \\ ND(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{--_D}{\longmapsto} N \\ NS(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{+_S}{\longmapsto} N \\ ND(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{-_D}{\longmapsto} N \\ NS(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D(O)}{\longmapsto} A_2 \\ PS(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D_{\gg}(O)}{\longmapsto} A_2 \\ unlikely(E) & \vee & \forall I \in \mathcal{I}\text{:} E \overset{++_i}{\longmapsto} N \\ rarely(E) & & \forall I \in \mathcal{I}\text{: } E \overset{+_i}{\longmapsto} N \end{cases} \tag{5.88}$$

Termination and Performance

The performance of backward analysis is closely dependant on the performance of underlying SAT solver. Note that SAT is an NP-complete problem [54], so that we can assume that there is no polynomial algorithm able to solve it. However, in reality we have never experienced a lack of performance in solving our problem (i.e., an encoded GR model) It is because a SAT solver is mainly developed for hardware verification where the size of the problem is much bigger (i.e., more than 10000 clauses) than our problems. Hence, we should not worry about the scalability of this analysis.

[15] $N = \langle A_1, O\rangle (non\text{-}input)$

Goal **Backward Relation Axioms**

$$N(\textit{non-input}): \quad FD(N) \rightarrow \begin{cases} \bigvee_i FD(N_i) & \vee & \text{If } (N_1,\ldots,N_i) \overset{and}{\longmapsto} N \\ \bigwedge_i FD(N_i) & \vee & \text{If } (N_1,\ldots,N_i) \overset{or}{\longmapsto} N \\ FD(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{++_S}{\longmapsto} N \\ FS(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{--_D}{\longmapsto} N \\ FD(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D(O)}{\longmapsto} A_2 \\ NS(\langle A_2, O\rangle) \vee ND(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D_{fd}(O)}{\longmapsto} A_2 \\ PD(\langle A_2, O\rangle) & & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D_{\gg}(O)}{\longmapsto} A_2 \\ likely(E) & & \forall I \in \mathcal{I}\text{: } E \overset{--_i}{\longmapsto} N \end{cases} \tag{5.89}$$

$$PD(N) \rightarrow \begin{cases} \bigvee_i (PD(N_i) \wedge \neg FD(N_i)) & \vee & \text{If } (N_1,\ldots,N_i) \overset{and}{\longmapsto} N \\ \left(\bigwedge_i PD(N_i)\right) \wedge \neg FD(N) & \vee & \text{If } (N_1,\ldots,N_i) \overset{or}{\longmapsto} N \\ (PD(N_i) \wedge \neg FD(N_i)) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{++_S}{\longmapsto} N \\ (PS(N_i) \wedge \neg FS(N_i)) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{--_D}{\longmapsto} N \\ PD(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{+_S}{\longmapsto} N \\ PS(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{-_D}{\longmapsto} N \\ PD(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D(O)}{\longmapsto} A_2 \\ ND(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D_{\gg}(O)}{\longmapsto} A_2 \\ occasionally(E) & \vee & \forall I \in \mathcal{I}\text{: } E \overset{--_i}{\longmapsto} N \\ likely(E) & & \forall I \in \mathcal{I}\text{:} E \overset{-_i}{\longmapsto} N \end{cases} \tag{5.90}$$

$$ND(N) \rightarrow \begin{cases} \bigvee_i \neg PD(N_i) & \vee & \text{If } (N_1,\ldots,N_i) \overset{and}{\longmapsto} N \\ \neg PD(N) & \vee & \text{If } (N_1,\ldots,N_i) \overset{or}{\longmapsto} N \\ \neg PD(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{++_S}{\longmapsto} N \\ \neg PS(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{--_D}{\longmapsto} N \\ \neg PD(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{+_S}{\longmapsto} N \\ \neg PS(N_i) & \vee & \forall R_i \in \mathcal{R}\text{: } N_i \overset{-_D}{\longmapsto} N \\ ND(\langle A_2, O\rangle) & \vee & \forall R_i \in \mathcal{R}\text{: } A_1 \overset{D(O)}{\longmapsto} A_2 \\ unlikely(E) & \vee & \forall I \in \mathcal{I}\text{: } E \overset{--_i}{\longmapsto} N \\ rarely(E) & & \forall I \in \mathcal{I}\text{: } E \overset{-_i}{\longmapsto} N \end{cases} \tag{5.91}$$

5.5 Summary

We have presented the formal model underlying the Goal-Risk modelling framework. This formalisation is developed adopting the intuition of the Dempster-Shafer theory of evidence [63, 182]. We also present two basic analysis techniques that are useful for analysts in assessing risk and analysing requirements. *Forward analysis* is proposed to assess the risk level in a GR model for a given set of inputs, while *Backward analysis* aims at eliciting possible input assignments that satisfy a particular value (e.g., risk level). In the following chapters, we demonstrate the usages of both analyses in some case studies. Moreover, we also illustrate some extensions of the *forward reasoning* in assisting analysts to perform cost-benefit analysis and trade-off analysis. The implementation of both analyses is available online.[16]. The detail of how forward reasoning is implemented quantitatively is available in [8].

[16] `http://yudis.asnar.net/tools`

Chapter 6

Goal Risk Methodology

6.1 Risk Management

The concept of risk is well known in literature [58, 101, 104, 107, 109]. In this work, we adopt the risk definition proposed by ISO/IEC: "*combination of the probability of an event and its consequence*" [109]. Risk Management is the continuous process of systematically identifying, analysing, treating, and monitoring risk throughout life cycle of a product or a service [104]. Risk Management comprises of a number of coordinated activities to achieve this purpose, namely risk assessment, risk treatment, risk acceptance, risk communication, and risk monitoring (see Figure 6.1).[1]

Risk assessment is comprised of risk analysis (identification and estimation) and risk evaluation. Starting from the identification of the business objectives of an organisation, analysts identify the events that can obstruct such objectives. Analysts also define the description of the events (e.g., the impacts over the objectives or assets), and estimate their likelihood and severity over objectives and assets. Risk evaluation compares the result of risk estimation with defined risk criteria (e.g., costs, benefits, priorities, acceptable loss). In case the risk level is too high (i.e., it is beyond the risk tolerance that an actor can sustain), analysts perform risk treatment for identifying the treatments necessary to mitigate risks by reducing their likelihood or severity. Often the selected treatments are not sufficient to mitigate risks due, for instance, to a limited budget. Therefore, stakeholders may decide to accept the risks. Risk acceptance thus intends to relax the level of acceptable loss by an actor. All these processes are reported for the purpose of communication among actors across the organisation as well as for monitoring and auditing purposes.

In the previous chapter, we have presented a modelling framework for risk identification, description, estimation, and treatment identification, within organisations. Here, we present step-by-step the Tropos Goal-Risk methodology (Figure 6.2) which is composed of risk assessment and treatment. We show how to contextualise such general processes in more rigorous processes using the GR framework (i.e., modelling language and analyses) and the Air Traffic Management (ATM) case study as a running example.

We will perform an operationalisation activity to refine imprecise concepts to be measurable in specific observations. The examples presented in this section are illustrator examples of the different steps of our approach in the context of the organisational domain for ATM. The general scenario can be instantiated tailoring the steps included in the proposed approach for a specific organisation. Acquiring

[1] The figure is built based on ISO Guide-73:2002 [109] as baseline, and several adjustments following ISO 16805:2006 [104], ISO 27005:2008 [107], Risk Management PMBOK [101], and COSO-ERM [58].

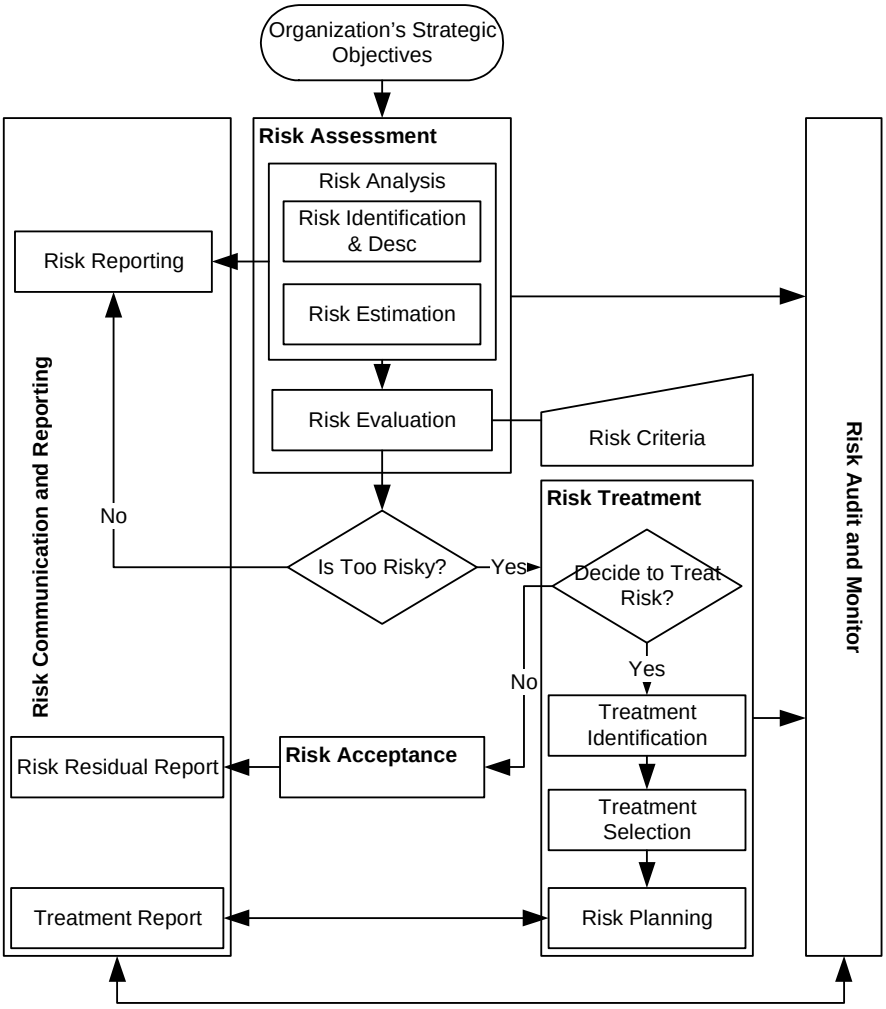


Figure 6.1: Risk Management Process

the organisational knowledge can be done by interviewing the managers of the selected organisation. However, such work is out of the scope of this thesis.

The methodology process is composed of four phases (see Figure 6.2) starting from (1) *value analysis* where value generators of each actor in the organisation are identified and analysed in detail. In GR framework, the value generators are captured in terms of the objectives that an actor intends to achieve including the means to achieve them. Afterwards, (2) *event analysis* analyses the events that affect the identified assets; the likelihood and severity of every events are also assessed in this phase. Once business assets and events are analysed, (3) *risk assessment* computes the risk level. If the risk level is beyond the specified risk tolerance, treatments can be introduced and analysed during (4) *treatments analysis*. Phases (3) and (4) are iterative, and the process stops when the risk level is acceptable by every actor in the organisation.

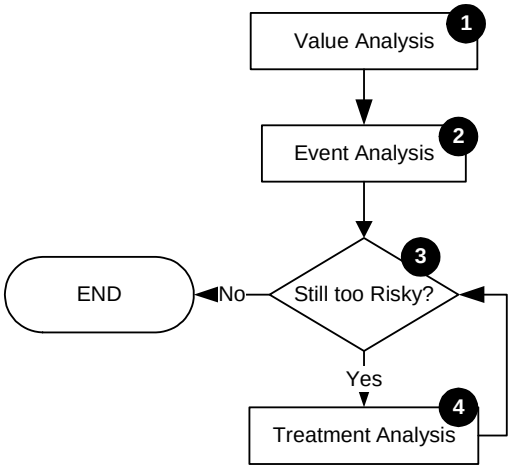
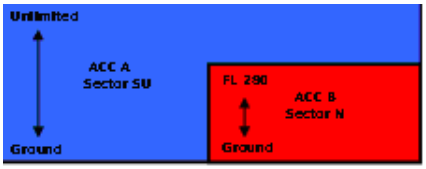
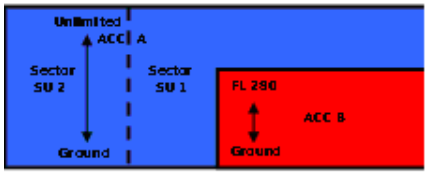


Figure 6.2: Tropos Goal-Risk Methodology



(a) Airspace before re-sectorisation



(b) Airspace after re-sectorisation

(c) EC (Left) and PC (Right) cooperate

Figure 6.3: Managing a Sector with Unexpected Increase in Traffic

6.2 Air Traffic Management Example

This case study has been studied as part of the work in the Industry led project SERENITY [3, 13]. Here we give a brief overview of Air Traffic Management (ATM) to give a feeling of the socio-technical system. In a nutshell, ATM is an aggregation of services provided by ground-based Air Traffic Controllers (ATCOs). ATCO should maintain horizontal and vertical separation among aircraft. They must ensure an orderly and expeditious flow of traffic, providing flight context information to the pilots, such as routes to waypoints, weather conditions and (last but not least) issuing orders and directions. These services are provided by Airport Control Towers for the arrival and departure flight phases and Air Traffic Control Centres (ACCs) for the en-route flight phase.

The airspace managed by each ACC is organised into adjacent volumes, called Sectors with a predefined capacity (i.e. number of flights that can be safely managed). Each sector is operated by a team of two ATCOs, consisting of a Planning Controller (PC) and an Executive Controller (EC), who work together as a team and share the responsibility for the safe operation of the sector.

The EC is in charge of all air-to-ground communication. He monitors the aircraft in the sector and provides pilots with instructions on route, altitude, and speed, and information on weather and air traffic information. When an aircraft approaches the sector boundary, he passes it off to the EC of the adjacent sector. The PC assists the EC, coordinating entry and exit flight level and entry and exit flight point with adjacent sectors in order to ensure a smooth air traffic flow. She monitors the traffic within the sectors

and in most of cases updates the ACC system with the clearances given by the EC.

Groups of neighbouring sectors are coordinated by a supervisor, who should adjust sector's configuration in order to manage the traffic forecast for the next period. The predefined procedures of sector configuration, requires the availability of an ATCOs' team (PC and EC) and artefacts to safely manage the traffic (Figure 6.3(c)). A supervisor should monitor as well as fluidly assist and temporarily take over the roles of controllers if required. The supervisor has also the responsibility to promptly avoid delays in crucial information transmissions due to partial failures of automatic information systems. He accomplishes this task directly communicating with controllers.

The story below is a simple example in which airspace's *re-sectorisation* solves an unexpected increase of air traffic.

1. It's 11 a.m. Paula (PC for Sector South Upper-SU of ACC A, see Figure 6.3(a)) notes in the Planning Traffic display an increase of the number of flights for next hour that is exceeding the capacity of Sector SU.
2. Paula asks Robert (Supervisor of South airspace) to reach her position and to proceed with the horizontal splitting of Sector SU into two Sectors. As shown in Figure6.3(b), Sector SU is divided along the dotted line, so that only Sector SU 1 is the neighbour of sector N from ACC B.
3. At 11.15 a.m. sector re-configuration is carried out and is registered by the System in the daily log.
4. As soon as the re-sectorisation is operational, Paula informs neighbouring Sectors of ACC A, providing them with the radio frequency of Sector SU 2 and instructions for the traffic handover.
5. Luke (EC for Sector SU 1) starts handing over to new Sector SU 2 all traffic flying in and to Sector SU 2. The system shows in the displays of both sectors (SU 1 and SU 2) the traffic that is changing sector.
6. It's 12 a.m. Paula and Luke can manage the expected peak of traffic in a safe way.

6.3 Value Analysis

The methodology starts by analysing how organisations generate values. Generally, the detail process of this phase of methodology is illustrated in Figure 6.4. It starts by identifying relevant actors for the application domain. Actors are not only internal to the organisation, but also include external actors participating to the business activities of the organisation such as pilots, passengers, and regulatory bodies.

Example 6.1. *In this scenario, we have identified 12 main actors which are 7 roles and 5 agents:*

- Executive Controller *(EC), responsible to manage current air traffic;*
- Planning Controller *(PC), which must plan and order incoming traffics;*
- Sector Group*, which is the pair of EC-PC that is responsible to manage a particular airspace or sector;*
- Team Sector SU *and* Team Sector SU2*, which are specific sector teams that are responsible for some sector (i.e., SU and SU2);*
- Supervisor*, which must coordinate and monitor the work of controllers within adjacent sectors;*
- Supervisor of South Airspace*, a supervisor which is responsible to supervise sector teams (e.g., Team Sector SU and Team Sector SU2) of the south airspace;*

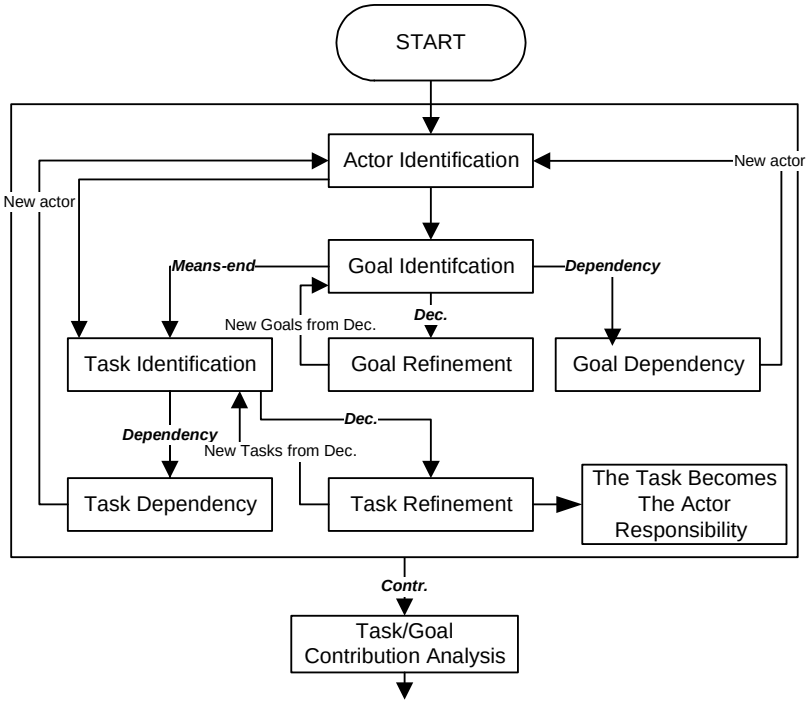


Figure 6.4: Detail Process of Value Analysis

- Luke *and* John *are EC that is part of Team Sector SU and SU2 respectively;*
- Paula *and* Mark *are PC that is part of Team Sector SU and SU2 respectively;*
- Robert *is the supervisor of SU and SU2.*

In Figure 6.5, we depict the actor model of the ATM scenario during all its operations (e.g., normal, re-sectorisation, flow restriction, and partial airspace delegation). In general, Luke and John play a similar role with different airspace authority. It can be seen in Figure 6.5 where both agents play the same role, but they are part different sector groups. The similar intuition holds also for Paula and Mark which are Planner Controllers (PC) in sector SU and SU2, respectively. Each Supervisor supervises several sector teams, where each of them is responsible for managing traffic in their sector. For instance, Robert supervises the south airspace (i.e., SU, SU2). Each team sector is composed of a Planner Controller (PC) and an Executive Controller (EC). EC is responsible to manage traffic in the sector, while PC is responsible to manage incoming traffic to the sector, and both roles work collaboratively. Therefore, a team sector can be seen also as a collaborative role which means that each agent that is part of the role must work collaboratively with its team-mate. The failure of a team member in carrying out its task (i.e., that related with the collaborative role) can be inferred as a failure of all the team members. Collaborative role is heavily used in the ATM case study, because its nature demands the responsibility of all agent members in case something goes wrong. In later example, we only concentrate on a fragment of the scenario (i.e., Collaboration between EC and PC in managing a sector in the scene of re-sectorisation and partial airspace delegation).

However, the value analysis may require the introduction of new actors to whom business objects (i.e., (soft)goals, tasks, or resources) are delegated leading to a new iteration of actor identification. For

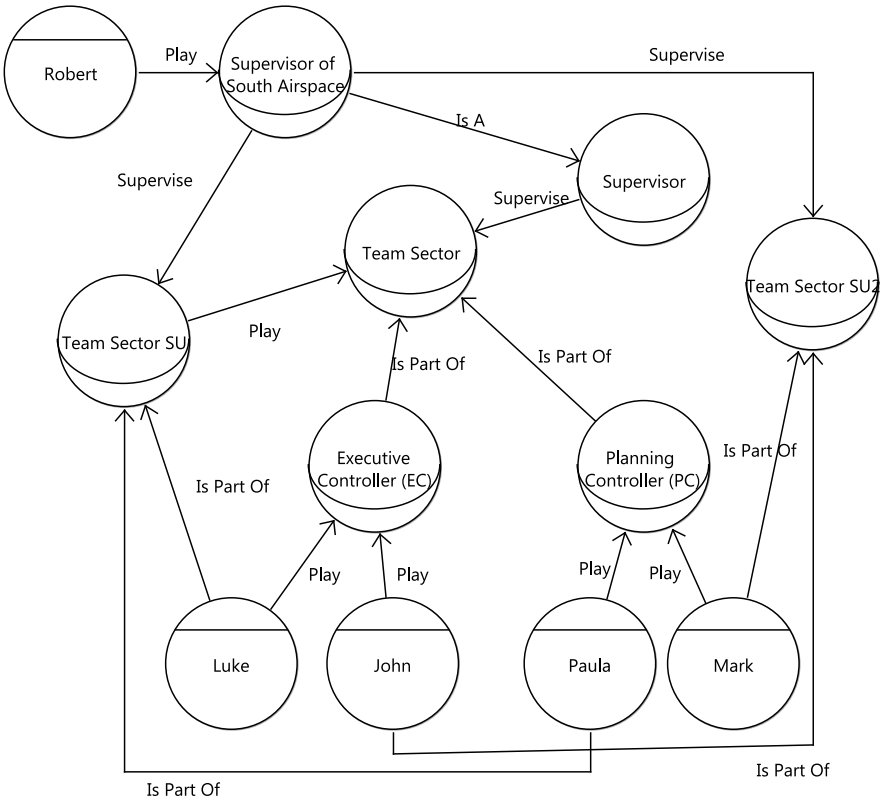


Figure 6.5: An Example of Actor Model at ATM scenario

each identified actor, the analyst identifies its goals. Actors may not be able to fully achieve their goals by themselves. They can either: 1) appoint other actors to fulfil goals entirely, or 2) decompose the goals and assign parts of them to other actors. Identified goals are thus used as input for goal refinement or goal dependency as depicted in Figure 6.4.

Example 6.2. *In Figure 6.6, an EC refines its main objective (depicted as an oval)* manage traffic in the sector *(G_{01}) into* maintain safe separation among traffic *(G_{02}) and* provide information to flights *(G_{03}).*

This *decomposition* continues until each goal is precise enough and all leaf goals are tangible which means they can be achieved by the actor itself or they are delegated to other actors. In other words, all leaf goals must be either assigned by a task to achieve it or passed to other actors.

Notice: In KAOS [210], the authors mentioned four patterns of temporal behaviour of a goal: achieve (or cease) and maintain (or avoid). The former indicates a goal that an actor intends to achieve (to avoid) sometime in the future, and the latter captures a goal that will be maintained to be satisfied (or to be avoided) throughout the operation time. Analysts can indicate all identified goals with these modalities which can be useful in identifying possible risks threatening them and defining the necessary treatments.

The tasks of providing means for achieving goals are also identified in this phase, and include the necessary resources to execute them. They are analysed in a manner similar to that for goals (i.e., decomposition and dependency). *Means-end* relation is used to capture a situation where a task (depicted

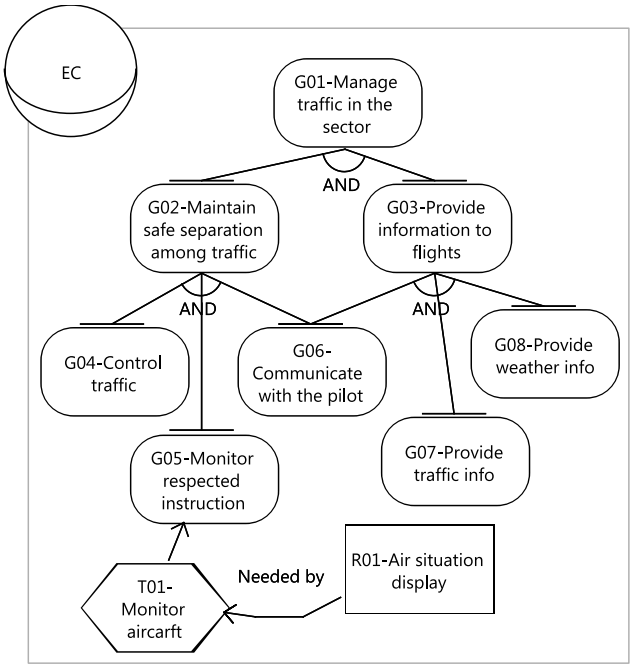


Figure 6.6: Decomposition, Means-End, and Needed-by Examples in ATM Scenario

as hexagon) is a means to achieve a goal, and *needed-by* relation is used to indicate which resources are required.

Example 6.3. *The execution of task monitor aircraft (T_{01}) (depicted as a hexagon)acts as a means to monitor respected information (G_{05}). However, T_{01} requires a resource (depicted as a rectangle) Air Situation Display (R_{01}).*

In this work, we leave it explicitly, if the goal control traffic *(G_{04})* is not decomposed any further, it infers that G_4 is achievable by means of some task.

Example 6.4. *In Figure 6.7, an EC obliges to manage traffic in the sector (G_{01}) and manage inbound traffic (G_{12}). EC should depend on the PC for G_{12} because he is incapable to achieve G_{12} .*

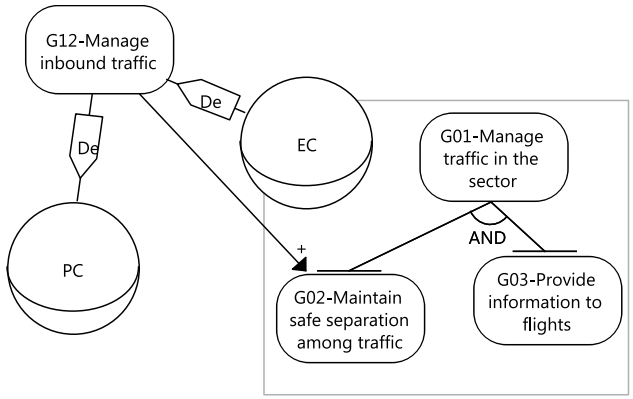


Figure 6.7: Goal Dependency and Contribution Examples in ATM Scenario

Dependency relations result in a new goal/task in the viewpoint of PC(called the dependee) and, consequently, require further analysis (i.e., decomposition, means-end, needed-by) similar with the previous one. However, there is a difference between goal and task dependency. Assuming Luke (called depender) depends on Bob (called dependee) for achieving a goal (called dependum). It implies that the dependee can decide the means (i.e., tasks) to fulfil the delegatum. Conversely, depending on an actor for executing a task means that the depender has determined the exact procedure to be followed by the dependee for executing the task. In other words, the depender specifies "how-to" perform the assigned duty.

Notice: In *i** [226], dependency is categorised into three levels (e.g., open, committed, critical) depending on the criticality of dependum for the dependee. In [73], the authors distinguish delegation (or dependency) on the nature how the dependency is established. The weak dependency does not presuppose any commitment before (e.g., we wait for the bus), and the strong dependency requires some commitment at the beginning. In other words, in a weak dependency the depender believes that the dependee will perform its activity and result in a dependum, while a strong dependency there is an agreement between the dependee to the depender. In this framework, we do not distinguish the types of dependency because it is irrelevant in assessing risk since the dependency is an action (i.e., do or not do) regardless how such dependency is established. Moreover, we argue the notion of dependency-criticality is misplaced because the criticality should lie in business objects (i.e., captured by value or utility function) and not in the dependency. The fact that the satisfaction of the goal manage incoming traffic is critical is independent whether the EC will do it itself or pass it to the PC.

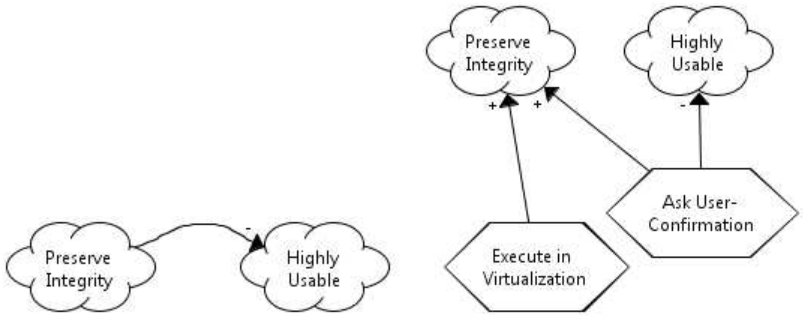
Next step consists of analysing the side-effects of business objects over others using contribution analysis.

Example 6.5. *In Figure 6.7, the achievement of* manage inbound traffic *(G_{12}) will deliver positive contribution for the achievement* maintain safe separation among traffic *(G_{02}).*

One may realise that contribution relations are the most "loose" relations (in terms of the usages) compared to mentioned relations (e.g., decomposition, means-end, needed-by, and dependency) in the GR framework. In nutshell, all relations, except contribution, have nature of *causation* relation while contribution relations are more similar with the notion of *probable causation* [176]. Thus, the meaning of a contribution is the achievement of G_{12} can support (or obstruct) the attainment of G_{02} , and it is **improper** to assume that G_{12} can result in (or inhibit) the achievement of G_{02} .

Notice: Analysts should be aware with the existence of spurious relations. For instance, one may argue preserve system integrity contributes negatively to having usable system because of the systems will nag the users each time there is a new executed process. However, it is just circumstantial because we can have a secure system without having a nagging computer system as illustrated in Figure 6.8. Moreover, (statistical) correlation or conditional probability [171] are other relations that have similar (i.e., not the same) behaviours. For instance, one can see how many ice cream sellers in a beach, then s/he concludes whether there are sharks or not in the sea. In fact, this does not imply that the ice cream sellers are shark-busters, but rather there is a correlation between two events. In other words, correlation is necessary for a contribution (probable causation), but it is not sufficient. The discussion about correlation and causality is deeply studied in [160].

Another issues related to the contribution analysis is how to define the extent of contribution relations. Though qualitatively the GR framework supports only four levels (i.e., $++$,$+$,$--$,$-$) with two modes (i.e., SAT and DEN) for each, we often experience some lacking of:



(a) Spurious Contribution (b) Contribution Relations

Figure 6.8: Spurious Relations in Contribution Analysis

- The "sign" does not indicate *not positive and negative* in common sense, but rather indicates the level and the propagation direction of evidence from the source node to the target node. Essentially, $+_S$/$++_S$and $-_D$/$--_D$ are considered as "supporting" contribution (i.e., positive contribution in commonsense), because they add the value of satisfaction evidence to the target node. Conversely, $+_D$/$++_D$ and $-_S$/$--_S$ introduce some denial evidence to the target node.
- Analysts are too concentrated on the SAT-related contributions (e.g., $+_S$,$-_S$). It is because they are the most commonsensical, and weak on analysing others. $+_S$ and $++_S$ captures the "commonsense" positive contribution, because they indicate if the target node is attained then there is new piece of information considered as SAT for the target node, and $-_S$ and $--_S$ behave on the contrary (i.e., common-sense negative contributions). However, there are some situations where the failure in the target node will bring some side effect to another node in a GR model.

  Example 6.6. *Assuming the unavailability of* air situation display *(R_{01}) adds the* SAT *of the goal of* test ATCO's readiness in emergency situation*.*

  Such situation is captured by "$-_D$" contribution relation which implies the DEN evidence of the resource is propagated to the goal as SAT evidence.
- Strong or weak contribution is the decision that is closely depending on the judgement of the analysts. However, the analysts must realise they often do not have sufficient knowledge to make such judgements. Some decision tools (e.g., Analytical Hierarchical Process (AHP) [174], Interviews) can be useful to define the extent of contribution relations.

In the case of analysing softgoal (e.g., having highly usable application), analysts can perform similar analysis as mentioned above. However, before performing means-end analysis, a softgoal must be operationalised into some precise/hard goal, since softgoals, in the GR framework, are treated as criteria or preferences, and often they are imprecise. For instance, the goal have a highly usable application is "hardened" into able to be studied in 2 weeks, and the last goal is analysed any further as mentioned before. To treat softgoals as purely criteria, analysts do not necessarily identify any means to achieve them. It is enough relating the softgoal with some goals that might affect the softgoals positively/negatively with contribution relations. Analysts then explore all alternatives aiming at maximising the satisfaction evidence of the softgoal, besides satisfying the achievement of top goals. By means of *forward reasoning* analysts can evaluate several alternatives and come up with the best-suited solution. This analysis is similar to AHP [174] or Multi-Criteria Decision Analysis (MCDA) [77].

		Utility	Input-SAT
Executive Controller (EC)			
G01	Manage traffic in the sector		
G02	Maintain safe separation among traffic	98	
G03	Provide information to flights	53	
G04	Control traffic		
G05	Monitor respected instruction		0.6
G06	Communicate with the pilot		0.9
G07	Provide traffic info		0.9
G08	Provide weather info		0.8
G09	Assume the traffic when enters the sector		0.8
G10	Navigate the traffic while in the sector		0.9
G11	Handover to adjacent sectors		0.8
G12	*Manage inbound traffic*	78	
R01	Air situation display		0.75
Planning Controller (PC)			
G12	Manage inbound traffic		
G15	*Maintain traffic acceptability of the sector*	68	0.9
G17	*Ensure traffic is conflict free*	88	
G18	Monitor traffic		0.6
G19	Plan incoming traffic		0.9
G20	Coordinate the change of flight plan with adjecent sectors		0.85
R01	Air situation display		0.75
Supervisor			
G13	*Prevent Delay*	63	0.9
G15	*Maintain traffic acceptability of the sector*	68	
G16	*Manage Sector*	68	
Team Sector			
G01	Manage traffic in the sector		
G12	Manage inbound traffic		
G16	Manage Sector		

Table 6.1: Final Results of the Value Analysis

Value Assessment

Once all the modelling steps have been done, analysts should assess the value of goals. The value also represents the criticality of a goal for the organisation business. It can be denoted in terms of financial unit (e.g., Euro, USD) or just a value in the range $[0, 100]$. Note the value here can be referred to *value-in-exchange* which is defined intrinsically and independent from any actors' perspective in the organisation. Moreover, analysts might defined the value of a goal in terms of *utility function* [23] (i.e., value-in-use) that represents how valuable/importance a goal for an actor in the organisation. The main difference is value (i.e., value-in-exchange) of a business object is independent on the actors' perspective, while the utility (i.e., value-in-use) does. The former is used to assess the actual risk, and the latter is used to assess perceived risk. Since in this running example we consider an organisation as a network of actors, we value business objects as a utility function.

The utility value here is only assigned over top goals and most critical goals, and not the subgoals. This approach is taken because often the achievement of subgoals (in AND-decomposition) does not mean anything if some sibling goals fail and the actor cannot achieve the upper level goal.

Example 6.7. *In Figure 6.6, Luke (an EC) can argue that there is no point having been able to perform* communicate with the pilot *(G_{06}) while he cannot fulfil the sibling goal* monitor respected instruction *(G_{05}) due to some malfunction in the* air situation display *(R_{01}).*

Table 6.1 presents the identified actors together with their goals and the tasks/resources employed to fulfil them. We here obtain the value based on the judgement by experts (i.e., DeepBlue's partners[2]).

Example 6.8. *As in Table 6.1, the goal to* maintain safe separation among traffic *(G_{02}) is the most important in the ATM scenario, and the goal to* provide information to flights *(G_{03}) is the least important one. From the value of utility, one can infer that the G_{02} is almost twice as important as G_{03} . Supervisor assesses that the importance of* maintain traffic acceptability of the sector *(G_{15}) and* manage sector *(G_{16}) has the same importance.*

Notice: To get a representative utility value, analysts can interview actors that are involved in the system and aggregate their opinion by some techniques (e.g., Delphi method [136], Bayesian Belief-Network [76]). As mentioned Surowiecki, in [198], often there is smartness in the crowds that is as good as the expert judgement. In case defining the value of business objects, analysts can quantify it following the priorities indicated by the stakeholders/shareholders. In [175, 201], the authors demonstrate how to prioritise several requirements/objectives based on the users' preferences. We then might assume the higher priority of a requirement is, the more valuable such a requirement is.

In the area of business management, Kaplan and Norton, in [118], proposed four perspectives (financial, internal business, innovation-learning, and customer) to analyse each strategy (i.e., goals in our case). In this way, organisations have broadened views to pursue their business objectives and to maintain their sustainability. Moreover, in [97] the author motivates the necessity for organisations to measure the value of intangible aspects (e.g., information, knowledge) besides the tangible ones (e.g., assets, business income) in analysing businesses. We leave the decision defining the most suitable measurement technique for defining value/utility to analysts. It is because most domain applications (e.g., ATM) have specific (performance) metrics (e.g., number of casualties of each accident) which have been well implemented and based on years of past experience.

Evidence Assessment

Analysts should quantify the evidence values (i.e., SAT and DEN) for each goals, resources, and tasks. The value is in the range $[0, 1]$. SAT (or DEN) represents the value of evidence a goal/task/resource will be attained (or be failed) in the future. In this case, we have specified SAT evidence (column Input-SAT in Table 6.1) based on the DeepBlue's assessment. This evidence is used, together with relations among goals and tasks, and dependency relations, by the risk assessment process (Section 6.5) to assess the final evidence of the business goals and finally to calculate the *loss expectancy*.

Notice: In [112], Jaquith points out the weakness of security risk practices where most the assessment is based on experts' judgement and severely lacks hard evidence. Therefore, to define the number of evidence of navigate the traffic while in the sector *(G_{10})* for an EC, we use the metrics of *workload of an actor*. In the ATM context, the workload of ATCOs is defined in terms of the number of flights that can be safely managed in an hour. In our scene at Roma ACC, an EC will manage 43 flights in one hour (in en-route sector). Given that fact, we can assume the EC will be able to achieve the goal navigate the traffic while in the sector *(G_{10})* relatively easy by having strong evidence of SAT (i.e., $Sat(\langle EC, G_{10}\rangle) = 0.9$). We can use similar metrics, as the measures, to define the number of evidence (i.e., SAT or DEN) [13] or other relevant metrics (e.g., security metrics [112], software metrics [84], economic metrics [168]). Indeed, this approach still suffers from some subjectivity of the experts, when they cast the results of

[2] http://www.dblue.it

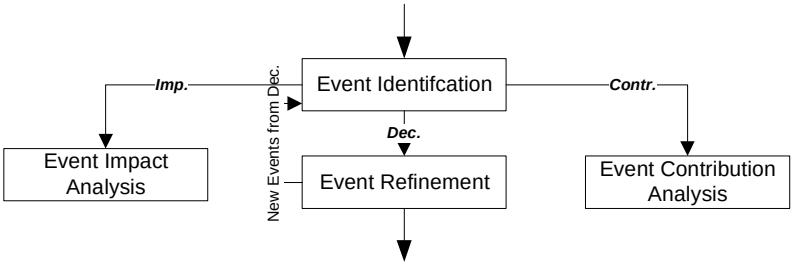


Figure 6.9: Detail Process of Value Analysis

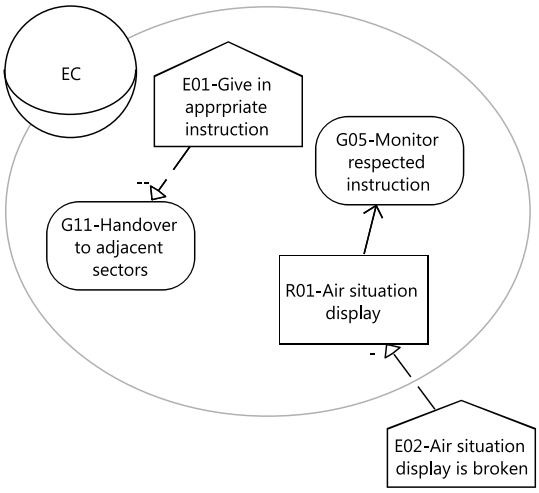


Figure 6.10: An Example of Event Analysis in ATM Scenario

metrics into the value of evidence, but it is much better because the expert judgements are based on existing facts/data (e.g., the actor workload).

6.4 Event Analysis

Event analysis is intended to analyse events and their impact on the value layer which is detailed in Figure 6.9. First, the process identifies the events might affect the attainment of business objects in the value layer. An event is characterised into two properties: likelihood and severity. To identify events, we can follow several guidelines: taxonomy-based [46, 129, 131, 145], failure analysis [17, 196], domain oriented [133, 210]. Moreover, analysts must also exploit the knowledge of experts in the application domain.

We here use the existing knowledge of ATM experts of DeepBlue for identifying the events that can obstruct the achievement of business objectives of the ATM organisation.

Example 6.9. *An EC is threatened by two negative events (called as risks) as indicated in Figure 6.10. First, the event of* give inappropriate instruction *(E_{01}) to the flights in-contact prevents the achievement of* handover to adjacent sectors *(G_{11}), while the second event of* air situation display is broken *(E_{02}) results in the resource* air situation display *(R_{01}) being unavailable and consequently it prevents* monitor respected instruction *(G_{05}) to be fulfilled.*

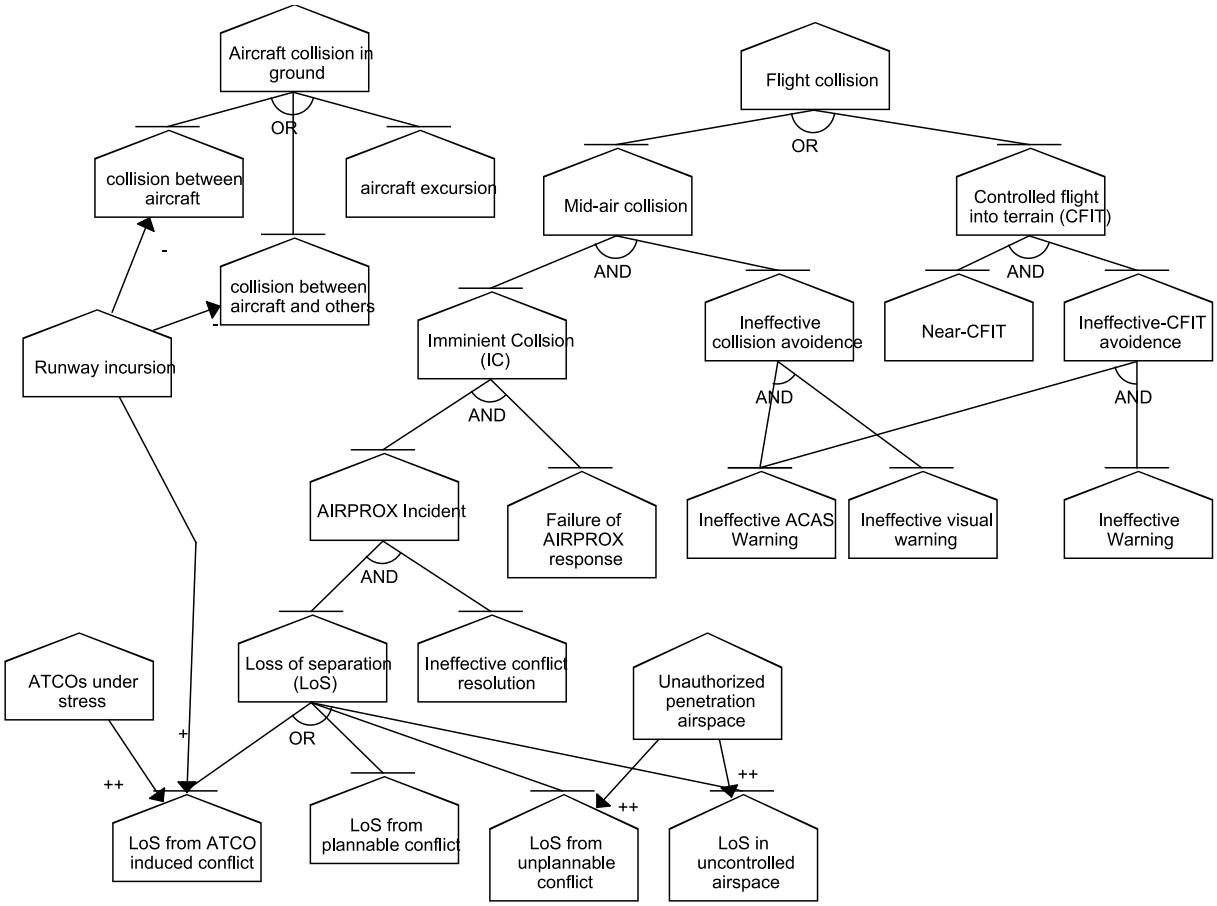


Figure 6.11: Event Refinement in ATM Accident

Each actor has goals to achieve, tasks to execute, and resource to provide. In example 6.9, we start analysing the events that can affect EC. At least there are two events (e.g., E_{01} and E_{02}) that can prevent him to attain his business objects. Notice E_{01} is *local event* and E_{02} is *global event*. In other words, the likelihood of E_{01} might be different for Luke and John though both play the same role. Conversely, the likelihood of E_{02} is independent from actors' perspective.

We then refine events into finer events using decomposition relations (AND/OR) as in Figure 6.11. The refinement process stops when all leaf-events are either easily assessed or their decomposition is negligible (i.e., their likelihood is too remote).

Example 6.10. *In Figure 6.11, we illustrate how to refine two major aircraft accidents:* flight collision *and* aircraft collision in ground*. Both events are refined into finer events.* Flight collision *can occur in two possible setting:* mid-air collision *involving more than one aircraft and* controlled flight into terrain (CFIT) *where resulted from* Near-CFIT *alarms and the pilot fails to avoid the crash due to* ineffective CFIT-avoidance *plan.*

Essentially, this is similar to the proposal by Fault-Tree Analysis (FTA) [196] which refines failure-modes of a technical system. It stops when reaching basic events where the data is available or further refinement does not make meaningful difference. In fact, all leaf nodes in Figure 6.11 are assessable using the data provided by EUROCONTROL's reports [71, 193, 194] and there is no point to refine them

	Event	Likelihood
E02	Air situation display is broken	Unlikely
E03	Failure Comm. Pilot-ACC	Unlikely
E04	Fail to access traffic info	Rarely
E05	Communication Failure between ACCs	Rarely
E06	Miscommunication in coordinating exit point	Rarely
E07	Sudden air traffic change	Occasionally
E08	PC is overloaded	Rarely
Executive Controller (EC)		
E01	Give in apprpriate instruction	Occasionally

Table 6.2: Events Likelihood

any further since EUROCONTROL only records the data in that granularities. To ensure whether the leaf event will not introduce significant uncertainties to the system, analysts can perform sensitivity analysis [69]. It considers a given set of possible values of the leaf nodes, and observes the variability of final likelihood of the top-level events. However, events, in the GR framework, can be quantified using subjective data (e.g., belief-value [115, 182], Bayesian probability [171], expert judgement [53, 155]).

Notice: Analysts must ensure that the event is decomposed into several disjoint subevents (**not** necessary to be independent). It is important to decide whether the assessment is done using qualitative and quantitative data and how they are gathered (i.e., subjective or objective). This decision is critical to decide which mathematical model is the most suitable.

All events then are also analysed through contribution analysis to capture the interdependence among them, and some **notice** of contribution relations mentioned before are also applicable here. Moreover, to identify the extent of contribution one can use also some (statistic) correlation/dependency tests [218] if the analysts have sufficient data/facts. Note analysts must ensure the correlation is based on the (probable) causation and not just circumstantial.

Event Identification

The fulfilment of business objectives might be disrupted by the occurrence of uncertain events, and consequently they prevent the organisation creating/generating values. Hence, in this framework we distinguish risks into three classes following the level in the value layer (i.e., risks at artefact, process, and objective level). Most of the state of practices define risks as the obstruction to the availability of some assets (e.g., servers, production machines). However, the availability of assets (or resources) is not sufficient to guarantee the continuity of business objectives. There could be circumstances where the disruption is introduced from the process or even the objective level.

Example 6.11. *The availability of* air situation display *(R_{01}) does not guarantee that the goal* monitor respected instruction *if the business process (i.e., implicitly captured by a task) is disrupted by some event (e.g., employee strike, EC is on relief-room).*

Moreover, sometime when all artefacts and business processes are functioned correctly an organisation can fail in realising its business objectives due to some circumstance (e.g., new competitors, new regulations, etc.). Therefore, analysts must analyse all possible risks that can obstruct the value generation at three levels.

To identify risks at objective level, analysts can benefit for the works at organisation theory [169], risk at IT governance [58, 154], the works on security requirement analysis [186, 209, 210, 230]. In

process level, there have been some works [26, 113] in identifying risks in business process management, and finally in artefact level, one can use some taxonomy-base approaches (e.g., computer program flaws [129], faults [17]) to identify some potential events, or domain-oriented analysis [86, 133, 224].

We recommend analysts to start the event identification process from the artefact level then move up to the process, and finally the objective level. In this manner, we prevent the spurious identification of an event's impact. If an event disrupts a resource, then certainly it will also produce a disruption effect to tasks that use such a resource, and consequently this will affect goals that the tasks are supposed to satisfy. Conversely, in the case of the risk at the process level (e.g., employee strike), the event cannot be captured as it is the risk for air situation display *(R_{01})*. Identified events then are analysed following similar steps as mentioned before.

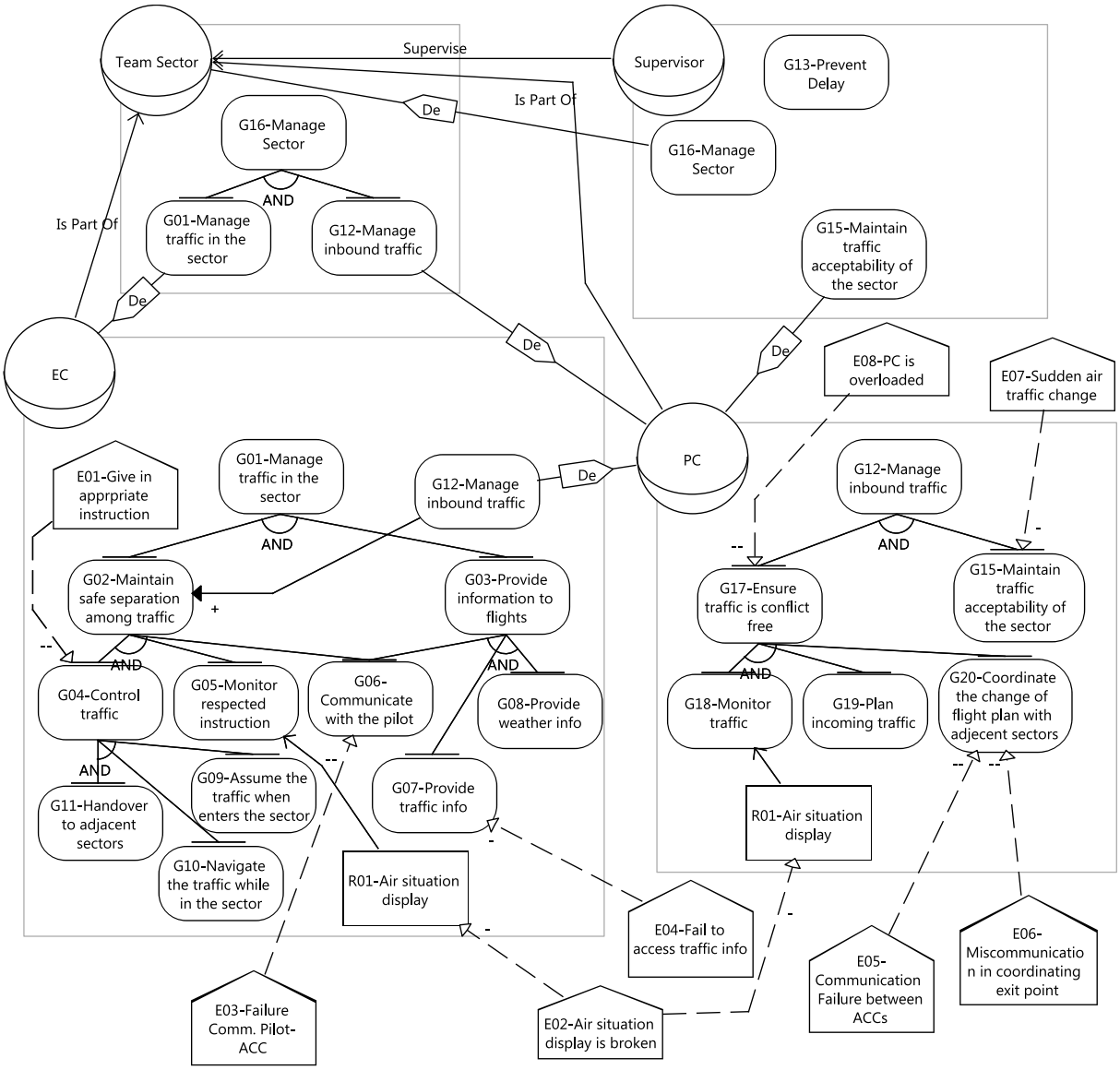


Figure 6.12: Value & Event layer of ATM Scenario

Likelihood and Severity Assessment

Afterwards, the impact of events over the value layer is described and their likelihood estimated. Impact relations ("dash relation" in Figure 6.11 and Figure 6.12) capture the severity of events impacting the value layer. Analysts should be aware with some issues related with this estimation. Some issues/notices mentioned in the contribution relations are applicable here also. Table 6.2 shows the likelihood of events. These values here have been defined according to the experts' knowledge of the application domain. To define the likelihood, the experts measure the frequency of an event in a year time period or use the probability data provided by EUROCONTROL [71, 193, 194] and stratified them into four qualitative values (e.g., likely, occasionally, rarely, and unlikely).

Example 6.12. *Our study, in [13], indicates that the air traffic (in ACC Rome) exceeds the sector capacity only at certain time of the day (i.e., 12.00 - 16.00). Thus, we can assume that controllers are rarely to be overloaded (i.e., E_{08} in Figure 6.12).*

In the case of defining severity, the experts assess the number of loss (i.e., causalities) resulted from the occurrence of an event. To simplify the assessment, we categorise the severity into two classes: minor ("−") or major ("−−").

Example 6.13. *Based on the data at [50], 88% of mid-air collision happens due to the failure of ATC in maintaining separation. In other words, the severity of the event* give inappropriate instruction *(E_{01}) by an EC to the pilot is major for achieving* control traffic *(G_{04}) safely.*

Notice that events can have multi-impacts toward the value layer. This allows analysts to perform trade-off analysis, especially over events that act as a risk (i.e., negative impact) for some goals and, at the same time, as an opportunity (i.e., positive impact) for other goals. However, we here concentrate on analysing events as risks.

Modelling Attacks

In Chapter 4-Figure 4.4, we consider two types of failure: attacks which are originated from malicious intents of a threat agent or accident which may or may not be triggered by an actor. A threat (or attack) must have three aspects: motivation, opportunity, and method [162]. We here capture the motivation using the notion of goal and the opportunity and method aspect of an attack are depicted as task representing the capability of an actor to launch an attack.

Example 6.14. *In Figure 6.13,* flight plan *has several risks that obstruct its provision. In other words, there is (at least) a vulnerability/fault that can be exploited/triggered so it results in a failure either by a circumstance* thunderbolt hit the FDP power grid*, a human error* mis-entry of "null" flight ID*, a random process* random failure of FDP server*, or an attack* injecting "null" flight ID*.*

In nutshell, human error and attack have similar typology (i.e., it is motivated by actors' interests), but the former does not have malicious intents, but the latter has. The main difference is in how we treat such events. A human error can be easily deterred by: 1) asking the actor not to do it again, 2) asking to act more carefully, 3) removing its capability to perform such faulty activities, and 4) employing some control mechanisms. However, it is not the case for an attack because we cannot ask attackers not to launch an attack or remove the capability performing the attack (i.e., task fault injection). Things that can be done as defender, we can employ a countermeasure detect intrusion to FDPS that causes the

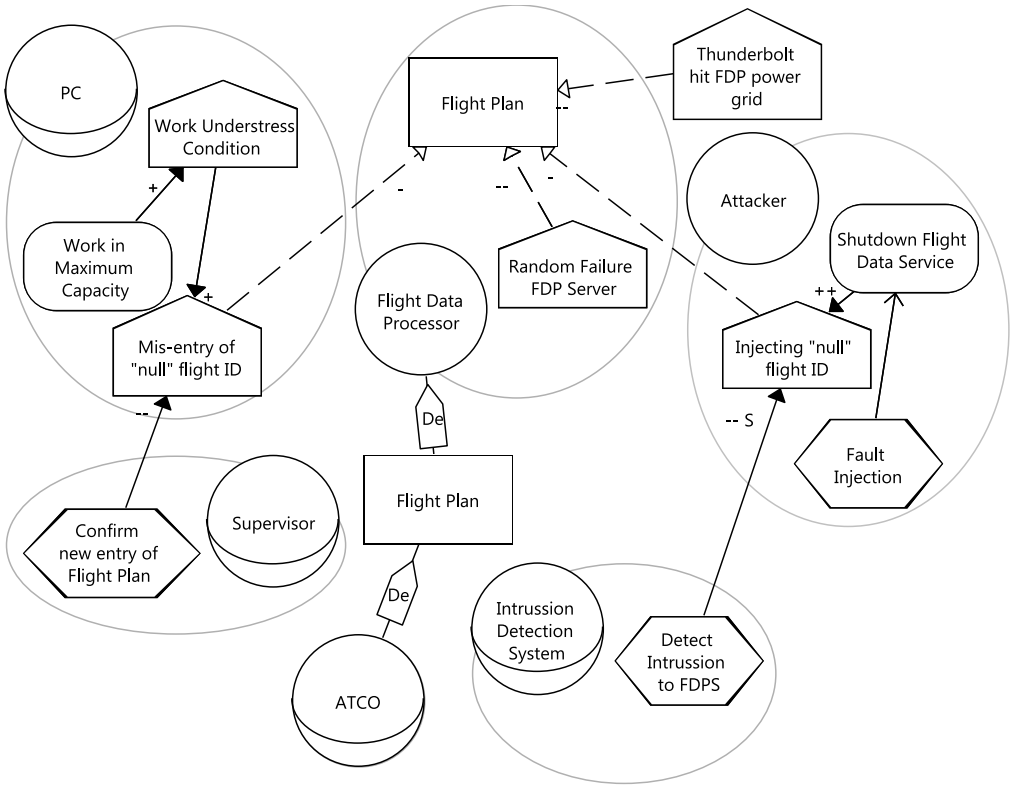


Figure 6.13: Attack and Accident Modelling in ATM Scenario

failure resulted from the attack (i.e., inject “null” flight) is less likely; or we can also employ some safeguards that will alleviate the impact of such attack.

Notice: One may be curious why we do not model a countermeasure as a treatment that contributes negatively to the method (fault injection) or the motivation (shutdown Flight Data Service) instead of inhibiting the failure. All those possibilities are possible though the last one is hardly possible to be realised in reality. Conceptually, negative contributions, as mitigation, can mitigate an attack in three manners:

- Contribution to the method refers to the situation where we can remove the attack capability of the attacker, and it does not refer to making the attack less severe;
- Contribution to the motivation implies there is a measure that discourages/deters/prevents the motivation of attackers. Having “honey pots” as a distraction is one of the example for this class.
- Contribution to the event indicates that the treatment can cause some phenomena (i.e., resulted from an attack) to be less likely, and consequently prevent the failure.

6.5 Risk Assessment

The risk assessment phase is intended to compute the final risk level and to evaluate whether the risk level is acceptable for the organisation (as depicted in Figure 6.13). Since the ATM scenario is a safety critical

system, all risks are considered unacceptable and should be mitigated as much as possible. Therefore, analysts are interested in performing cost-benefit analysis among possible treatments and ideally, they adopt the most beneficial alternative. To assist analysts in performing risk assessment, we have developed a qualitative and a quantitative reasoner of *forward analysis* explained in Chapter 5. It computes the risk level for every actor within an organisation and evaluate whether it is within the specified risk tolerance. In a nutshell, the reasoner computes the final SAT and DEN evidence of business goals for a given set of inputs. In particular, it takes an input of a GR model, the evidence values specified by the analyst (column Input-SAT Table 6.1), and the utility of business goals (columns Utility in Table 6.1). The reasoner also requires the likelihood of events (columns likelihood in Table 6.2). The loss is then computed as the reduction of utility value on the basis of the "negative" evidence (X_{AG}) returned by the reasoner, as follows:

$$Loss(\langle A, G\rangle) = Utility(\langle A, G\rangle) - X_{AG}$$

$$X_{AG} = \begin{cases} \mathsf{DEN}_{AG} + 0.7 \times (1 - \mathsf{SAT}_{AG} - \mathsf{DEN}_{AG}), & \text{if } \mathsf{SAT}_{AG} + \mathsf{DEN}_{AG} \leq 1; \\ \mathsf{DEN}_{AG} + 0.3 \times (1 - \mathsf{SAT}_{AG} - \mathsf{DEN}_{AG}), & \text{if } \mathsf{SAT}_{AG} + \mathsf{DEN}_{AG} > 1. \end{cases} \quad (6.1)$$

where $Utility(\langle A, G\rangle)$ is the value specified by the analyst for business goal G with respect to actor A, and SAT_{AG} and DEN_{AG} are the evidence returned by the *forward reasoner* specified in Algorithm 5.1 Chapter 5. Note that we here use quantitative values instead the qualitative ones. In other words, the function of max or min are applied for the evidence values SAT or DEN (i.e.,$\langle 0 \ldots \rangle$) instead of the qualitative ones (i.e., $full$, $partial$, $none$. Under complete knowledge, the total evidence, which is the summation of SAT_{AG} and DEN_{AG}, should be 1. Unfortunately, the world is imperfect, there is a case where we *lack of knowledge* (i.e., the total evidence < 1) - as in the first part of Axiom (6.1), or there is conflicting information (i.e., evidence > 1) - as the second part. These situations result in uncertain evidence. Since the domain application is safety critical, we have the following assumptions:

- 70% of uncertain evidence is considered as negative evidence. Consequently, we define the equation that the *negative-evidence* (X_{AG}) constitutes as DEN and 70% of uncertain evidence;
- in the presence of conflicting evidence, the negative-evidence is DEN that has been subtracted by 30% of conflicting evidence.

The factors of 30% and 70% are defined by domain experts (i.e., DeepBlue) taking into account that it is a safety critical information system.

In this work, we have analysed the risks affecting the ATM organisation as shown in Table 6.3. From that table, we can see how severe risks affect the ATM organisation. The expected loss is around 33% of the total utility which is considered as catastrophic.

Example 6.15. *In Table 6.3, the goal of* maintain safe separation among traffic *(G_{02}) suffers almost half of its utility because occasionally, an EC* gives inappropriate instruction *(E_{01}) which affects severely to G_{02}. Conversely, supervisor suffers only 4.41 out of 63 over the goal of* prevent delay *(G_{13}) though there is no event that threatens the goal.*

This situation occurs because in Table 6.2 the analysts state that only 0.9 of SAT evidence. In other word, there is 0.1 of lack evidence and 70% of it we consider as the negative evidence.

For each goal that is affected by risks, analysts must mitigate unacceptable risks by introducing treatments. Once treatments are adopted, the risk assessment step is re-executed to verify whether the

Actor	Node		Utility	Expected Loss
EC	G02	Maintain safe separation among traffic	98	41.65
EC	G03	Provide information to flights	53	14.31
EC	G12	Manage inbound traffic	78	30.42
PC	G15	Maintain traffic acceptability of the sector	68	23.46
PC	G17	Ensure traffic is conflict free	88	25.08
Supervisor	G13	Prevent Delay	63	4.41
Supervisor	G15	Maintain traffic acceptability of the sector	68	23.46
Supervisor	G16	Manage Sector	68	28.90

Table 6.3: The Result of Risk Calculation

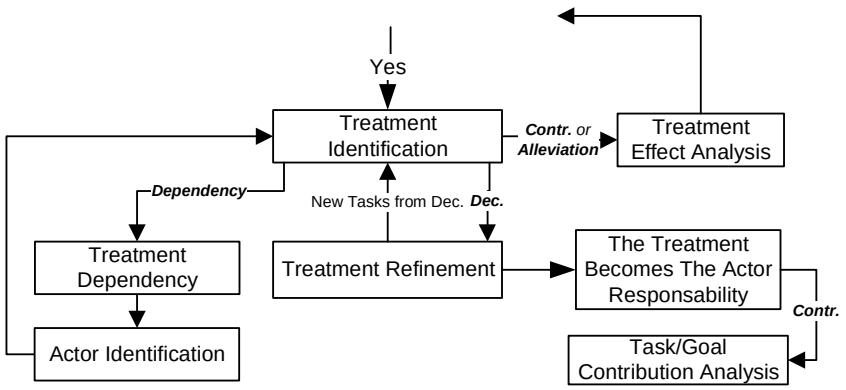


Figure 6.14: Detail Process of Treatment Analysis

residual risk (i.e., the loss after the adoption of treatments) is within the specified risk tolerance; and to evaluate how cost-effective the treatments with respect to the additional cost. To perform cost-benefit analysis, there are some factors to be considered: 1) expected loss (before and after mitigation) and 2) cost of treatments.

As shown in Table 6.3, the loss expectancies of ensure traffic is conflict free *(G_{17})* and maintain traffic acceptability of the sector *(G_{15})* go beyond the thresholds defined in the risk tolerance. Hence, both goals should be protected by some additional treatments.

6.6 Treatment Analysis

This phase aims at analysing treatments required to mitigate risk. The detail steps of this phase are illustrated in Figure 6.14. Actually, this phase shares similar conceptual and methodological approaches used to analyse tasks during the value analysis. First, treatments are identified together with the actors responsible to execute them; then treatments can be either delegated to other actors or refined. During this phase, new actors can be introduced to the system. There are several works detailing how to identify treatments that can mitigate the risks in the organisation, such as best practices (e.g., ISO 27002:2005 [106]), organisational patterns [4], software pattern [90, 180]. Moreover, analysts often still need to exercise their creativity to come up with a new robust solution. Afterwards, analysts assess the effect of treatments mitigating the risks (either reducing the likelihood of events or reducing their severity). The reduction of the severity of events is modelled using *alleviation relations* as shown in the following example:

Example 6.16. *In Figure 6.15, Supervisor can* apply air traffic limitation *(TR_{03}) to alleviate the impact*

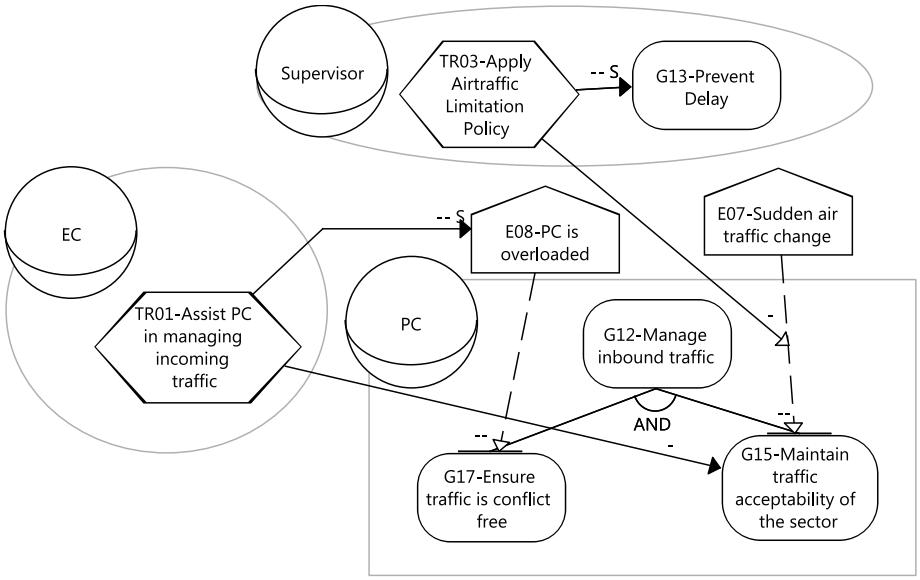


Figure 6.15: Examples of Treatments in ATM Scenario

of sudden air traffic change *(E_{07}).*

To model the reduction of likelihood of risks, one can model a treatment that delivers negative contribution to the risk.

Example 6.17. *EC needs to* assist PC in managing incoming traffic *(TR_{01}) that (probably) results in the event of* PC is overloaded *(E_{08}) is less likely.*

During treatment analysis, the side-effect of treatments over the value layer also needs to be analysed. This step allows analysts to ensure that treatments do not introduce any unacceptable negative side-effects to existing business objects.

Example 6.18. *In Figure 6.15, the introduction of* assist PC in managing incoming traffic *(TR_{01}) delivers negative contribution to the goal* maintain traffic acceptability of the sector *(G_{15}), because EC, now, has a new task and consequently it results in the EC having a smaller capacity to control air traffics than before.*

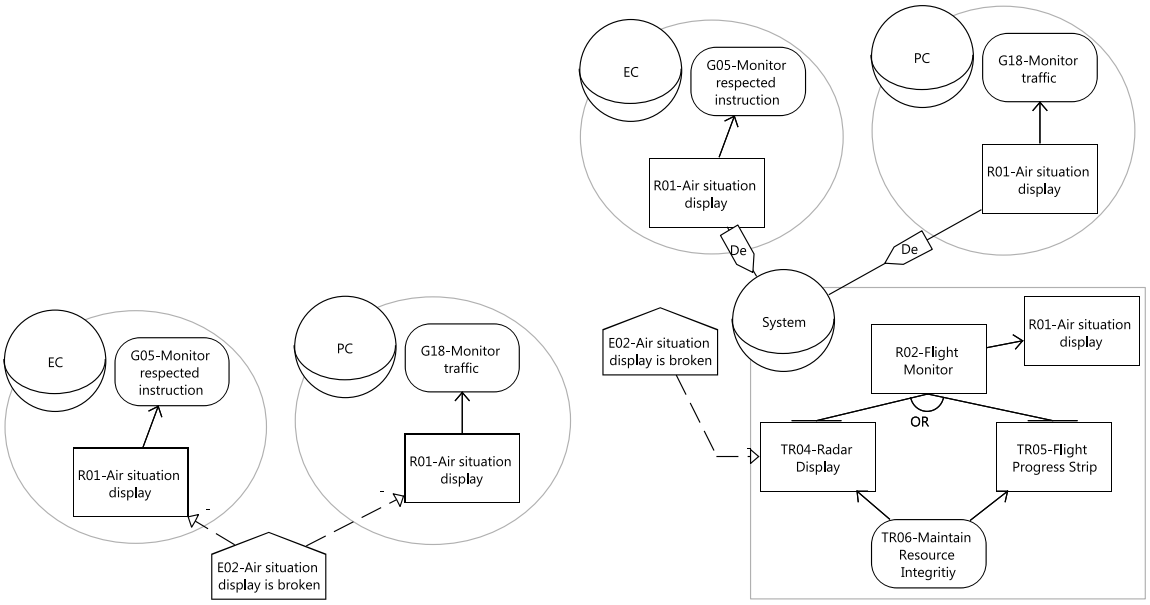
Finally, each treatment is associated with the cost of its adoption. The complete list of possible combination of treatments and their cost is presented in Table 6.4. In the table, we can see there are 7 possible solutions (e.g., S1-S7) that are applicable to mitigate the risks in the organisation.

In this scenario, most of the treatments do not introduce excessive additional cost, because most of them have been in place in the organisation, rather a small one. For instance, an EC has most of the equipments (e.g., workstation, flight progress strips bay, air situation display, incoming flight plans) required to assist PC in managing incoming traffic *(TR_{01})*; therefore, to enable the equipments to mitigate the risk of PC is overloaded *(E_{08})*, we assume that TR_{01} costs 10 points. However, to enable other treatments (e.g., apply air traffic limitation policy *(TR_{03})*), it requires a significant amount of efforts/costs.

Performing monitor respected instruction *(G_{05})* and monitor traffic *(G_{18})*, EC and PC, respectively, require the availability of air situation display *(R_{R01})* to know the situation of air traffic. To avoid

Actor	Node		Cost	S1	S2	S3	S4	S5	S6	S7
EC	TR01	Assist PC in managing incoming traffic	10	X						
Supervisor	TR02	Monitor issued instructions	10		X		X	X		X
Supervisor	TR03	Apply Airtraffic Limitation Policy	20			X	X	X		X
System	TR04	Radar Display	1						X	X
System	TR05	Flight Progress Strip	1						X	X
System	TR06	Maintain Resource Integritiy	20						X	X
		Reduction			5.81	34.88	58.12	7.47	7.47	65.59
		Cost		10.00	10.00	20.00	30.00	30.00	22.00	52.00
		Ratio			0.58	1.74	1.94	0.25	0.34	1.26

Table 6.4: List of Available Treatment



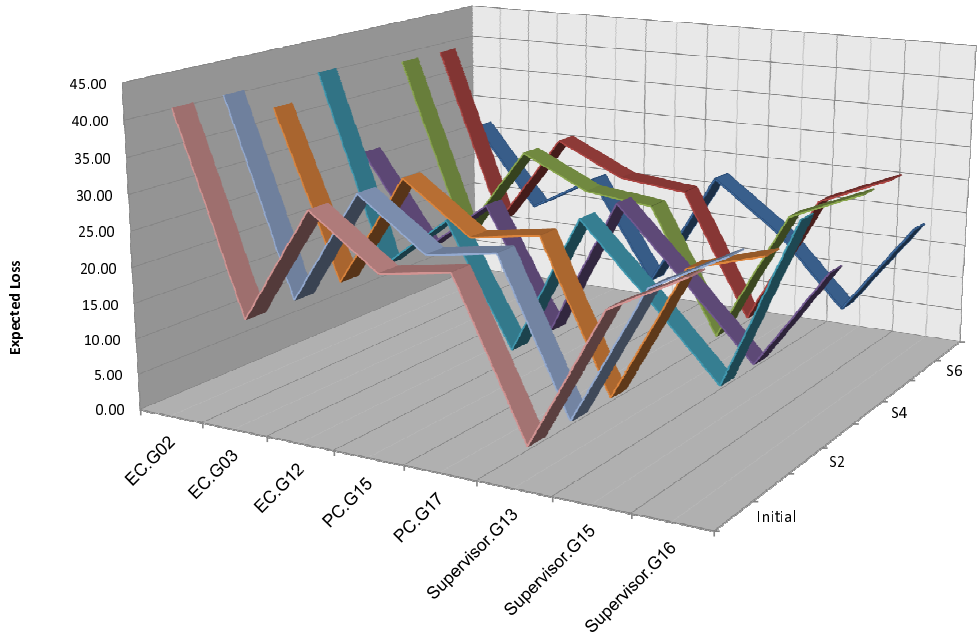
(a) Initial Organisation-setting

(b) After the application of pattern

Figure 6.16: Application of S&D Pattern in ATM Scenario

unavailability of such resource, we can employ some Security & Dependability patterns proposed in [4], such as the *multi representation of information* pattern. Essentially, the pattern aims at representing the information about the air traffic in several modalities (e.g., radar display and flight progress strip) that has existed in the system (i.e., therefore in Table 6.4 their cost is negligible). However, to implement this pattern, analysts should modify some organisation-settings as indicated in Figure 6.16(a)) into the new setting in Figure 6.16(b). The pattern employs a new mechanism to maintain resource integrity *(TR_{06})* between radar display *(TR_{04})* (i.e., it is the same as R_{01} and flight progress strip *(TR_{05})*).

In Table 6.4, we can see that the combination of treatments S4 and S7 mitigate the risk significantly (i.e., 58.12 and 65.59 respectively) while S1 does not deliver risk reduction. By seeing the "ratio" fields, analysts can decide how effective the treatments in comparison with the additional costs. Analysts can perform cost-benefit analysis (detailed in Chapter 7) to see the proportion between the cost and the benefit, in terms of reduction risk, for each alternative (as depicted in Figure 6.18(a)). However, analysts should not base their decision only with the value of "cost-benefit" (i.e., ratio), because the most benefi-



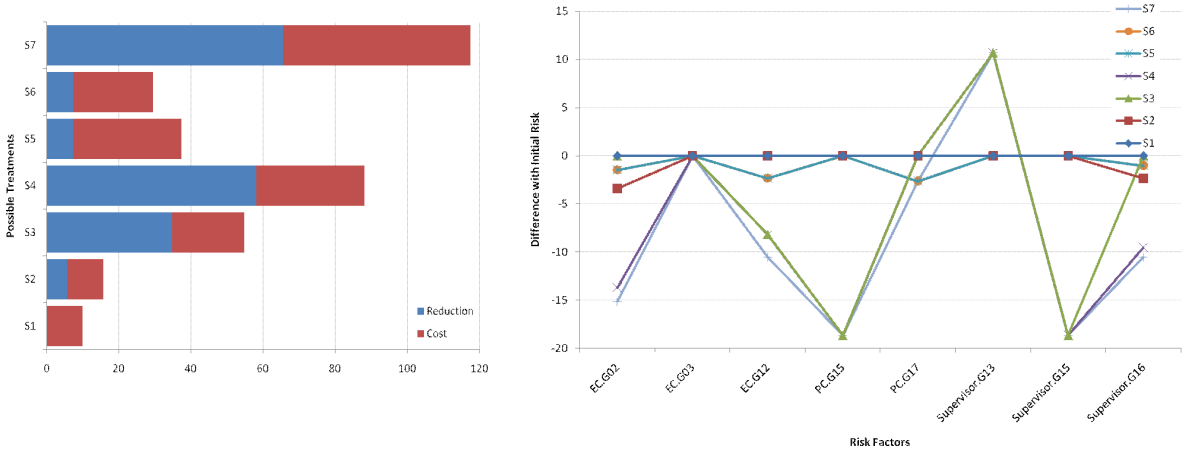
	EC.G02	EC.G03	EC.G12	PC.G15	PC.G17	Supervisor.G13	Supervisor.G15	Supervisor.G16
Initial	41.65	14.31	30.42	23.46	25.08	4.41	23.46	28.90
S1	41.65	14.31	30.42	23.46	25.08	4.41	23.46	28.90
S2	38.22	14.31	30.42	23.46	25.08	4.41	23.46	26.52
S3	41.65	14.31	22.23	4.76	25.08	15.12	4.76	28.90
S4	27.93	14.31	22.23	4.76	25.08	15.12	4.76	19.38
S5	40.18	14.31	28.08	23.46	22.44	4.41	23.46	27.88
S6	40.18	14.31	28.08	23.46	22.44	4.41	23.46	27.88
S7	26.46	14.31	19.89	4.76	22.44	15.12	4.76	18.36

Figure 6.17: Risk Level for Each Possible Treatments

cial alternative might introduce some side-effects that are unacceptable. As complementary, we propose trade-off analysis the effects of each alternative to each risk factor (as depicted in Figure 6.18(b)). For instance, in Figure 6.18(b) we can see that S3, S4, and S7 result in worse situations for the achievement of G_{13} by supervisor where the treatments make the risks even worse. Notice, there is no solution/possible treatments that mitigate the risks of G_{03} of an EC. Essentially, trade-off analysis uses *forward reasoning* as underlying reasoner to compute all possible trade-off for each possible alternative.

The framework allows analysts to perform cost-benefit analysis over alternative strategies by considering the cost of every strategy besides risks, and adopt the most suitable solution. Analysts thus can negotiate a solution with the actors even if residual risk is higher than risk tolerance due to the cost of treatments. When actors cannot accept the residual risk, analysts need to identify additional treatments until the residual risk is acceptable. In this exercise, the domain experts decide to adopt S7. Though it is less cost-effective, it results in the "safest" setting among all possible solutions, and their cost is acceptable for the organisation. The final GR model can be seen in Figure 6.19.

However, analysts can analyse risks of these selected treatments (i.e., 2nd order risks). In some fields (e.g., finance and safety-critical), regulations (e.g., SoX [177] and Basel II [19]) requires the organisation to not just mitigate/control all critical risks, but also to give a high assurance that those treatments/controls



(a) Cost-Benefit Analysis (b) Trade-Off Analysis

Figure 6.18: Cost-Benefit and Trade-Off Analysis for Each Possible Treatments

are reliable.

Treatment Identification

Identifying treatments are one of the most crucial tasks that analysts must perform to protect the system. Actually, there have been several technologies or techniques are proposed in the literature or the practices to control risks. Indeed, new protection techniques are kept being developed as an answer for some incidents/breaches, but often analysts are unaware about new treatments to employ, what are their effects and consequences which are the main issues in treatment identification. In fact, ISO 27002:2005 [106] (i.e., improvement of ISO 17799) has structured best-practices in security controls into a taxonomy. This taxonomy is useful to guide analysts in identifying the best-suited treatments for mitigating risk based on the general idea of security controls, before looking for the enabling technology for implementing the control. Unfortunately, the taxonomy does not indicate the behaviours of those controls in terms of how they affect the risks (i.e., preventing, detecting, etc.)

We here intend to elaborate some guidelines in identifying the most appropriated class of treatment. In [5], we have presented classes of treatments depending on their behaviour. We here refine that proposal into the following categories: *avoidance*, *prevention*, *detection & attenuation*, *retention*. Essentially, risk avoidance aims at avoiding the existence of risk sources and it is done during the system development, while the rests aim at controlling risk during the operation of the system. One can imagine diversity of risk controls as a bull's eye (depicted in Figure 6.20)[45] where the centre is the system, which we intend to protect. A prevention protects the system from the occurrence of negative events (e.g., give authorisation to trusted actors), and detection mechanism detects some events that cannot be prevented and tries to attenuate the impact (e.g., alarm system+response). Finally, a retention works once the risk has succeeded to disrupt the organisation, but it tries to make the overall loss is less catastrophic (e.g., insurance, recovery).

In the following, we detail these classes and pointing out the characteristics (i.e., depicted in a GR model) that must be considered before analysts choosing the most appropriate class of treatments.

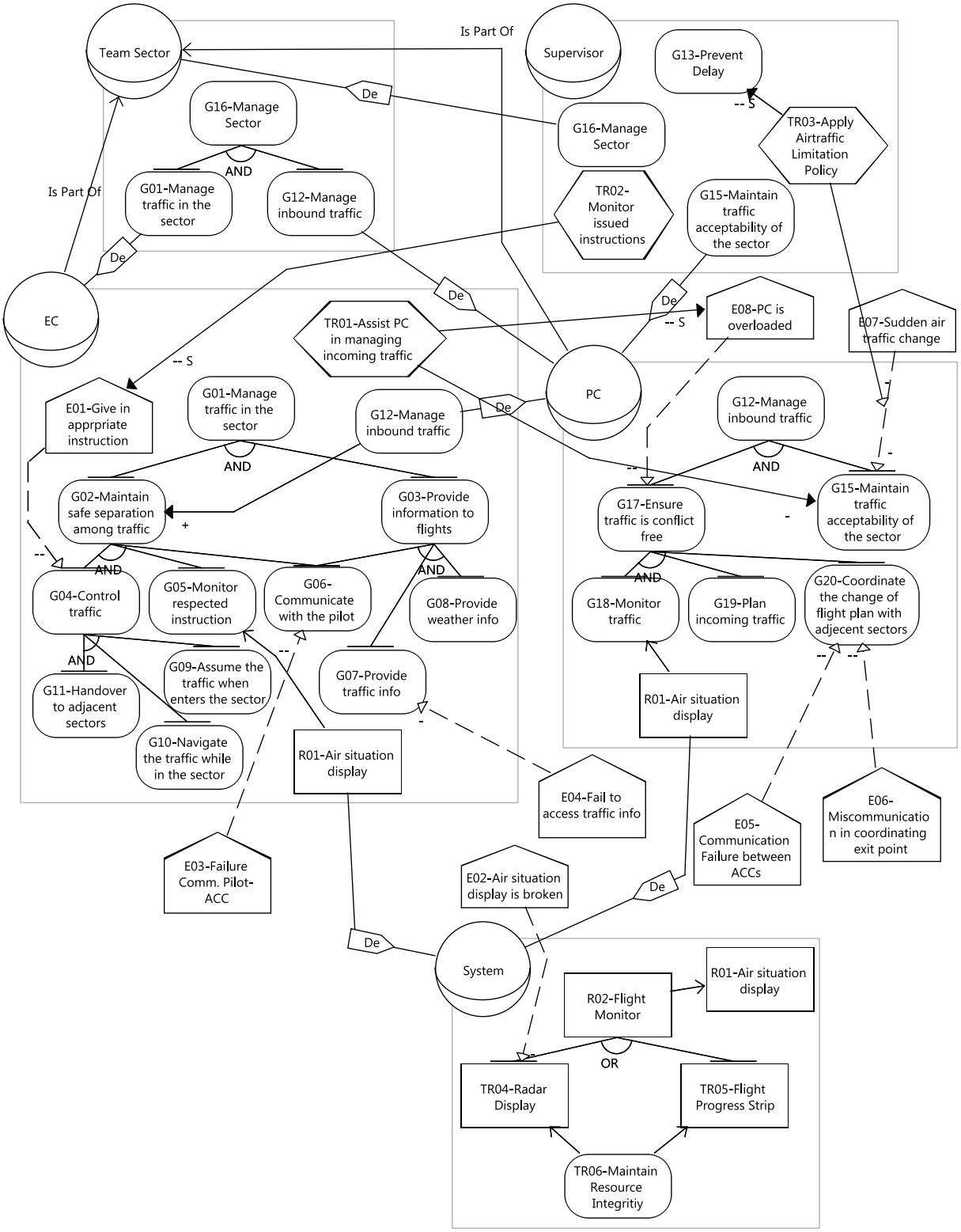


Figure 6.19: The final GR model in ATM scenario

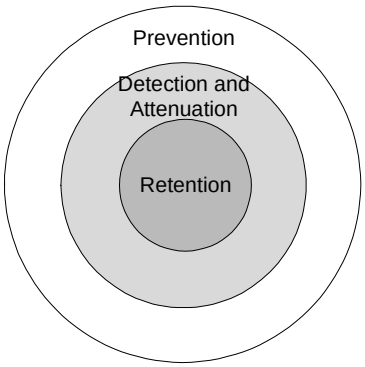


Figure 6.20: Treatments Bull's-eye

Avoidance

We here do not consider avoidance as treatments since it will not manifest as the capability of the system. Avoidance is more an activity that tries to ensure the attainment of stakeholders' strategic objectives by removing all risk sources (e.g., faults, vulnerabilities, flaws) or selecting no-risk alternatives. In other words, this class is considered more as the principles that one must keep in-mind during system development.

Characteristic Analysts need to identify some alternatives that are risk-free or an alternative in which they know how to remove (not mitigate) the source of risks (e.g., vulnerability/bug removal by testing or static analysis [51] or removing organisational design flaws [140]). The idea of goal substitution or wakening [210] is considered as a way to avoid risks. However, this class is not always possible because often risks are introduced because of some business decision, and removing them is considered as ignoring business opportunities. Moreover, often it is impossible to identify all possible vulnerabilities or flaws in the system.

Example 6.19. *In smart-card technology, attackers can extract secret keys and consequently compromise the integrity of smart-card by analysing the power consumption of a smart-card [123].*

This attack is never anticipated before because the designers never think such attack is possible. Avoidance does not introduce any new additional treatment and/or additional cost to employ-operate treatment. Sometime analysts do not need to assess the likelihood (precisely) of risk, but it is enough by deciding whether the cost to avoid is acceptable for the organisation. Notice there could be a setting where avoiding risks is much more costly, than just controlling the risks.

Prevention

This class aims at preventing some negative/bad events to occur in the system. Analysts must define precisely the *unit of mitigation* [90] from the system that need to be protected. It is hardly possible to protect a whole system, hence analysts must define a unit of mitigation which is smaller than the whole system. For instance, in ATM to ensure the reliability operation of controllers, analysts can decide which units of mitigation (e.g., *sector team* or *controller*) are worth to protect. This decision consequently drives which treatments are best-suited to prevent the risk of the unit of mitigation being unreliable.

Example 6.20. *If analysts decide that* sector team *needs to be always functioned, then some measures are allocated to support EC and PC in performing their duties and their interactions (e.g., each of*

controllers support/monitor its team mate, employ redundancy for critical artefacts for their tasks). In the case of controller *as the unit of mitigation, analysts assume that the collaboration in a sector team is not critical. Consequently, it is not necessary to enforce EC and PC to collaborate tightly.*

Characteristic It operates by mitigating leaf-events in the event tree (i.e., negative-contributions to events) so that the likelihood is acceptable.

Example 6.21. *To prevent leaking of sensitive information, a secure system employs an access control [158] to limit/control users' access to the sensitive data. Note that the breach of access control does not always infer to the breach of confidential data.*

Prevention requires analysts to identify the entry-point of attacks or the critical state that can lead to failures. Thus, to protect the system from an attack (e.g., DoS attack) it requires several treatments because it can be originated from several entry points (e.g., outside, trusted host inside the organisation). Moreover, vulnerabilities keep being discovered; hence analysts need to keep employing additional treatments to patch the discovered vulnerabilities. However, the idea of choking-point (e.g., turnstile in the museum, gateway in LAN) is one way to pool the entry point of attacks, and consequently analysts can concentrate all treatments in that point [179]. Moreover, prevention measures keep operated with/without any presence of attacks or errors, thus they are less cost-efficient to address some rare risks. Finally, prevention is good, but detection is necessary because it will protect the system when the prevention fails to prevent an attack.

Detection and Attenuation

This measure intends to detect the occurrence of risks (e.g., attacks or errors) and mitigate them before they compromise the system. The basic consideration in choosing this class is: *detectability* of the attack/error and available *time* to attenuate before it compromises the system. In other words, it is not suitable for addressing attacks/hardly that are hardly detectable (e.g., man-in-the middle attack) or has small window time before it manifests as a system failure.
Characteristic Detection operates by suppressing intermediate events (i.e., negative contribution to intermediate or top-level event) or reducing the impact of an event (i.e., alleviation to the impact relation). This class is highly recommended to be taken as the complementary measures with the prevention. Moreover, detection is best-suited where there are many entry points to launch an attack, and it only operates at the presence of an attack/error.

Example 6.22. *To mitigate the risk of fraud in credit card, banks prefer to detect bootleg credit cards and revoke them if it is necessary. This countermeasure is far less costly compared with protecting all credit cards from being stolen or copied.*

Retention

It is the last alternative to deal with risks. Unlike prevention and detection, analysts do not need to know entry-points/vulnerabilities of the system or possible attacks to retain the system from attacks or failures. Analysts just need to know the existence of deviation for the normal operation in the system, and elicit the measures to recover (or to retain) the system to (up to) the normal operation.

Example 6.23. *Audit of financial transaction is not aiming at detecting and preventing any frauds while they are occurring, but it intends to catch the fraudsters.*

In the example, auditors do not need to know a priori how an attack will be carried on, but they detect the deviation in the system state and investigate backwards how such state occurs.
Characteristics It operates by supporting business objects to be more robust against risks (i.e., positive contribution to the value layer). In the present situation, GR framework is lacking in representing this class of treatments. Notice insurance, recovery, and digital forensic are considered as retention measures because they do not act against the risks, but more to make the system more retentive in the presence of failure.

Notice: Though analysts have employed prevention and detection, sometimes it is necessary to adopt some recovery or retention measures because new vulnerabilities and attacks are being discovered or invented. Moreover, analysts ideally employ several controls from difference classless for an critical business object (i.e., security in-depth principle).

Example 6.24. *To treat credit card frauds, banks employ prevention measures (e.g., provide physical cards, CCV code, signature) that limit the possibility of credit card abuse by fraudsters. However, the banks still operate a fraud detection system that analyse and detect the anomalies in credit card transactions. It will alert customers if there are suspected frauds and responds accordingly. At last, banks still allows customers to repudiate some purchases that have been done without their consent. The last measure is considered as retention measures that protect the interests of customers. Note that this class of treatments will cause a certain period of time where the goal is unsatisfied before it is restored, which in some domain (e.g., safety critical) it is unacceptable.*

Notice the system is a network of dependency among actors, hence the system is as strong as the weakest link. Therefore, we need to compartmentalise a system into several subsystems to limit the propagation of a failure to the entire system.

6.7 Summary and Remark

We have presented the Tropos Goal-Risk methodology developed based on best-practices in risk management (e.g., ISO 27005, COSO) and our experiences in developing several case studies within SERENITY and TOCAI project. It guides analysts to identify the value generators in the organisation and analyse them in detail (value layer). Analysts then is guided to identify and to analyse relevant events that can obstruct the value generation. In general, we can distinguish events into two classes: attacks where there is malicious intents behind them and accident that can be originated from random errors or normal operations of actors. Finally, analysts need to assess the risk level and decide, which treatments must be adopted.

The methodology does not address when and how to perform trust analysis. It is because trust analysis is done independently from all phases of the methodology. Ideally, each dependency of an actor is done for all actors that are trusted. However, it is not always the case in the real world, indeed there might be the case where there are no trusted actors that are capable or some regulations prescribed to do so (e.g., credit rating is defined by credit bureau). We recommend analysts to identify trusts in the organisation independently from all these steps, and then verify whether all dependencies are covered by some trust relations or not. If there is untrusted dependency, then analysts can introduce some additional measures to ensure the reliability of the dependencies (e.g., employ monitoring mechanisms, introduce trusted third-party, insure the dependum).

The methodology is open to any language. For instance, analysts can use FTA or the Attack Graphs or the Bayes Belief-Networks instead of using the event modelling illustrated in Chapter 4. Moreover,

one can also use BPMN to model the process level in the value layer, instead of using the notion of tasks. We adopted this approach to ease analysts in incorporating their knowledge and to reduce their learning effort.

One may argue that this methodology is "very subjective" because it is based the assessment on some subjective data sources (e.g., expert judgements). To minimise the subjectivity of assessment inputs, analysts can obtain the judgements from various experts and aggregate them following some techniques [53, 136, 155, 201]. Moreover, one must realise building a model is also a subjective activity, therefore different analysts may model the same world with different models. Therefore, it is essential for analysts to understand the nature subjectivity in the model, and consequently its impacts. Several analysis techniques (e.g., sensitivity analysis [69], what-if analysis [93], regression test [98]) are useful for such proposes. Moreover, in IT security often analysis must be done under the situation where lacks of hard evidence, and security attacks and vulnerabilities keep being discovered.

The methodology does not have high precision given the nature of requirements, which are less precise than specifications or programming codes. Moreover, building a precise model is costly and takes time. In requirement engineering, we do not always need a precise risk assessment. It is because the usage of risk level to guide in alternative evaluation and to decide whether some additional countermeasures are required. Expressiveness is considered as the most critical characteristics of a risk framework at requirements level. It must be able to capture concepts and relationships relevant for identifying and evaluating risks and requirements. Moreover, it should also be easy to perform repetitively since requirements often suffer some changes during the system development.

To make this methodology more usable, we have built a CASE tool, namely SI*-tool[3], to draw a GR model and analyse it. Moreover, we have defined a requirement template (in a .doc document) that is proved to be useful as a starting point to gather users' requirements. This template has been used in a number of industrial case studies within several research projects (e.g., EU-SERENITY, EU-MASTER, TOCAI). This methodology here certainly does not make risk management becoming an exact science, rather than an art activity. Hopefully, this methodology provides a set of structured processes that guide analysts in eliciting requirements together with managing risks.

[3] http://sesa.dit.unitn.it/sistar_tool

Chapter 7

Risk Management in Organisations

In this chapter, we illustrate the application of the GR Framework in assessing risk in an organisation. The framework is organised in a three-layers model founded on the concepts of *values*, *events*, and *treatments*. Values, modelled in terms business objects (i.e., goals, tasks, resources), are analysed and related to external events that can influence negatively (i.e., risk) their satisfaction. Treatments are then introduced to mitigate the effects of such events. To demonstrate the application of the GR framework in organisations, we here use two case studies originated from industrial partners within our research project: the EU-SERENITY project[1] with the case study of *Loan Origination Process* (LOP) of a bank, the TOCAI.IT project[2] with the case study of *Business Solution Provision* for manufacturing Small-Medium Enterprises (SMEs).

In Section 7.1, we illustrate how risks are assessed and treated to manage the risk level of the business process and the bank in general is below the acceptable risk. Moreover, the GR framework is useful to evaluate alternative business solutions in a multi-actors setting using the second case study in Section 7.2. In this exercise, we also experience that several organisations (i.e., Manufacturing SMEs) may have similar organisation-settings though they have different business models (i.e., product-oriented or order-oriented). The main difference between those types of SMEs is located in the risk model in which they are exposed to. For instance, the risk of *reducing time-to-market* is considered as critical when the business model of the organisation is a produce-based, while it is not the case if the business model is the order-oriented one. Hence, having a "generic" organisational model allows our industrial partners (i.e., Think3 `http://www.think3.com/`) to reuse their knowledge in evaluating alternative business solutions for their clients, and consequently reduces the time and cost of evaluation.

We then demonstrates the extension of the GR framework to assess the business continuity plan of the bank in protecting the LOP in Section 7.3. In the continuity analysis, some of business objects (i.e., supporting infrastructures, business tasks) are allowed to be disrupted as far as they are recovered before the disruption causes the business being discontinued and resulted in catastrophic losses. Finally, we draw some conclusions in Section 7.4

7.1 Risk Assessment and Treatment an Organisation

Every resource and activity in an organisation is directed to realise the business objectives of the organisation. However, there are number of risks that might prevent the organisation in achieving its objectives.

[1] `http://www.serenity-project.org/`

[2] `http://www.dis.uniroma1.it/~tocai/`

Therefore, analysts must assess possible risks and treat them if they are unacceptable.

In this section, we focus on demonstrating the application of the GR framework in assessing risks in an organisation; and then finding and evaluating all possible alternatives (called strategies) for satisfying the organisation's objectives (i.e., top goals) with the acceptable level of risk. In other words, given a GR model each OR-decomposition introduces alternative modalities for the top goals' satisfaction, represented in terms of different sets of leaf goals that can satisfy the top goals. Each of these alternative solutions may have a different cost and expose to different level of risk. Risk can be mitigated with appropriate countermeasures, which however introduce additional costs and further complications.

7.1.1 Risks in Loan Origination Process

A *Loan Origination Process* (LOP) in a bank starts when a loan application is received, and ends with a decision about the application. Clearly, the bank aims to *earn income*, *accept loan applications*, *handle the applications*, and *ensure loan repayment*. When the bank receives the application, it starts the process of verifying data and calculating the credit rating of the applicant. The rating may be computed internally (in-house assessment) or externally by a Credit Bureau. Next, the bank decides on a loan scheme consisting of the loan cap and the interest to be paid. We assume that a loan scheme is initially proposed by the customer, and then the bank makes the final decision. The bank is, of course, also interested in ensuring the repayment of the loan.

Moreover, several events (i.e., threats and/or unintended circumstances) may endanger the success of business objectives. For instance, *forgery of the loan application*, *fake identity document*, *inaccurate credit rating* are potential dangers for the LOP. Accordingly, analysts need to assess their potential impact. If this is deemed unacceptable, then one needs to introduce measures to reduce the likelihood, or to mitigate the impacts of these events. Of course, these additions have to be analysed carefully before their adoption because they introduce additional cost and could delay in the origination process, or even can introduce new risks.

7.1.2 Assessment Process

In Figure 7.1, we depict a GR model resulted from the methodological processes presented in Chapter 6. In this scenario, we assume a bank as one big actor rather than as a multi-actors setting. Given this model, we evaluate possible strategies to realise the stakeholders' objectives (i.e., top goals) with an acceptable level of risk.

Essentially, the assessment process, described in the Algorithm 7.1, consists of three basic steps:

1. find alternative solutions (line 2-3),
2. evaluate each alternative against relevant risks (line 6-7)
3. assess the countermeasures to mitigate risks (line 9-14).

The process requires several inputs: a GR model as input, the desired satisfaction level of each important goal/top goal ($desired\_labels$), the acceptable risk values (acc_risks), the evidence values of goals that are possible candidates for the final solution ($input\_goals$), and finally the evidence values of events ($event\_labels$). This analysis is assumed the following inputs:

- a GR model - Figure 7.1;

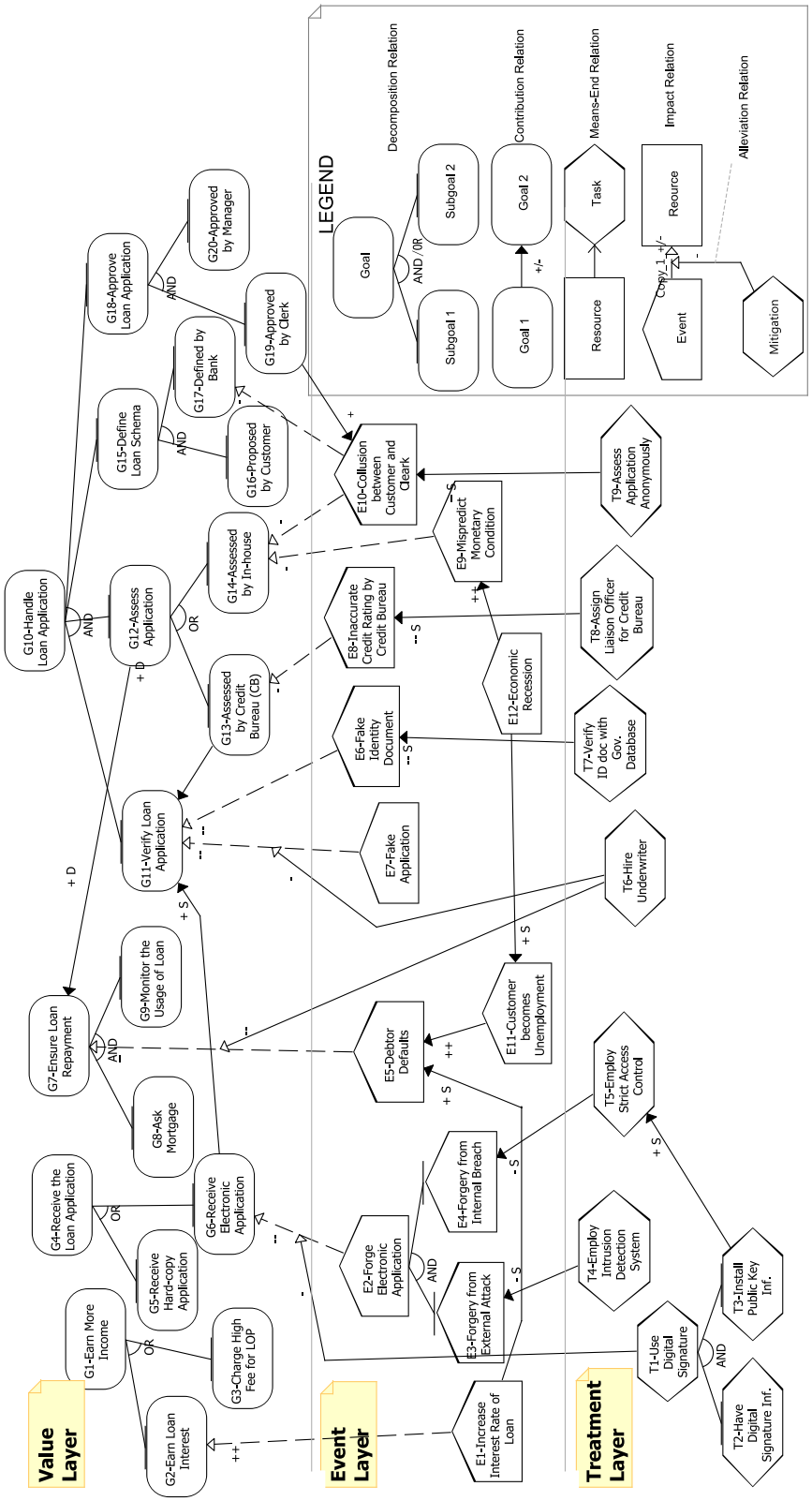


Figure 7.1: The GR model of LOP scenario

- $desired\_labels$ - interested in *partial* evidence for the satisfaction of top-goals G_1 and G_7 (i.e., $Sat(G_1) = P$, $Sat(G_7) = P$) whilst we insist on *full* evidence for top-goals G_4 and G_{10} (i.e., $Sat(G_4) = F$, $Sat(G_{10}) = F$);
- acc_risks - no risk for G_1 and G_7 (i.e., $Den(G_1) = N$ and $Den(G_7) = N$) while we allow *partial* evidence for the denial of G_4 and G_{10} (i.e., $Den(G_4) = P$ and $Den(G_{10}) = P$);
- $input\_goals$ - G_2 , G_3 , G_5 , G_6 , G_8 , G_9 , G_{11} , G_{13} , G_{14} , G_{16} , G_{17} , G_{19} , G_{20} as illustrated in Table 7.2 column "Goal-In";
- $event\_labels$ as illustrated in Table 7.2 column "Event-In".

Require: gr_model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$
label_array desired_labels, input_goals, acc_risks, event_labels
1: solution_array solution {solution that has already encompassed risks and necessary countermeasures}
2: alt_solution $\leftarrow Backward\_Reasoning(\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle, desired\_labels,$ **nil**, $input\_goals)$
3: candidate_solution $\leftarrow Select\_Can\_Solution(alt\_solution)$ {candidate_solution $\subseteq$ alt_solution}
4: **for all** $S_i \in candidate\_solution$ **do**
5: cur_labels $\leftarrow Forward\_Reasoning(\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle, \langle S_i, event\_labels, \mathbf{nil} \rangle)$
6: **if** $Satisfy(cur\_labels, desired\_labels, acc\_risks)$ **then**
7: $add(solution, \langle S_i, \mathbf{nil}, Calc\_Cost(S_i, \mathbf{nil}) \rangle)$
8: **else**
9: max_labels $\leftarrow Backward\_Reasoning(\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle, desired\_labels, acc\_risks, goals(S_i))$
10: rel_treatments $\leftarrow Find\_Treatments(\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle, cur\_labels, max\_labels)$
11: **for all** $C_j \in 2^{rel\_treatment}$ **do**
12: cur_labels $\leftarrow Forward\_Reasoning(\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle, \langle S_i, event\_labels, C_j \rangle)$
13: **if** $Satisfy(cur\_labels, desired\_labels, acc\_risks)$ **then**
14: $add(solution, \langle S_i, C_j, Calc\_Cost(S_i, C_j) \rangle)$

Algorithm 7.1: Risk Assessment and Treatment Process

The process starts from to seek all of possible assignment values of evidence for the input goals that satisfy the desired values ($desired\_labels$). It is done using $Backward\_Reasoning$ (line 2), explained in Chapter 5 - Section 5.4, which amounts to the top-down reasoning mechanism proposed in [181]. Here the GR model is encoded into satisfiability formulas and $desired\_labels$ as Φ_{val}, then a SAT solver is used to enumerate all possible assignments of input-goals that can satisfy $desired\_labels$ of the top goals. In this step, the use of backward reasoning is limited to the value layer (i.e., not considering the relations with the other two layers). In our case study, to achieve $desired\_labels$ we consider 8 alternative solutions as presented in Table 7.1.

Essentially, these alternatives prescribe for the *full* SAT evidence for all goals except for G_2 , G_3 , G_8 , and G_9 which might be *partial* one. Among these alternatives, the analysts might select some on the basis of some criteria (e.g., maximum-cost for fulfilling all input goals). Suppose that the analysts decide to consider alternatives with a cost no more than 30 as $candidate\_solution$ (S4, S6, S7, and S8 in our case – see Table 7.1). Of course, other criteria can be adopted for the selection of the solution. As discussed in [52], for instance we could also evaluate the effects of the solution over some qualities (softgoals).

Input Goal	Cost	S1	S2	S3	S4	S5	S6	S7	S8
G2-Earn Loan Interest	3	X		X		X		X	
G3-Charge High Fee for LOP	2		X		X		X		X
G5-Receive Hard-copy App.	5	X	X			X	X		
G6-Receive Electronic App.	3			X	X			X	X
G8-Ask Mortgage	2	X	X	X	X	X	X	X	X
G9-Monitor Usage of Loan	4	X	X	X	X	X	X	X	X
G11-Verify Loan Application	3	X	X	X	X	X	X	X	X
G13-Assessed by CB	10	X	X	X	X				
G14-Assessed by In-house	8					X	X	X	X
G16-Proposed by Customer	1	X	X	X	X	X	X	X	X
G17-Defined by Bank	3	X	X	X	X	X	X	X	X
G19-Approved by Clerk	1	X	X	X	X	X	X	X	X
G20-Approved by Manager	1	X	X	X	X	X	X	X	X
Cost		33	32	31	30	31	30	29	28

Table 7.1: Cost of Alternative Solutions

Each $candidate\_solution$ is now evaluated against risks and possible countermeasures are introduced in case the risk is unacceptable (line 4-14). Let's concentrate on S4 with the initial assignment for input goals as reported in Table 7.2 (column "Goal-In"). The analysts assess whether the $candidate\_solution$ (e.g., S4), together with risks in the event layer ($event\_labels$), can still lead to desired levels of evidence for top goals. To accomplish this, $Forward\_Reasoning$, explained in Chapter 5 - Section 5.3, is used to propagate evidence labels (from input goals and events) as reported in column "Event-In" throughout the GR model.

Later, $Satisfy$ compares final evidence labels for top goals with those desired. If DEN values for top goals are equal/less than the maximum risk admitted (acc_risks) and SAT values for top goals are equal/greater than the desired values ($desired\_labels$), then the $candidate\_solution$ is considered as the $solution$ and its cost is calculated (line 7). Otherwise, countermeasures must be introduced in the $candidate\_solution$ (line 10-14). In this case, S4 is still too risky, for instance, G_4 has *full* DEN evidence, while the acceptable value is at most *partial*. It happens since goal G_4 is satisfied through the achievement of G_6, which has *full* DEN evidence (i.e., $Den(G_6) = F$). We have a similar situation for G_7 and G_{10}. To make S4 acceptable, possible sets of treatments must be introduced to reduce the DEN evidence.

In order to define possible countermeasures, the analyst first requires to calculate the maximum DEN values of input goals (max_labels) so that it results in acceptable DEN values for top goals. In other word, we need to find a set of countermeasures able to mitigate risk, so that we end up with acceptable risk levels (acc_risks). The calculation of max_labels values is done using $Backward\_Reasoning$ (line 9) with considering the goals in the $candidate\_solution$ as input goals, and acc_risks and $event\_labels$ as constraints for the DEN value of input goals and events respectively. By having two sets of values (i.e., cur_labels and max_labels), analysts can identify which input goals are at risk and require some treatments. Essentially, treatments aim at mitigating the events so that they propagate less DEN evidence. $Find\_Treatments$ (Algorithm 7.2) is then used to enumerate all possible sets of treatments that can mitigate risks to acceptable levels. $Find\_Treatments$ compares DEN from the max_labels and the one in cur_labels to identify which goals are overvalued in their DEN value. For instance, given S4 the max_label of goal G_6 is $Den(G_6) = P$, whereas its cur_labels is $Den(G_6) = F$ (see Table 7.2 column

Require: gr_model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$, label_array current, label_array max
1: array rel_events, rel_impact, rel_treatments
2: **for all** $c_i \in current \wedge isGoal(c_i)$ **do** {find all events that threaten the value layer}
3: **if** $c_i.den > max_i.den$ **then**
4: tmp_events $\leftarrow related(\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle, c_i, ' impact')$
{find all related events to the goal c_i}
5: $add(rel\_events, tmp\_events)$
6: tmp_impact $\leftarrow R_k \in \mathcal{I}$ s.t. $source(R_k) = e_j \wedge target(R_k) = c_i$
{identify severity/impact relation of the event}
7: $add(rel\_impacts, tmp\_impact)$
8: **for all** $e_i \in rel\_events$ **do** {find all possible treatments to reduce the likelihood of the events}
9: tmp $\leftarrow related(\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle, e_i, ' negative - contribution')$
{treatments for reducing likelihood}
10: $add(rel\_treatments, tmp)$
11: **for all** $i_i \in rel\_impact$ **do** {find all possible treatments to alleviate the severity of the events}
12: **for all** $R_j \in \langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$ s.t. $target(R_j) = i_i$ **do** {identify all alleviation relations}
13: $add(rel\_treatments, source(R_j))$
14: **return** rel_treatments

Algorithm 7.2: Find_Treatments

"Event-Out"). The algorithm identifies related events (rel_events) that may cause this level of risk.[3] In our case, the relevant events for G_6 are E_2 , E_3 , and E_4 . Once we have identified the rel_events, we can identify the treatments ($rel\_treatments$) that can mitigate these events (line 8-13).

As discussed earlier, mitigation may reduce likelihood and/or severity of a risk. In the case of E_2 , E_3 , and E_4 , the treatments that reduce the severity are TR_2 and TR_3 , while TR_4 and TR_5 reduce likelihood. Moreover, analysts should aware that there could be the case where a treatment in $rel\_treatments$ has an overlapped effect in mitigating risks with other treatments. Thus, we evaluate each subset of $rel\_treatments$ to check whether it is sufficient to mitigate the risks, such that they (i.e., $candidate\_solution$, subset of $rel\_treatments$) satisfy the evidence values specified in $desired\_labels$ and acc_risks. In this way, we are able to seek the strategy which is the most cost-effective. Moreover, through this evaluation, analysts can estimate the side-effects of additional treatments to the value layer besides their effects to the event layer. If the $desired\_labels$ and acc_risks are achieved then the treatments and the input goals can be added to the $solution$ and the cost is calculated (Algorithm 7.1 line 14).

In this exercise, to make S4 acceptable we consider only some possible sets of treatments, namely C1, C2, C3, and C4 in Table 7.3. As reported in column "Treat-Out" in Table 7.2, C1 satisfies again the desired values for top-goals (DEN values for G_4 and G_{10} are now *partial*, while goal G_7 has no evidence for denial). However, the adoption of C1 introduces an additional cost for S4 that is now 30+13=43.

7.2 Business Solution Evaluation by means of Risk

The definition of metrics, for evaluating candidate solutions, is critical for most organisations before they implement the solutions. This is, for instance, the case for a consultant (i.e., Think3 Inc.) where is required to supply product development solutions, consulting services, and customer care for the optimisation of Small and Medium Enterprises' (SMEs) product development processes. For the success of

[3] $related$ is meant to enumerate all nodes that are reachable from a given node in $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$ through a particular type of relation.

	Goal			Event			Treat.		
	In	Out		In	Out		In	Out	
	S	S	D	S	S	D	S	S	D
E1-Increase Interest Rate of Loan	-	-	-	-	-	-	-	-	-
E2-Forge Electronic Application	-	-	-	-	F	-	-	F	-
E3-Forgery from External Attack	-	-	-	F	F	-	F	F	-
E4-Forgery from Internal Breach	-	-	-	F	F	-	F	F	P
E5-Debtor Defaults	-	-	-	-	P	-	-	P	-
E6-Fake Identity Document	-	-	-	F	F	-	F	F	F
E7-Fake Application	-	-	-	P	P	-	P	P	-
E8-Inaccurate Credit Rating by CB	-	-	-	P	P	-	P	P	-
E9-Mispredict Monetary Cond.	-	-	-	F	F	-	F	F	-
E10-Collusion Customer-Clerk	-	-	-	P	P	-	P	P	-
E11-Cust. becomes unemployment	-	-	-	P	P	-	P	P	-
E12-Economic Recession	-	-	-	-	-	-	-	-	-
G1-Earn More Income	-	P	-	-	P	-	-	P	-
G2-Earn Loan Interest	-	-	-	-	-	-	-	-	-
G3-Charge High Fee for LOP	P	P	-	P	P	-	P	P	-
G4-Receive the Loan Application	-	F	-	-	F	**F**	F	F	**P**
G5-Receive Hard-copy App.	-	-	-	-	-	-	-	-	-
G6-Receive Electronic App.	F	F	-	F	F	**F**	F	F	**P**
G7-Ensure Loan Repayment	-	P	-	-	P	**P**	-	P	**-**
G8-Ask Mortgage	F	F	-	F	F	-	F	F	-
G9-Monitor the Usage of Loan	P	P	-	P	P	-	P	P	-
G10-Handle Loan Application	-	F	-	-	F	**F**	-	F	**P**
G11-Verify Loan Application	F	F	-	F	F	**F**	F	F	**P**
G12-Assess Application	-	F	-	-	F	**P**	-	F	-
G13-Assessed by Credit Bureau	F	F	-	F	F	**P**	F	F	-
G14-Assessed by In-house	-	-	-	-	-	-	-	-	-
G15-Define Loan Schema	-	F	-	-	F	P	-	F	P
G16-Proposed by Customer	F	F	-	F	F	P	F	F	P
G17-Defined by Bank	F	F	-	F	F	-	F	F	-
G18-Approved Loan Application	-	F	-	-	F	-	-	F	-
G19-Approved by Clerk	F	F	-	F	F	-	F	F	-
G20-Approved by Manager	F	F	-	F	F	-	F	F	-
T1-Use Digital Signature	-	-	-	-	-	-	-	P	-
T2-Have Digital Signature Inf.	-	-	-	-	-	-	P	P	-
T3-Install Public Key Inf.	-	-	-	-	-	-	P	P	-
T4-Employ Intrusion Det. Sys.	-	-	-	-	-	-	-	-	-
T5-Employ Strict Access Control	-	-	-	-	-	-	-	P	-
T6-Hire Underwriter	-	-	-	-	-	-	P	P	-
T7-Verify ID doc with Gov. DB	-	-	-	-	-	-	F	F	-
T8-Assign Liaison Officer for CB	-	-	-	-	-	-	P	P	-
T9-Train Internal Actuary	-	-	-	-	-	-	-	-	-
T10-Assess App. Anonymously	-	-	-	-	-	-	-	-	-

Table 7.2: SAT-DEN Values of S4-alternative and C1-treatments

		S4				S6	S7/S8			
Treatment	Cost	C1	C2	C3	C4	C5	C6	C7	C8	C9
T02	2	X			X		X			X
T03	2	X		X	X		X		X	X
T04	1		X	X	X			X	X	X
T05	2		X		X			X		X
T06	4	X	X	X	X	X	X	X	X	X
T07	3	X	X	X	X	X	X	X	X	X
T08	2	X	X	X	X					
T09	3					X	X	X	X	X
T10	2					X	X	X	X	X
Total Cost		13	12	12	16	12	16	15	15	19

Table 7.3: Cost of Possible Treatments

provided solutions, they must be evaluated the solution prior their deployment to the clients (i.e., SMEs). Moreover, Think3 has to provide business solutions consistently with the business model/strategy (e.g., product-oriented, order-oriented) chosen by the client.

Based on the experience of Think3, often they need to evaluate alternative solutions for various clients which have similar organisational-setting. Therefore, it is beneficial when the can capture their knowledge into a generic model so that it can be reused in the future. In our case, we aim to develop a "generic" organisational-setting of Think3's clients (i.e., manufacturing SMEs). In fact, most of manufacturing SMEs have similar organisation settings though they have different business models. For instance, manufacturing SMEs, Think3's client are distinguished into two categories: The first category of SMEs is mainly involved in the production of a fixed set of products. The second category concerns the production of a variable set of products depending on the specific requirements given by customers. Both categories have different business model in manufacturing their products. However, both categories are founded on a similar organisational setting. In manufacturing SMEs, the production planning division is typically appointed to define the most suitable solution for the product. To achieve this goal, that division needs to consider the feedback from different sources, such as manufacturing, customer care, and marketing divisions. Accordingly, the production planning division depends on all those divisions for the feedback necessary to come up with the most suitable solution. A failure in fulfilling the assigned duty will affect the goal of the production planning division. Thereby, the last needs to ensure that each division provides the necessary feedback. Moreover, there is a situation where an appointed division may argue that they are not at risk, but in the point of view of the planning division is otherwise. As a consequence, the production planning division adopt additional safeguards to secure its objective. These issues demand the use of risk assessment frameworks that is able to model the generic business setting in an organisation, and evaluate risks on the basis of the model.

Such requirements spur the application of the GR framework that has been developed to assess and treat risks in a given organisational setting to solve such challenges. Initially, we applied the Tropos [38] (including its descendants - the GR framework) for the elicitation of such requirements on the basis of Think3's clients [121]. Starting from that work, we have identified the risks affecting intra-manufacturing SMEs and the treatments that are usually adopted in industrial practices. However, we discover there are some particularities in each category of manufacturing SMEs. As regards product-oriented SMEs, risk analysis has to deeply investigate aspects concerning how to "produce the product right", while for order-oriented ones the analysis considers the issues related to producing the "right product". The

former considers the compliance of the final product with its requirements, whereas the latter focuses on the acceptance of the final product by the client. In a product-oriented SME, the acceptance process of the products by clients is not feasible due to the large set of products and clients to be considered. Similarly, it does not make sense for an order-oriented company to evaluate the compliance with the requirements of a product without considering its acceptance by a specific client.

Since our goal is to perform risk analysis on the core requirements for SMEs, the GR model presented in Figure 7.2 is essentially a generalisation of organisational model for both product- and order-oriented SMEs. Such a model presents how the divisions within an SME collaborate together achieving the objectives of enterprise and managing the level of risk.

In this case study, we apply the similar analysis techniques as presented in the previous section. However, the analysis requires different sets of inputs depending on the organisational setting and the business model of the SME. For instance, some events can have a different likelihood depending on the strategy adopted by the SME, such as the event likelihood of *modification order* in order-oriented SMEs considered more likely compared to the product-oriented one. Moreover, for a product-oriented SME the achievement of the goal *reduce time-to-market* is considered much more critical than the order-oriented SMEs because of competition with other SMEs. Though the analysis is done using the same GR model, it does not imply it will result in the same evaluation results. The details about the application of the GR framework in this case study can be found in [14, 15, 121].

Evaluation

We have applied the GR framework to analyse intra-manufacturing organisations (presented in [14, 15]). Specifically, we have assessed the risks affecting intra-manufacturing organisations when they employ different business strategies and identified the measures to be adopted in order to mitigate such risks with respect to the selected strategy. From this exercise, our industrial partner benefits from the reuse of the "generic" organisational-setting (i.e., a GR model) so that it reduces the efforts in performing the evaluation of business solutions, and improves the quality of evaluation by learning from past experiences.

Moreover, this case study is also useful to evaluate the expressiveness of the GR modelling language and to validate the formal framework against industrial case studies (i.e., the evaluation of business solutions in manufacturing SMEs). In fact, in [15] we adjust the framework in order to work with likelihood instead of the evidence value. It is because our industrial partners found more intuitive and practical to provide values in term of likelihood of events instead of the evidence value.

7.3 Business Continuity Analysis in an Organisation

In this section, we demonstrate how the GR framework can support the analysis of business continuity from a socio-technical perspective. Unlike "conventional" risk management, business continuity analysis aims at ensuring the continuity business under disruptions. In other words, the organisation allows some disruption to occur as far as it does not result in the discontinuity to the business service. Typically, the organisation defines its plan, namely Business Continuity Plan (BCP) or Electronic Data Processing (EDP) contingency plan, to react responding the disruptions before their impacts are materialised in the business level. For instance, the credit rating service might be disrupted as far as it can be recovered within specified time before the service is used for some business processes. To allow continuity analysis, analysts need a framework that is being able to support the following tasks:

1. analyse how an organisation generates the values and their related risks;

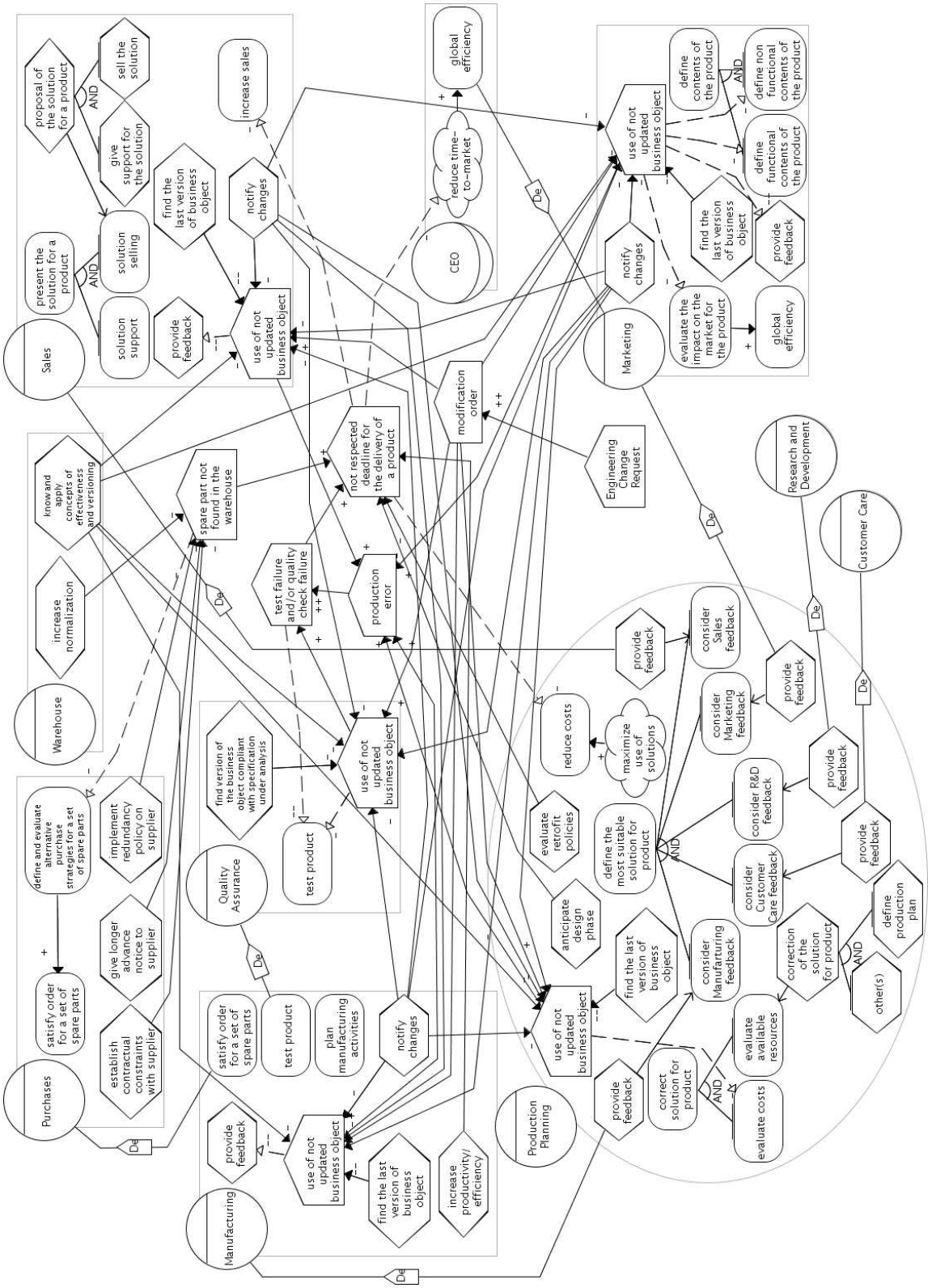


Figure 7.2: Intra-Manufacturing Organisation Model including Events and Risk Treatments

2. define additional measures that are available to ensure the continuity of business in generating the values;

3. define the most cost-effective mitigation plans.

Previously, we have explained the Goal-Risk framework which is developed to analyse the business-objectives, and to operationalise them at more detail level (i.e., tasks model, business process [222]) and, finally, to identify the required artefacts to execute the processes. All those constructs, so called business objects, extends the notion of asset, as something that is valuable for an organisation, beyond only artefacts. Though the GR framework is able to assists risk management process, it lacks some supports for the continuity analysis.

In this section, we extend the GR framework to support the analysis of business continuity of an organisation. We use *Loan Origination Process* (LOP), taken from the European project SERENITY [4], to illustrate our approach. It starts by receiving a loan application and ends, possibly, with the loan approval.

However, uncertain events (i.e., threats, un/intentional events, incidents, risks) may affect the availability of some assets. For instance, events like *computer virus outbreak*, *database failure*, *the outage of national identity service* are considered as disruptions for the loan department. Essentially, these disruptions are hard, or impossible, to avoid, but they might be still acceptable if their disruptions vanish within an acceptable period (called Maximum Tolerable Period of Disruption-MTPD). For an example, the goal of *receive loan* is still satisfied though the *credit rating service* is disrupted for 2 hours due to some failures. It is because the bank processes the loan application in every 3 hours. To maintain the MTPD, all responsible stakeholders need to establish a contingency plan in case their assets are disrupted. In this work, we limit ourselves by addressing only availability issues. The plan (BCP), typically, consists of the Recovery Time Objectives (RTOs) representing the objective of an asset recovery time in case the asset is disrupted. For instance, the IT department ensures that the database of loan application system will be recovered within 1 hour after the disruption. Under a given risk model, analysts should assess the sufficiency of RTOs to meet the MTPDs. In the case of insufficiency, additional measures need to be introduced. Moreover, these additions should be analysed carefully before their adoption because they introduce additional costs, and often they may introduce other kind of problems to the system.

7.3.1 An Extension of the GR Framework for Continuity Analysis

To model and assess BCPs, we need to analyse 1) business objectives and their realisations (process and artefacts), 2) interdependencies among assets, and 3) the level of risk that threatens business objectives, directly or indirectly. However, the GR modelling framework (presented in Chapter 4) is able to address the 1st and 3rd issues, but it is still insufficient to capture the interdependencies among assets that support an organisation. Hence, we here extend some of GR constructs (e.g., needed-by, dependency) to be able to support continuity analysis.

Essentially, *needed-by* and *dependency* relations are able to capture this situation, but they are insufficient. It is because they assume the failure of the source node will affect the target node immediately which is not always the case in reality. For instance, the failure of *credit rating service* computing the credit ration of customers does not result in immediate failures to the LOP business process, because loan applications are processed in batch every 3 hours. Therefore, *needed-by* and *dependency* relations need to be extended with the notion of time-dependency as proposed in the Time Dependency and Recovery

[4] http://www.serenity-project.org/

(TDR) model [228]. The time-dependency denotes the elapsed time that is tolerable by the target node in case the source node fails.

Let's assume there are two business objects: the service loan processing *(N_1)* and credit rating service *(N_2)*, previously we capture such relationship by indicating $N_2 \overset{n}{\longmapsto} N_1$ which implies N_2 is needed by N_1 without any notion of time. Such a relation implies the failure N_2 will immediately cause N_1 to fail. However, it is not the case because for N_1 concern N_2 may fail as far as it is recovered within 2 hours. To capture such situation, we annotate *needed-by* and *dependency* relations with t indicating the tolerable outage period.

Definition 7.1. `Needed-by` is indicated as $N_2 \overset{n(t)}{\longmapsto} N_1$ where $t > 0$ is the tolerable time for N_1 before it follows N_2 to fail.[5]

Definition 7.2. `Dep` is indicated as $A_1 \overset{D_t(O)}{\longmapsto} A_2$ where $t > 0$ is the time that is tolerable by Actor A_1 about the availability of the dependum-O by Actor A_2.[6]

The former captures an inter-dependency between resources in supporting a task execution within a single actor's perspective, while the latter models an inter-dependency between actors about the availability of the dependum in a multi-actor setting.

Following the "generic" methodology (in Chapter 6), analysts model business objectives of the stakeholders and their realisation in terms of tasks and resources in the value layer. Relevant risks are modelled and quantified their severity and likelihood in the event layer, and finally some available treatments are captured in the treatment layer. Some particular steps are different from the "generic" methodology. To analyse BCP, a GR model needs also to capture inter-dependencies between resources in supporting a task execution. We have extended *needed-by* relation with t indicating the maximum disruption period that is tolerable by dependent resources. For example, in Figure 7.3 Secure desktop client for Loan Agent *(R_{01})* is needed by the task T_{01} (i.e., the time-dependency is 20 minutes) and R_{01} requires to access database of loan applications *(R_{04})* (i.e., the time-dependency is 2 minutes). The disruption of R_{01} will not result in the failure of T_{01} if it happens not more than 20 minutes. For given a GR model-$\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$ and desired MTPDs, we will compute the Maximum Recover Time (MRT) of each assets. If RTOs of all assets, in the BCP, are less-or-equal to its MRTs then we can conclude the BCP is sufficient to guarantee the continuity of the business.

However, the fulfilment of goals, the execution of tasks, and the provision of resources might be disrupted by the occurrence of uncertain events. Here, an event is characterized into two attributes: likelihood (λ) and its consequences [109].[7] We here quantify *likelihood* in terms of the number of occurrences within a specific period (e.g., a year) judged by experts.[8] For instance, if experts judge that the likelihood of an event is 0.1, and then they refer to the event took place once in 10 years. The severity of consequences is captured by the extent of *impact* relations. If the risks is too high then some treatments are needed either to reduce the likelihood (i.e., reduce the frequency of unacceptable disruptions) or to alleviate the severity (i.e., shorten the length of disruption).

Initially, the GR framework specifies `Impact` into four levels in $\{++_i, +_i, --_i, -_i\}$. To be able to reason about business continuity, we revisit the definition of `Impact` since it, initially, refers to the propagation of evidence from an event to an business object and not the length of a disruption.

[5] We assume here the time unit of t is minute

[6] In this chapter, we concentrate on a single actor perspective. Moreover, this extension requires us to make some adjustments to the pre-processing procedure of $\langle \mathcal{N}, \mathcal{R}, \mathcal{I}, \mathcal{S} \rangle$ into $\langle \mathcal{N}, \mathcal{R}, \mathcal{I} \rangle$ presented in Chapter 5-Section 5.2

[7] We here consider only events with negative consequences (i.e., risks, threats, incidents).

[8] The model allows analysts to represent the likelihood in terms of *Probability Distribution Function* for a better result (i.e., precision), but it requires more complex mathematical computation

Definition 7.3. `Impact` is indicated as $E \overset{t_i}{\longmapsto} N$ where $t > 0$ is the length of disruption suffered by N in case E occurs.

Unlike, the previous `Impact` (presented in Chapter 5) we here consider only `Impact-` and not the opportunity ones. In addition, we also need to revisit relations that are used to capture mitigation actions. First, we need to revise the definition of `Alleviate` capturing the effect of treatments in reducing the severity of an event (i.e., shorten the disruption time).

Definition 7.4. `Alleviate` is indicated as $TR \overset{x_a}{\longmapsto} [E \overset{y_i}{\longmapsto} N]$ where $-1 \leq x \leq 0$ representing the reduction percentage of the length of disruption by E to N.

We also redefine the `Contr` relations that relate a treatment to an event as a means to reduce its likelihood.

Definition 7.5. `Contr` in that circumstance is define as $TR \overset{x_c}{\longmapsto} E$ where $-1 \leq x \leq 0$ representing the percentage of E likelihood reduction.

7.3.2 Continuity Analysis

After analysts have modelled the business-setting of an organisation, possible risks, and available treatments, one can perform the following analyses to manage the business continuity:

- *Treatments Analysis*, intended to seek all possible sets of treatments that are able to mitigate the risk so that the continuity is guaranteed. Analysts will choose the most adequate mitigation following some criteria (e.g., additional costs, possible side-effects);
- *Cost-Benefit Analysis*, aim to identify the most cost-effective treatments to reduce the loss caused by business discontinuity.

Moreover, the cost-benefit analysis is useful when there is no possible set of treatments that is able to reduce the level of risk until the acceptable level. In this case, analysts typically choose the most cost-effective set of treatments. However, both analyses can be performed complementary one to another.

Both analyses require the following inputs:

1. A GR model - see Figure 7.3;
2. Acceptable risk, represented in terms of pairs Maximum Time Period of Disruption (MTPD) and Maximum Likelihood (Max.λ) of unacceptable disruption for each top goal (e.g., $MTPD(G_{01}) =$ 60 minutes - Max.$\lambda(G_{01}) = 2$, $MTPD(G_{04}) = 120$ minutes-Max.$\lambda(G_{04}) = 2$) - see Table 7.4;
3. "Significant" business objectives, which are defined as top-level goals and other subgoals that the stakeholders believe to be important for the organisation. For each of these goals, we specifies its utility for the organisation (e.g., $Utility(G_{01}) = 80$, $Utility(G_{02}) = 50$) - see Table 7.4;[9]
4. Likelihood of events (e.g., $\lambda(E_{01}) = 12$, $\lambda(E_{03}) = 3$) - see Table 7.5;
5. Treatments costs (e.g., $Cost(TR_{01})$=200, $Cost(TR_{02})$=70) - see Table 7.6;
6. RTOs (we assume that they have the same value as MRTs).

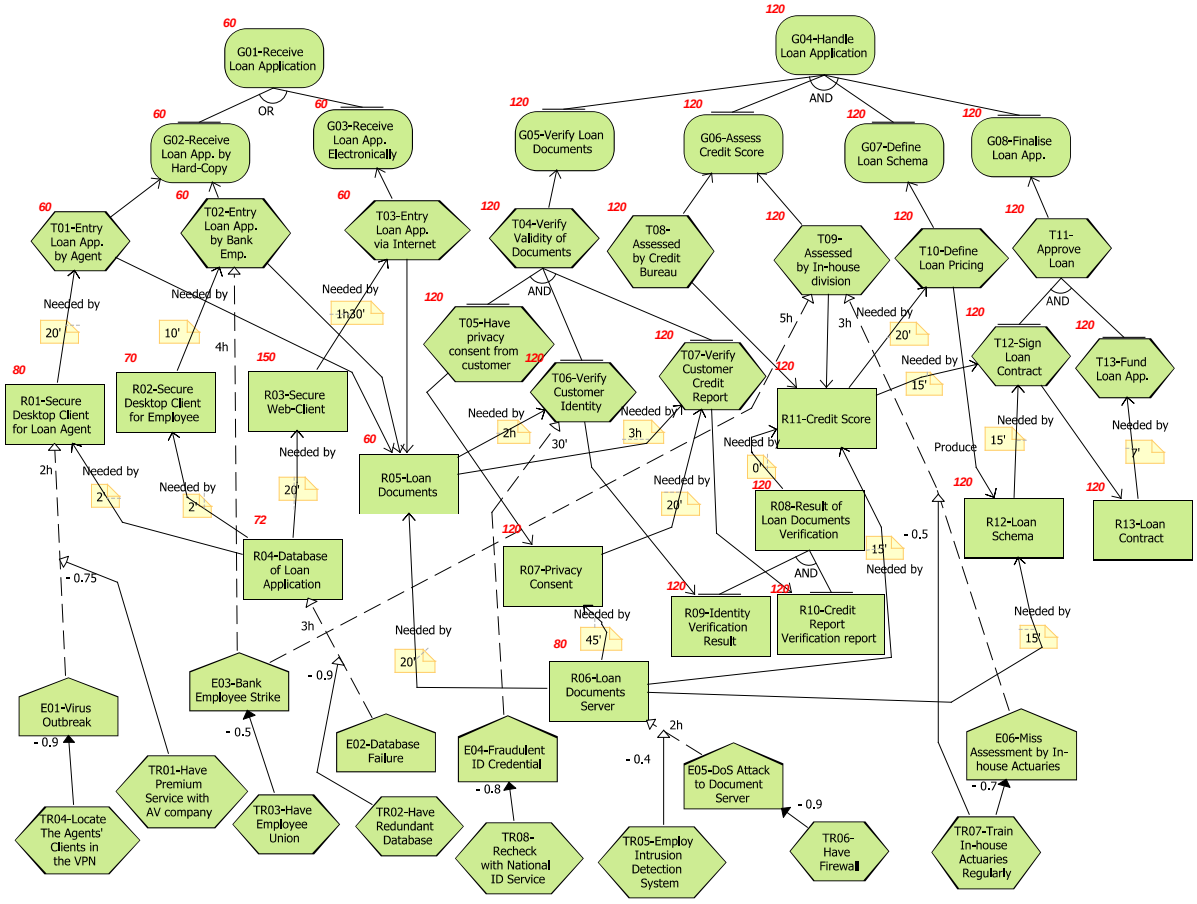


Figure 7.3: The Model for Assessing the BCP of Loan Originating Process

For a given GR model $\langle \mathcal{N}, \mathcal{R}, \mathcal{I}, \mathcal{S} \rangle$, we compute the required MRT for each business objects (i.e., goals, tasks, resources) as follow:

1. Build Time-Dependency graph $(E, \rightarrow)$ where

   - $E = \{(a, o) | (a, o, p) \in \mathcal{N} \wedge o \in \mathcal{O}\}$;
   - $\rightarrow_r$ is constructed from $\mathcal{R}$(i.e., $(N_1, \ldots, N_n) \overset{r}{\longmapsto} M$) where $N_1, \ldots, N_i, M \in E$

$$\rightarrow_r = \bigcup_{\mathcal{R}} \begin{cases} N \overset{t}{\longmapsto} M, & \text{if } r = \texttt{Needed-by}^{10}; \\ N \overset{0}{\longmapsto} M, & \text{if } r = \texttt{Means-end} \wedge M \in \mathcal{G}; \\ M \overset{0}{\longmapsto} N, & \text{if } r = \texttt{Means-end} \wedge M \in \mathcal{M}; \\ \bigcup_{i=1\ldots n} (N_i \overset{0}{\longmapsto} M), & \text{if } r = \texttt{Dec}. \end{cases}$$

and from $\mathcal{S}$ $a_1 \overset{D_t(o)}{\longmapsto} a_2$ and $(a_1, o), (a_2, o) \in E$

$$\rightarrow_s = (a_2, o) \overset{t}{\longmapsto} (a_1, o) \qquad \text{if } s = \texttt{Dep}$$

[9] We quantify the utility in the range $[0, 100]$.

Goals		MTPD(G)	Max. $\lambda(G)$	$Utility(G)$
G01	Receive Loan Application	60	2	80
G04	Handle Loan Application	120	2	40
G02	Receive Loan App. by Hard-Copy			50

Table 7.4: The Inputs of Top Goals and "Significant" Goals

Event-Src		Target	λ	Impact
E01	Virus Outbreak	R01	12	2h
E02	Database Failure	R04	10	3h
E03	Bank Employee Strike	T02	2	4h
E03	Bank Employee Strike	T09	2	5h
E04	Fraudulent ID Credential	T06	24	30'
E05	DoS Attack to Doc. Server	R06	20	2h
E06	Miss Ass. In-house Actuaries	T09	4	3h

Table 7.5: Likelihood and Impact of Uncertain Events in The LOP scenario

2. Compute MRT for each $N \in E$

$$MRT(N) = \begin{cases} MTPD_n, & \text{if } N \text{ is a top-goal;} \\ min\{MRT(M) + t) | N \xmapsto{t} M\}, & \text{otherwise.} \end{cases}$$

Treatment Analysis

The overall analysis (Figure 7.4) goes as follows:

1. Propagate the impact of uncertain events throughout the model. It is done by means of the adaptation of $Forward\_Reasoning$ introduced in Chapter 5;

2. Evaluate the risk level of top-goals by computing their Time Period of Disruption (TPD) and the likelihood of unacceptable disruption;

3. Decide whether the risk level is acceptable for the organisation or not. The notion of acceptable risk refers to the situation where the likelihood of unacceptable of disruption (i.e., $TPD(G) > MTPD(N)$) is less then Max. $\lambda(N)$;

4. Introduce treatments if the risk is unacceptable. $Find\_Treatments$ (detailed in Algorithm 7.2 in Chapter 7) is used to elicit possible alternative solutions for a given set of available treatments. In nutshell, that algorithm is an adaptation of the *greedy search* algorithm [173] that aims at suppressing the increase of costs because of new treatments. Notice in the worst case the algorithm needs to explore all possible subsets of treatments (i.e., $2^{|}treatments|$ sets) though it is hardly happened in practice.

Finally, the analysis results in several possible solutions consist of a set of treatment that satisfies the acceptable risk. Analysts, then, need to decide among possible solution based on some criteria (e.g., the cheapest, stakeholders' preference, company culture, etc.) which solution to implement.

In the context of *loan origination process*, we have modelled the organisation in terms of a GR model as depicted in Figure 7.3. We analyse two top goals for the bank: receive loan application *(G_{01})* and

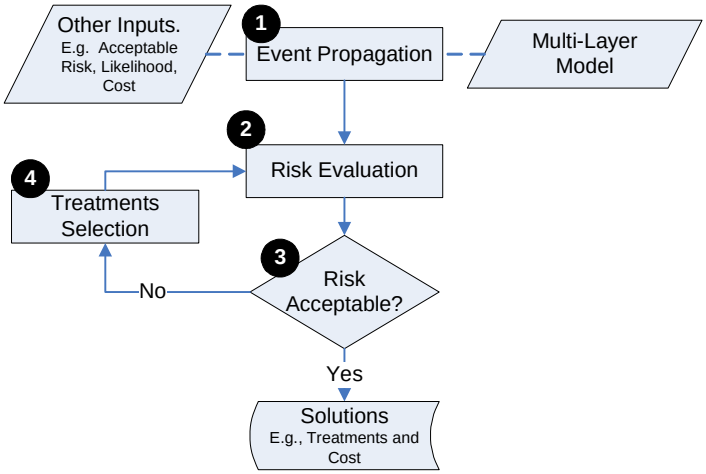


Figure 7.4: Treatment Analysis

Treatment		Cost	S_1	S_2	S_3	S_4	S_5
TR01	Have Premium Service with AV company	200	X			X	X
TR02	Have Redundant Database	50	X	X	X	X	X
TR03	Have Employee Union	100					
TR04	Locate The Agents' Clients in the VPN	90		X	X		
TR05	Employ Intrusion Detection System	30	X		X		X
TR06	Have Firewall	10		X		X	X
TR07	Train In-house Actuaries Regularly	70	X	X	X	X	X
TR08	Recheck with National ID Service	40					
Total Cost			350	220	240	330	360

Table 7.6: Total Cost of Possible Treatments

handle loan application *(G_{04})*. Suppose stakeholders expressed their acceptable risks (i.e., MTPD, Max.λ) for the two goals as indicate in Table 7.4. For given MTPDs, we compute the MRT of every asset (indicated in the upper-left of each construct in Figure 7.3). In Table 7.7 column "MRT", we present the Max.λ and MRT for each risky business object (i.e., business object with risks) required to satisfy the MTPDs of top-goals.

However, by considering the risks in Table 7.5 we realise that some risks cause the unacceptable disruption for the business. For instance, R_{01} must have at most 2 times of 80 minutes of disruption (MRT) in one year (see Table 7.7 column "MRT"). Unfortunately, the impact of E_{01} results in 12 times of 2 hours disruption, which is unacceptable (see Table 7.7 column "Init"). However, T_{02} , T_{09} , and T_{06} are acceptable because either they might have unacceptable disruption less than 2 times per year, or their disruption is less than the required MRT. To mitigate such risks, treatment analysis enumerates 81 possible solutions (i.e., sets of treatments) that can satisfy the stakeholders' inputs. For the sake of simplicity, we concentrate only on five of them, namely $S_1 - S_5$, as indicated in Table 7.6.

From Table 7.7 column "MRT", we can observe that the Max.λ-MRT for R_{06} is 80 minutes for 2 times per year. The application of S_2 results in the event E_{05} is being mitigated by TR_{06} into 2 hours for 2 times/year to R_{06} , which is acceptable by the stakeholders because the likelihood of the disruption is less-equal than Max.λ. In the case of S_5 which includes TR_{05} besides TR_{06} , it results in 72 minutes for 2 times/year. It implies the business is never discontinued because the TR_{05} can be

Event-Src	Target	MRT	Init	S_1	S_2	S_3	S_4	S_5
E01	R01	2-80’	12-2h	12-30’	1.2-2h	1.2-2h	12-30’	1.2-30’
E02	R04	2-72’	10-3h	10-18’	10-18’	10-18’	10-18’	10-18’
E03	T02	2-1h	2-4h	2-4h	2-4h	2-4h	2-4h	2-4h
E03	T09	2-2h	2-5h	2-5h	2-5h	2-5h	2-5h	2-5h
E04	T06	2-2h	24-30’	24-30’	24-30’	24-30’	24-30’	24-30’
E05	R06	2-80’	20-2h	20-72’	2-2h	20-72’	2-2h	2-72’
E06	T09	2-2h	4-3h	1.2-1.5h	1.2-1.5h	1.2-1.5h	1.2-1.5h	1.2-1.5h
Total Cost				350	220	240	330	360

Table 7.7: Risks at Initial and After Treatments Adoption in LOP Scenario

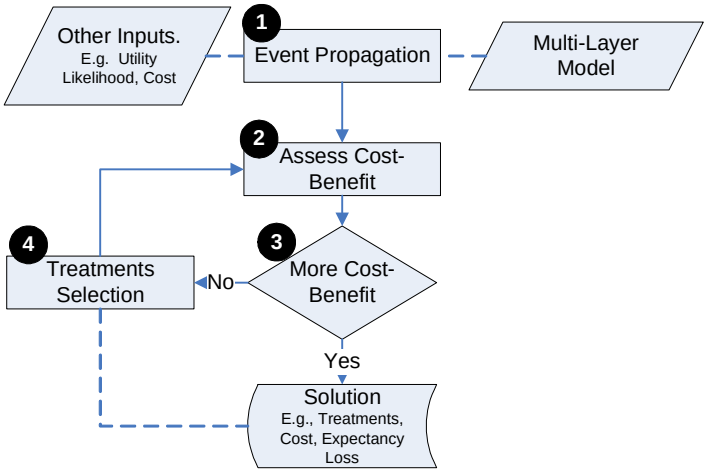


Figure 7.5: Cost-Benefit Analysis

recovered before the disruption appears in the business level. Each solution has a different cost and also a different effect on the reduction of risks, as presented in Table 7.7. Notice that all solutions ($S_1 - S_5$) produce an acceptable level of risk, but S_2 results being the cheapest solution. However, S_3 can be also a good candidate since it can reduce, further, the outage-period of R_{06} from 2 hours to 72 minutes with only a bit higher cost. Decision about S_2 or S_3 is now responsibility of analysts; they have to evaluate which one is the best suited for the organisation.

Cost-Benefit Analysis

Cost-benefit analysis is useful when analysts cannot find any possible combinations of treatments that are able to mitigate the risk until the acceptable level. In other words, it aims at finding the best solution from available treatments. We here define the notion of “best” as the most advantageous (i.e., cost effective) solution represented in terms of the ratio between benefit and cost (Equation 7.1)[11], while benefit is modelled as an inverse function of the total of loss expectancy.

[11] Analysts must adopt at least a treatment to mitigate risk and therefore the $Cost$ cannot be 0

$$ADV(S) = \frac{1}{\sum_{G \in significant\text{-}goals} LE(G) \times Cost(S)} \quad (7.1)$$

$$LE(G) = \lambda_{not-acceptable} \times Utility(G) \times [\![TPD(G) - MTPD(G)]\!]^{12} \quad (7.2)$$

Essentially, the loss is introduced when where the TPD of a business objective is greater than the specified MTPD, and it happens more often than Max.λ. Thus, in this framework the loss expectancy (LE) is calculated as multiplication of the likelihood of unacceptable disruption and the losses (i.e., the utility of the goal $\times$ the overhead of disruption period) - see Equation 7.2.

The overall process of cost-benefit analysis (in Figure 7.5) goes as follow:

1. select a set of treatments to be evaluated - S_i;
2. calculate the ADV for a given selected treatments;
3. compare the ADV of selected treatments with the best ADV of the candidate solution obtained from the previous evaluation - S;
4. if it is more advantageous then record S_i as the candidate solution (S), and back to Step 1 until all possible combination of treatments are evaluated.

Moreover, the notion of cost-benefit might be enriched by considering other factors (e.g., time of implementation, intangible values) besides only loss-expectancy and cost. To reduce the possible search space, the algorithm records the most cost-effective solution so far, and ignores the branch of search space, which is less beneficial than the recorded solution. The result of this analysis is only a solution that theoretically, based on the equation (7.1), is the most cost-effective solution. Typically, this type of solution would be easy to get an approval by the stakeholders because it proposes the set of treatments, which is the most cost-effective. Notice this process is an exhaustive process, because it requires exploring all possible subset of treatments. Hence, we recommend to perform it if there is no available set of treatments that can satisfy the acceptable risk, or one can perform this analysis over proposed alternatives resulted from the treatment analysis to evaluate which alternative is the most advantageous/cost-effective.

To demonstrate the cost-benefit analysis, we alter the previous case study by assuming now the stakeholders are more risk averse than in the previous case. MTPD for goals G_{01} and G_{04} are reduced to 2 and 50 minutes, respectively with the same Max.λ for both business objectives. Moreover, stakeholders here argue about the importance of subgoal G_{02} that can endanger the image of the organisation in case it fails (even if G_{01} is satisfied). We quantified the G_{02} utility as 50, which is slightly lesser than the utility for G_{01} ($Utility(G_{01}) = 80$). However, goal G_{04} $Utility(G_{04}) = 40$) results being less important than G_{01} and G_{02} since its failure will not be visible from outside of the organisation. The new MTPDs, consequently, require shorter MRTs for all business objects supporting them (Table 7.8 - Column "MRT"). Once we consider risks in Table 7.5, then we can see in Table 7.8 - Column "Init" that most of business objects (i.e., except T_{02} and T_{09}) are unacceptable. In fact, T_{02} and T_{09} are still acceptable because they fail only twice a year. Moreover, we also realise those treatments result in T_{06} being acceptable, because its TPD, due to E_{04} , is still smaller the MTPD. However, the treatment

[12] $[\![x]\!]$ is a function where $[\![x]\!] = x$ if $x > 0$; otherwise $[\![x]\!] = 0$

Event-Src	Target	MRT	Init	S_1	S_2	S_3	S_4	S_5
E01	R01	2-22'	12-2h	12-30'	1.2-2h	1.2-2h	12-30'	1.2-30'
E02	R04	2-14'	10-3h	10-18'	10-18'	10-18'	10-18'	10-18'
E03	T02	2-2'	2-4h	2-4h	2-4h	2-4h	2-4h	2-4h
E03	T09	2-50'	2-5h	2-5h	2-5h	2-5h	2-5h	2-5h
E04	T06	2-50'	24-30'	24-30'	24-30'	24-30'	24-30'	24-30'
E05	R06	2-22'	20-2h	20-72'	2-2h	20-72'	2-2h	2-72'
E06	T09	2-50'	4-3h	1.2-1.5h	1.2-1.5h	1.2-1.5h	1.2-1.5h	1.2-1.5h
Exceeded Disruption after Treatments								
R01				12-8'			12-8'	
R04				10-4'	10-4'	10-4'	10-4'	10-4'
R06				20-7'		20-7'		
Results								
Cost				350	220	240	330	360
LE				21920	4800	10400	16320	4800
ADV (in 10^{-7})				1.30344	9.4697	4.00641	1.8568	5.78704
Total Cost				350	220	240	330	360

Table 7.8: ADV of Possible Treatments in LOP scenario

analysis cannot be able to elicit any solutions that can satisfy defined acceptable risks (i.e., particularly for R_{01} , R_{04} , and R_{06}). We here consider previous candidate solutions (i.e., $S_1 - S_5$), then they are fail to meet the MRTs (Table 7.8 - Column $S_1 - S5$). In this situation, analysts might simply ignore this fact and accept the risk per se, or consider adopting the most advantageous solution.

Considering all the outcome of applying possible treatments (e.g., $S_1 - S_5$), we discover that by S_1 the system still suffers 12 times/year an outage of 30 minutes for R_{01} where the MRT of R_{01} is 22 minutes. In other words, the system might be discontinued for 8 minutes, 12 times/year (see Table 7.8 for the complete ones). Consequently, these outages will introduce a loss (expectancy) for G_{01} , G_{02} , and G_{04} , as defined in equation (7.2). For instance, the resulting loss expectancy for S_1 is 21920 with a cost of 350. Looking at Table 7.8, S_2 is considered as the most cost-effective solution with the lowest level of LE and the cheapest cost.

7.4 Concluding Remark

We here have presented some applications of the framework for managing risk in an organisation. Essentially, this chapter details three use cases: 1) the framework is used to assess the risk while analysing requirements in an organisation, and 2) the extended framework is used to assess the business continuity.

First, we have shown the capability of the framework to model, analyse, assess, and treat risks during the requirement analysis process. Moreover, in this case we proposed qualitative reasoning algorithms to analyse risk during the process of evaluating and selecting among alternative requirements. Hence, it assists analysts to avoid eliciting risky requirements, and also to reduce the number of requirement changes due to the revision of risky requirements or due to the introduction countermeasures. The performance of the risk assessment process depends on the number of treatments that are available to mitigate risks. Note that the assessment process still requires human intervention in selecting the alternatives to be analysed further. Actually, this step might be skipped, but it can significantly reduce the space of possible alternatives so that results in better performance for the overall risk assessment process. The final decision of choosing among possible strategies is left to the analysts. We also illustrated the usage of the frame-

work in order to evaluate several business solutions considering the organisational setting and business strategy. We used the intra-manufacturing scenario to demonstrate the usages of the framework. In fact, the framework is able to capture a "generic" organisation-setting, in terms of a GR model, in a particular domain application (i.e., manufacturing SMEs). The model later can be used by analysts to evaluate alternative business solutions by means of assessing and treating risks in an SME with considering the business model of the SME (e.g., product-oriented and order-oriented).

Second, we presented an extension of the framework to analyse the business continuity of an organisation. The framework models all levels of assets (e.g., objective, process, and artefact) that may be involved in the continuity of the business. In order to guarantee the continuity of business under uncertainty (e.g., incidents, attacks, human-errors, hardware-failures), analysts need to introduce a set of treatments to mitigate risks. Moreover, the framework proposed a two analyses: 1) treatment analysis which results in a set of possible alternatives to ensure the business continuity, and 2) cost-benefit analysis which aims to seek the most cost-effective set of treatments to maintain the business continuity. Though, this approach does not require very precise inputs (e.g., likelihood, time-dependency, etc.), we recommend to analysts to use the worst possible scenarios while assessing the inputs, though it means "overshooting risks".

The GR framework with all new features presented is supported by the S&D Tropos Tool.[13] This tool is an Eclipse plug-in developed to analyse security and dependability requirements of socio-technical systems. The tool provides requirements engineers with a graphical interface that allows them to draw GR models. It also supports the automatic transformation of GR graphical models into formal specifications for the automated analysis.

[13] Available on the web at `http://sesa.dit.unitn.it/sistar_tool/`.

Chapter 8

Forming an Organisation: a Risk-driven approach

In this chapter, we demonstrate the usage of the GR framework in synthesising the organisational-setting within a set of possible alternative designs. In the previous chapter, we have shown how to manage risks by analysing alternative requirements (i.e., OR decomposition), and integrating them with additional countermeasures. In this chapter, we present how to build a network of multi-actor dependencies (i.e., social organisation) where risks are minimised by delegating goals to capable and trustworthy actor.

We use planning techniques to elicit possible design alternatives as presented in [41, 42] The process works as follows:

- actors, goals, capabilities, their knowledge (e.g., possible goal decompositions), possible dependency relationships among actors are identified;
- planner searches for a possible plan that satisfies all actors' goals;
- the plan is evaluated against risk. It assesses whether the level of risk associated to goals is acceptable;
- if the risk is unacceptable then changes are needed and a new plan is generated and again evaluated.

However, in safety critical systems it is important to know who is responsible for any decision taken. Therefore, it requires that the designer is part of the decisional process and, particularly, responsible of the approval of the final solution (i.e., the plan). Our framework here is meant to provide a Computer-Aided Software Engineering (CASE) tool that helps a designer in defining and evaluating each design alternative with respect to the associate risk level. Moreover, this approach can also be used to assist the designer in performing the runtime design of a Multi-Agent System (MAS) [229]. In this chapter, we use the case study of Air Traffic Management (ATM) to demonstrate our approach. First, we will briefly explain about ATM case study (Section 8.1). We then explain how to represent the problem of defining organisational-setting in terms of a description of planning domain (Section 8.2), and then how a plan is evaluated (Section 8.3). We demonstrate the application of our approach (Section 8.4), and then is followed by an overview of related work (Section 8.5) and some remarks (Section 8.6).

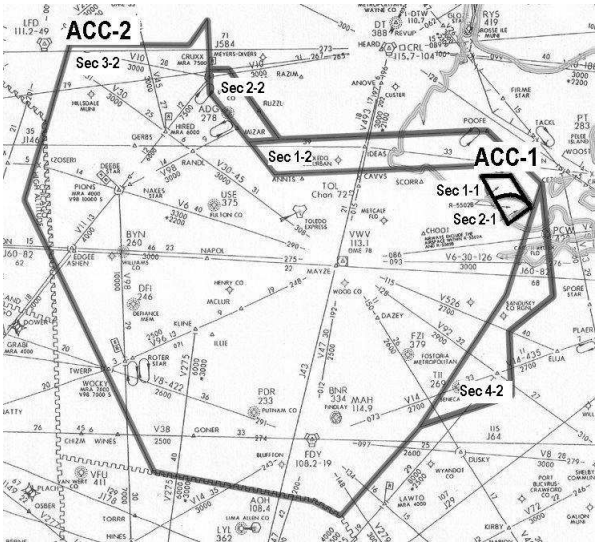


Figure 8.1: Airspace Division between ACC-1 and ACC-2

8.1 Partial Airspace Delegation in ATM

Air Traffic Management (ATM) case study has been used in the SERENITY Project[1] to validate security and dependability patterns. An ATM is categorised as a safety-critical system because it is closely related to the safety of human lives. Therefore, an ATM system is required to be available and reliable all the time of its operation. However, having a 100% available and reliable system is hardly possible, because many events can obstruct the system which cannot be known in advance (i.e., during the system development phase). For instance, aircraft traffic can increase beyond the safety threshold which was not anticipated during the design of the ATM. These events can compromise the availability and reliability of sub-components of the ATM system (e.g., controllers, radar processors, CWPs[2]).

An Air traffic Control Center (ACC) is a body authorised to provide air traffic control (ATC) services in certain airspace. These services comprise controlling aircraft, managing airspace, managing flight data of controlled aircraft, and providing information about the air traffic situation. Suppose there are two adjacent ACCs, namely ACC-1 and ACC-2 (Figure 8.1), where the airspace of ACC-1 is surrounded by the airspace of ACC-2. The airspace is organised into several adjacent volumes, called sectors. For instance, the airspace of ACC-1 is divided into two sectors (Sec 1-1 and Sec 2-1), and ACC-2 has its airspace divided into 4 sectors (Sec 1-2, 2-2, 3-2, and 4-2). Each sector is operated by a team, consisting of a controller (e.g., Sec 1-1 has C1-1 as a controller), and a planner (e.g., P1-1 is a planner for Sec 1-1). For the sake of communication and coordination, several adjacent sectors in an ACC are supervised by a supervisor (e.g., SU1-1 supervises Sec 1-1 and 2-1, and SU1-2 supervises Sec 1-2 and 2-2). The supervisor also acts as a designer and so, responsibly, approve/decline the new plans defining some re-sectorisation changes. The scenario starts from the normal operation of ATM in which SU1-1 delegates the control of sector 1-1 to team 1 formed by the executive controller C1-1 and the planning controller P1-1. C1-1 and P1-1 work together providing ATC services to the aircraft in sector 1-1. C1-1 controls aircraft to guarantee the safe vertical and horizontal separation of each aircraft, while P1-1 manages the

[1] http://www.serenity-project.org

[2] Controller Working Position (CWP) is a set of resources allocated to support a controller to perform his/her tasks

flight data of the controlled aircraft and the airspace of sector 1-1.

One day during summer holidays, a flight bulletin reports that there will be an increase of the en-route traffic in sector 1-1. According to the analysis made by P1-1, this goes beyond the capacity of a single controller (C1-1). Thus, SU1-1 needs to re-design his/her sectors in a way so that the en-route traffic can be handled safely. Essentially, there are two possibilities to resolve such a situation:

- divide the airspace into smaller sectors s.t. each controller covers a smaller area and consequently, the overall airspace capacity is increased along the increase of the sector number supervised by SU1-1; or
- delegate a part of the airspace to the adjacent supervisor (it could be from the same or different ACC).

Each alternative introduces different requirements. For instance, when dividing the airspace, SU1-1 needs to ensure the availability of a controlling team (G_{14} , G_{21} in Table 8.1) and the availability of a set of CWP (G_{15} , G_{22} in Table 8.1). Conversely, if SU1-1 decides to delegate a part of his airspace to another supervisor, then SU1-1 needs to define delegation schema (G_{10} in Table 8.1) and to have sufficient level of "trust" towards the target supervisor and his team to manage the delegated airspace. Moreover, SU1-1 needs to be sure that the target supervisor has sufficient infrastructure (e.g., radar, radio communication coverage) to provide ATC services in the delegated airspace.

The details of the ATM case study are presented in Section 8.4, including organisational-setting and capabilities of each actor. In the following sections, we explain how to encode the problem as a planning problem, and then how to evaluate and refine the candidate plan in order to maintain the level of risk below a predefined threshold.

8.2 Planning Domain

Generating a design alternative can be framed as a planning problems in which the planner is used to construct a plan that satisfies the stakeholders' goals. The main idea is to automatically determine the course of actions (i.e., a plan) needed to achieve a certain goal, where an action is a transition rule from one state of the system to another [161, 221]. Actions are described in terms of preconditions and effects. If a precondition is true in the current state of the system, then the action can be performed. As a consequence of an action, the system will be in a new state where the effect of the action is true.

Thus, to define the planning problem, we need to formalise

- the initial and the desired states of the system;
- the actions that are possible in the system;
- the planning domain axioms.

In order to represent the initial state of the system (i.e. actor and goal properties, social relations among actors), first order logic is used with conjunctions of predicates and their negations. To describe our domain, we use the following predicates proposed previously in [41, 42]:

- For the goal properties:
  - satisfied(G – goal), which becomes true when the goal G is fulfilled. The predicate is used to define the goal of the planning problem (i.e., to specify, which goals should be satisfied in the final state of the system);

  - and/or_subgoal$_n$(G, G$_1$, G$_2$, ..., G$_n$ – goal) represents the predefined way of goal refinement, namely, it states that G can be and/or-decomposed into n and/or-subgoals;
  - type(G – goal, GT – goal_type) is used to typify goals;
  - criticality_h/m/l(G – goal) represents the criticality of the goal, one of high, medium, or low. The criticality level implies the minimum level of trust between the actors required for a goal to be delegated. For instance, if the criticality of the goal G is high, then the goal could be delegated from the actor A$_1$ to the actor A$_2$ only if the trust level between A$_1$ towards A$_2$, about the goal type in which G belongs to, is at least high.

- For the actor properties:
  - wants(A – actor, G – goal) represents the initial actor's desires;
  - can_satisfy(A – actor, G – goal) and can_satisfy_gt(A – actor, GT – goal_type) are used to represent the capabilities of an actor to satisfy a goal, or a specific type of goal, respectively.

- For the social relations:
  - can_depend_on_gt_h/m/l(A$_1$, A$_2$ – actor, GT – goal_type) indicates the trust level in which actor A$_1$ can delegate the fulfilment of the goal of type GT to actor A$_2$. The trust level of the dependency is defined as (h)igh, (m)edium, or (l)ow, respectively.

A plan can contain the following actions:

- *Goal satisfaction*. An actor satisfies a goal if it is among its desires (either initially, or after the delegation from another actor), and it has the capability to satisfy it;

- *Goal decomposition*. A goal could be decomposed either into the and-subgoals, meaning that all of subgoals should be satisfied to satisfy the initial goal, or into the or-subgoals, which represent alternative ways of achieving the initial goal;

- *Goal delegation*. An actor might not have enough capabilities to achieve its goals by itself and therefore, it has to delegate the responsibility of their satisfaction to other actors. As was mentioned before, the delegation can only take place if the level of trust between the actors is not lower than the criticality level required for the goal to be delegated;

- *Goal relaxation*. An actor might need to delegate a goal to another actor where the trust level between them is insufficient. To be able to perform delegation, an actor needs to relax the goal criticality (i.e., lowered) [210] so that it satisfies the trust level. However, this action is considered as a risky action, because it infers the actor delegate the fulfilment of the goal to the other actor which is untrusted. Therefore, to minimise the risk we strongly averse such an action unless there is no other way to achieve the goal.

To complete the planning domain, we add some axioms which mimic the behaviours of a GR model. For example, to propagate goal properties through goal refinement we introduce such axioms: a goal is satisfied if all its and-subgoals or at least one of the OR-subgoals is satisfied.

We have chosen LPG-td [139], a fully automated system for solving planning problems, for implementing our planning domain. LPG-td supports the PDDL (Planning Domain Definition Language) 2.2 specification, which was used to formalise system states, actions and axioms described above. The details on how and why this planner has been chosen and how the actions and axioms of the planning domain are implemented in PDDL 2.2. have been presented in [42] and [41].

Require: domain {domain description in PDDL}
problem {goal and initial state of the problem in PDDL}
whitelist {a list of allowed action}
relax{allow/not relaxation}
1: boolean finish←false
2: **while not** finish **do**
3: plan ←$run\_planner(domain, problem, relax)$
4: **if not** $evaluate\_sat(plan)$ **then**
5: $refine\_sat(plan, problem)$
6: **else if** relax **and not** $evaluate\_act(plan)$ **then**
7: $refine\_act(plan, problem, whitelist)$
8: **else**
9: finish ←true
10: **return** plan

Algorithm 8.1: Planning and Evaluation Process

8.3 Evaluation Process

After a design alternative, called a candidate plan, is generated by the planner, it should be evaluated based on a number of criteria. If it satisfies the criteria then a designer reviews and approve it, otherwise the problem definition is refined by identifying the actions that should be avoided to get the less risky design alternative. The refinement of the problem definition is followed by replanning.

We here evaluate the candidate plan using a risk evaluation metric [6]. The objective of the iterative planning-and-evaluation procedure is to select a plan among the available alternatives that has an acceptable level of risk. In this framework, we consider two types of risk: the risk about the satisfaction of a goal (called sat-risk) and the risk related to goal delegation. sat-risk represents the evidence value of a goal being denied/failed when an actor attempts to fulfil it. The evidence value is represented in terms of the following predicates: *FD* (Fully Denied), *PD* (Partially Denied), and *ND* (Not Denied). These predicates are associated with the high, medium, and low level of sat-risk, respectively. The risk of goal delegation occurs when the goal delegation occurs after the criticality of the goal has been relaxed.

The process of planning and evaluating a suitable design alternative is illustrated in Algorithm 8.1. However, we recommend to execute the algorithm without allowing any relaxation (i.e., $relax = false$). If it could not elicit any design alternative, then the designers can re-execute with allowing the relaxation action. Note that some steps in the algorithm are fully automated (e.g., *run_planner* line 3), while some still need a human involvement (e.g., adding the allowed actions to the *whitelist* in line 7). The algorithm is iterative and comprises the following phases: planning, evaluation, and, finally, plan refinement if necessary. There are two evaluation steps in the algorithm (in case relaxation is allowed): STEP-1 evaluates the risks of goal satisfactions (line 4), and STEP-2 evaluates relaxation actions (line 6). Each evaluation step is followed by a refinement action (line 5 or 7), which aims at changing the planner input so that during the next iteration it will produce the better (i.e. less risky) candidate plan.

STEP 1: Goal Satisfaction Evaluation

After a candidate plan is elicited (line 3), it should be evaluated if it meets the requirements imposed on it (i.e., the level of risk associated with the plan is below the predefined threshold). The aim of this

evaluation step (line 4) is to assure that sat-risk values of the candidate plan (i.e. the likelihood of each system goal being denied/failed) are at most equal to the accepted ones specified by designers.

By examining the candidate plan, a GR model can be constructed (depicted in Figure 8.3). The GR model shows how a top goal is refined into tangible leaf goals (i.e. a leaf goal where there is an actor that can fulfil it). Using the sat-risk values of leaf goals, the risk level of the top goal are calculated using the forward reasoning (presented in Chapter 5). Notice different candidate plans can result in different sets of inputs goals and vary on defining which actors will satisfy the leaf goals, and consequently they result in different sat-risk of top goal.

If sat-risk of one top goal is higher than the specified threshold, then the refinement process is performed. The refinement (line 5) identifies those assignments of the leaf goals to actors that should have been avoided in order to have the sat-risk values of the top goals within the specified thresholds. The refinement process starts by generating a possible set of assignment (i.e., sat-risk values of the leaf goals) that can result in the sat-risk of top goals being within the specified thresholds. This set of assignments is called a *reference model* and obtain by means of backward reasoning (explained in Chapter 5). The reasoning requires the following inputs: the constructed GR model, the specified thresholds as desired values, and leaf-goals as input goals.

By comparing the sat-risk value of leaf goals in the GR model with the corresponding values in the reference model, the designers can identify which actions of goal satisfaction should be avoided. In other words, we should not delegate a goal G to the actor A if the actor A can satisfy the goal G with sat-risk which is greater with the one prescribed by the reference model. For instance, in Figure 8.3 the risk-sat of goal capable managing airspace *(G_{19})* that is satisfied by actor P1-1 is FD (Fully Denied), while according to the reference model, the value of G_{19} should be at most PD (partially denied). Therefore, the problem definition needs to be refined such that P1-1 does not satisfy G_{19} .

However, we cannot refine the problem by simply specifying G_{19} must not be satisfied by P1-1, because there might be a situation where the provision of G_{19} by P1-1 is sufficient for some other top goals. Ideally, we specify "the path of actions" from the top goal that lead to the goal G_{19} being satisfied by P1-1. To simplify the refinement process, we only consider one action involving G_{19} , performed just before P1-1 satisfies it. In case of Figure 8.2(b), such an action (called *previous related action*) is $and\_decompose$ G_3 into G_{18} and G_{19} , which is performed by P1-1. Thus, the refinement predicate that should be introduced in the problem definition is the following.

$$\neg(satisfy(P_{1-1}, G_{19}) \wedge and\_decompose_2(P_{1-1}, G_3, G_{18}, G_{19}))$$

After the problem definition is refined, the planner is run again to elicit a new candidate plan using the refined problem definition. All the above described steps can be done automatically, without any interference a designer.

STEP 2: Action Evaluation

The second evaluation step (line 6) is performed to evaluate all relaxation actions in a candidate plan whether they are acceptable/not risky based on the designer judgements. In this framework, we consider two types relaxation: direct and further relaxation. The former refers to the situation where it is done by the owner of the goal[3] and is considered as non-risky action. The letter is *further relaxation* referring to the situation where the relaxation is done by another actor (i.e., not the owner). *Further relaxation* is

[3]Goal G is owned by actor A if G was initially wanted by actor A. (i.e., in the initial state A wants G to be satisfied) and not because of some delegations. Moreover, all the subgoals of G are also considered to be owned by Actor A

considered a risky action, though sometimes it is impossible to find a candidate plan without it. Thus, this action should be explicitly evaluated and allowed by the designer. To approve *further relation*, the designer adds the predicate of such actions into the *whitelist*.

For instance, in the ATM case study SU1-1 intends to increase his airspace capacity in response to the traffic increase by delegating his airspace *(G_{11})* to SU1-2. Since the fulfilment of G_{11} is critical (the criticality level is high), SU1-1 needs to have high trust level towards SU1-2 for delegating G_{11} (i.e., $can\_depend\_on\_gt\_h(SU_{1-1}, SU_{1-2}, G_{11})$ should be true). Later, SU1-2 refines G_{11} into the subgoals control the aircraft *(G_2)* and manage the airspace *(G_3)*. For satisfying these goals, SU1-2 needs to depend on the controller C1-2 for G_2 , and on the planner P1-2 for G_3 , because they are the ones that have the capabilities to satisfy the corresponding goals. Let us assume the trust level of the dependency of SU1-2 towards C1-2 for G_2 is medium while G_2 was considered as critical by SU1-1. To delegate G_2 , SU1-2 needs to *further relax* the criticality of G_2 .

Basically, the evaluation aims to guarantee that there is no-relaxation action taken by an actor which is not the owner of the goal. Otherwise, the designer needs to explicitly allow such actions (i.e., add it to the *whitelist*). If a further relaxation is considered to be dangerous, then the designer needs to refine (line 7) the problem definition forbidding such an action and followed by the replanning.

8.4 Experimental Results

In this section, we illustrate the application of our approach to the ATM case study. The following subsections detail the case study formalisation, and the planning-and-evaluation process, performed in accordance with Algorithm 8.1. The aim of the process is to elicit an appropriate plan for SU1-1's sector, taking into account the constraints and the risk of each alternative. The scenario starts with the intention of SU1-1 to increase the capacity of airspace *(G_6)* as a response to the air traffic increase in sector 1-1. SU1-1 faces a problem where C1-1 is not available to control *(G_{14})* more traffic. Therefore, SU1-1 needs to modify sector 1-1 without involving C1-1 s.t. the increase of air traffic can be handled.

8.4.1 Case Study Formalisation

The following inputs should be provided for Algorithm 8.1:

- A formalised problem definition, which contains all the properties of the actors of the ATM system, and their goals. The complete list of properties can be found in Table 8.2.
- Goals of the planning problem (e.g., satisfy G_6 without involving C1-1 in satisfying G_{14}).
- A list of authorised further relaxation actions ($whitelist$).
- Risk levels of goal satisfaction actions. Table 8.1 shows all sat-risk values of the satisfaction actions.
- Accepted risk values (e.g., risk value of G_6 is at most PD).

In Table 8.1, the goal criticality values are presented in column Crit. Moreover, goal criticality (i.e., high, medium, or low) denotes a minimum level of trust between two actors required to delegate a goal. For instance, goal manage airspace *(G_3)* is categorised as a *highly* critical goal, and goal analyze air traffic *(G_8)* has *low* criticality. Thus, these goals require different level of trust to be delegated.

Goal \ Actor			C1-1	C2-1	P1-1	P2-1	SU1-1	C1-2	P1-2	SU1-2
Id.	Description	Crit.								
G1	Manage Aircraft within ACC									
G2	Control Aircraft	H								
G3	Manage Airspace	H								
G4	Manage Flight Data	M						PD		
G5	Maintain Air Traffic Flow in Peak-Time									
G6	Increase Airspace Capacity									
G7	Analyze Air Traffic	L								
G8	Re-sectorize within ACC									
G9	Delegate Part of Sector									
G10	Define Schema Delegation	M			ND					PD
G11	Delegate Airspace	H								
G12	Have Controlling Resources									
G13	Have Capability to Control the Aircraft		ND	PD				PD		
G14	Avail to Control		FD	ND						
G15	Have Control Working Position for Controller	H					ND			PD
G16	Have Authorization for FD Modification	M	ND	ND						
G17	Have Capability to Manage FD				ND	PD				
G18	Have Resources for Planning	M					ND			
G19	Have Capability to Manage Airspace				FD	PD			PD	
G20	Have Capability to Analyze Air Traffic				PD					
G21	Avail to Plan				ND	ND			ND	
G22	Have Control Working Position for Planner	H					ND			ND

Table 8.1: Goals Criticality and Satisfaction Risk (Criticality = H: High, M: Medium, L: Low and sat-risk = Full Denied, Partial Denied, and Not Denied)

Moreover, Table 8.1 presents the sat-risk of a goal when an actor tries to achieve it. Note that, the sat-risk level of a goal depends on which actor satisfies the goal. For instance, the table states G_{19} can be satisfied either by actor P1-1, P2-1, or P1-2, and each actor has different level of risk (sat-risk) – *full*, *partial*, and *partial*, respectively. The empty cells in Table 8.1 imply the actor does not have capabilities to fulfil the corresponding goal.

Table 8.2 presents properties of actors and their goals in ATM case study. Namely, it represents actor capabilities (can_satisfy), possible ways of goal refinements (decompose), and possible dependencies among actors (can_depend_on) together with the level of trust for each dependency. For instance, actor SU1-1 can_satisfy goals G_{15}, G_{18}, and G_{22} , and the actor has knowledge to decompose G_1, G_5, G_6, G_8, and G_9 . And SU1-1 has *high* level of trust to delegate G_2 to C1-1 or C2-2. The same intuition is applied for the other cells.

8.4.2 Planning and Evaluation Process

STEP 0: Planning. After specifying the inputs, the planner is executed to elicit a candidate plan to fulfil the predefined goals, which is shown in Figure 8.2(a). These goals state that the plan should satisfy G_6 , and the solution should not involve C1-1 to satisfy G_{14} because C1-1 is already overloaded controlling the current traffic. Moreover, the planner should not involve the other ACC (i.e., SU1-2) by avoiding the delegation of G_{11} to SU1-2 even it is possible in Table 8.2. Before adopting the candidate plan (Figure 8.2(b)), two evaluation steps explained in previous section should be performed to ensure the risk of the candidate plan is acceptable.

STEP 1: Goal Satisfaction Evaluation assesses the satisfaction risk of a candidate plan. The GR model of goal G_6 (in Figure 8.3) is constructed on the basis of the candidate plan (in Figure 8.2(b)). The GR

Actor	can_satisfy	decompose			can_depend_on		
		type	top-goal	sub-goals	level	dependum	dependee
SU1-1	G15	And	G1	G2, G3	H	G2	C1-1, C2-1
	G18	And	G5	G6, G7	H	G3	P1-1, P2-1
	G22	Or	G6	G8, G9	H	G4	P1-1, P2-1
		And	G8	G2, G3, G4	M	G7	P1-1
		And	G9	G10, G11	M	G10	P1-1
					L	G10	SU1-2
					H	G11	SU1-2
P1-1, P2-1	G17	And	G3	G18, G19	H	G22	SU1-1
	G19	And	G4	G16, G17, G18			
	G21	And	G7	G18, G20			
P1-1	G10				L	G16	C1-1
	G20						
P1-2					L	G16	C2-1
C1-1, C2-1	G13	And	G2	G4, G12, G13	H	G15	SU1-1
	G14	And	G12	G14, G15			
	G16						
C1-1					M	G4	P1-1
C2-1					M	G4	P2-1
					M	G4	P1-1
SU1-2	G10	And	G11	G2, G3	M	G2	C1-2
	G15				M	G3	P1-2
	G22						
P1-2	G19	And	G3	G19, G21, G22	M	G22	SU1-2
	G21						
C1-2	G4	And	G2	G4, G13, G15	M	G15	SU1-2
	G13						

Table 8.2: List of Actors and Goal Properties for the ATM Case Study. (Level of trust: H: High, M: Medium, L: Low)

model shows which actors are responsible to satisfy the leaf goals. For instance, G_{19} must be satisfied by P1-1 and in this scenario, G_9 is left unsatisfied because the other or-subgoal, G_8 , was selected to satisfy G_6 .

In this scenario, we assume that the acceptable sat-risk value for G_6 is PD. To calculate the sat-risk value of goal G_6 , forward reasoning is performed (i.e., the sat-risk values of leaf goals in Table 8.1 are propagated up to the top goal). This reasoning mechanism is a part of the functionality of the SI*-Tool.[4] By means of the forward reasoning, we obtain that the sat-risk of G_6 is FD, which is higher than the acceptable risk (i.e., PD). Thus, the refinement is needed to adjust the problem definition, so that a less risky plan is constructed during the next replanning. The refinement starts with the elicitation of a reference model using backward reasoning. The reference model specifies that all leaf goals must have at most PD sat-risk value in order to have the sat-risk of top goal G_6 at most PD.

By comparing the sat-risks of leaf goals in the GR model with the reference model, G_{19} (satisfied by P1-1) is detected to be a risky goal; its sat-risk (in Table 8.1) is FD which is higher than the one in the reference model. Therefore, the problem definition is refined to avoid P1-1 satisfying G_{19} . As G_{19} is a subgoal of G_3 , the decomposition action is also negated, as the previous related action, according to the procedure explained in the previous section. Thus, the problem definition is refined, and the goal

[4]Available at `sesa.dit.unitn.it/sistar_tool`

```
(satisfied G6)
(not(satisfy C1-1 G14))
(not(delegate SU1-1 SU1-2 G11))
```

(a) Goal of Problem Definition

```
0: (OR_DECOMPOSES2 SU1-1 G6 G8 G9)
1: (AND_DECOMPOSES3 SU1-1 G8 G2 G3 G4)
2: (DELEGATES SU1-1 C2-1 G2)
3: (AND_DECOMPOSES3 C2-1 G2 G4 G12 G13)
4: (SATISFIES C2-1 G13)
5: (AND_DECOMPOSES2 C2-1 G12 G14 G15)
6: (SATISFIES C2-1 G14)
7: (DELEGATES C2-1 SU1-1 G15)
8: (SATISFIES SU1-1 G15)
9: (DELEGATES C2-1 P2-1 G4)
10: (DELEGATES SU1-1 P2-1 G4)
11: (AND_DECOMPOSES3 P2-1 G4 G16 G17 G18)
12: (SATISFIES P2-1 G17)
13: (DELEGATES P2-1 SU1-1 G18)
14: (SATISFIES SU1-1 G18)
15: (RELAX2L P2-1 G16)
16: (DELEGATES P2-1 C2-1 G16)
17: (SATISFIES C2-1 G16)
18: (DELEGATES SU1-1 P1-1 G3)
19: (AND_DECOMPOSES2 P1-1 G3 G18 G19)
20: (SATISFIES P1-1 G19)
```

(b) The Candidate Plan after STEP 0

Figure 8.2: Plan for Increasing Air Space Capacity

of the planning problem is now of the form shown in Figure 8.4(a). Afterwards, the planner is run to elicit a new candidate plan. Basically, the new candidate plan is almost the same with the previous plan (Figure 8.2(b)), the only difference is in lines 18-20 (see Figure 8.4(b)). If the candidate plan is generated by allowing some relaxation, then the candidate plan is evaluated by going through the next step to ensure all the actions (especially, further relaxations).

STEP 2: Action Evaluation filters the malicious relaxation actions. The scenario starts from the goal G_6 which is wanted by SU1-1. As all the other goals of the candidate plan are the result of the refinement of G_6 , the owner of all of them is again SU1-1. Thus, relaxing the criticality of any goals that is performed by any actors except SU1-1 is seen as a risky action.

For instance, P2-1 relaxes the criticality of G_{16} (line 15 in Figure 8.2(b)) to low instead of medium. By default, this action is a risky one and should be avoided, unless the designer states explicitly that this action is not risky by adding it to the *whitelist*. Once it is considered unacceptable, the planning problem should be refined by adding the negation of the relaxation action (i.e., (not (relax2l P2-1 G1))).

Moreover, the designer can also introduce rules to avoid certain actions. For instance, the designer may prevent C2-1 from delegating G_4 to P2-1 (line 9 in Figure 8.2(b)) by adding a new predicate to the goal of the planning problem (namely, (not (delegate C2-1 P2-1 G4))). For the sake of simplicity, all the possible relaxation actions in the candidate plan are put to the *whitelist*, so we do not refine the problem definition any further. Thus, the last candidate plan to redesign SU1-1's sector is approved s.t. the traffic increase can be handled. Moreover, the plan suffers acceptable risks according to the

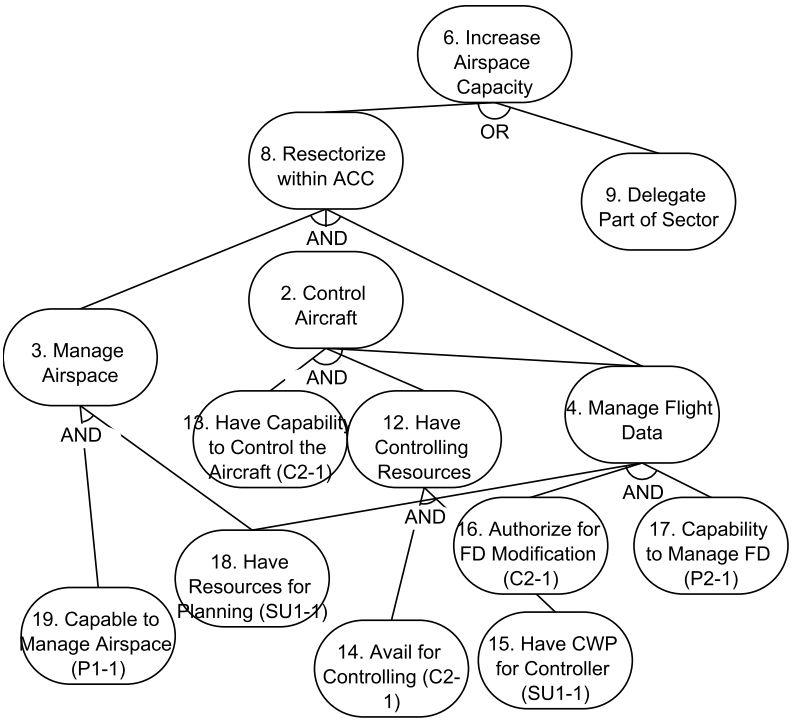


Figure 8.3: The GR Model of Candidate Plan in Figure 8.2(b)

```
(satisfied G6)
(not (satisfy C1-1 G14))
(not (delegate SU1-1 SU1-2 G11))
(not (and (satisfy P1-1 G19)(and_decompose2 P1-1 G3 G18 G19)))
```

(a) Problem Definition Refinement after STEP 1

```
.............
18: (DELEGATES SU1-1 P2-1 G3)
19: (AND_DECOMPOSES2 P2-1 G3 G18 G19)
20: (SATISFIES P2-1 G19)
```

(b) Final Plan for Satisfying G_6

Figure 8.4: Final Problem Definition and Plan for increase the airspace capacity *(G_6)*

predefined thresholds (i.e., sat-risk of G_6 is less or equal than PD).

8.5 Related Work

AI planning has been intensively developed during the last decades, and has found a number of interesting applications (e.g., robotics, process planning, autonomous agents, etc.). There are two basic approaches to the solution of planning problems [221]. First approach is graph-based planning algorithms in which a compact structure, called Planning Graph, is constructed and analysed. In the other approach the plan-

ning problem is transformed into a SAT problem and a SAT solver is used. There exist several ways to represent the elements of a classical planning problem (i.e. the initial state of the world, the system goal, or the desired state of the world, and the possible actions system actors can perform). The most widely used, and to a certain extent standard representation is PDDL (Planning Domain Definition Language), the problem specification language proposed in [79]. Current PDDL version, PDDL 2.2 [70] used during the last International Planning Competition [102], supports many useful features (e.g., derived predicates and timed initial literals).

A few works can be found which relate planning techniques with information system design. In [1] a program called ASAP (Automated Specifier And Planner) is described, which automates a part of the domain-specific software specification process. ASAP assists the designer in selecting methods for achieving user goals, discovering plans that result in undesirable outcomes, and finding methods for preventing such outcomes. The disadvantage of the approach is that the designer still performs a lot of work manually while determining the combination of goals and prohibited situations appropriate for the given application, defining possible start-up conditions and providing many other domain-specific expert knowledge. Some works present a planning application to assist an expert in designing control programs in the field of Automated Manufacturing [48]. The system they have built integrates POCL (Partial Order Causal Link), hierarchical and conditional planning techniques [48, 161]. The authors consider standard planning approaches to be not appropriate with any ready-to-use tools for the real world, while in our work the opposite point of view is advocated, and the off-the-shelf planner is used.

8.6 Concluding Remark

In this chapter, we have proposed an approach to incorporate risk analysis into the process of MASs design. The approach is based on the use of planning techniques to explore the space of alternative designs and risk-based evaluation metrics to evaluate the resulting solutions. We argue that the approach is particularly suitable for the design of critical and responsive systems, such as air traffic management, health-care systems, disaster management, traffic management, etc.

The proposed framework is meant to support a designer in generating, exploring, and evaluating design alternatives either during the initial, “classical” design, or during runtime redesign of a MAS. We consider runtime redesign of high importance for modern information systems, which operates in continuously changing environment and then require highly adaptable characteristics. Among the limitations of our approach, we would like to mention that it only supports a centralised viewpoint (i.e., the designers viewpoint), while the different actors in a system may have different priorities and criticalities (i.e., perceive risks).

Chapter 9

BDI Agent's Deliberation Process: a Risk-driven approach

The Goal-Risk (GR) framework has been developed using some notion related to agent-oriented systems. However, most of its use cases aim at assisting analysts during the system development phase by providing a modelling framework, methodology, and a CASE tool to develop the models and analyse them. We here aim at enabling a software agent to perform risk analysis using the GR framework. Consequently, the software agent can anticipate and react towards risks during runtime as part of its autonomous behaviour.

Agent technology is becoming more and more an emergent alternative to build safety-critical systems [126, 130]. In some high risk situations or safety-critical missions, humans are replaced by autonomous agents (e.g., reconnaissance, battle-combat). However, developing autonomous agents for such tasks require the introduction and consideration of uncertainty concepts (especially risks) and related problems within the agent's deliberation process.

There have been several proposal in modelling agent's mental states proposed in the literature [143, 165, 184]. Most of them describe agent's mental states in terms of Belief, Desire, and Intention (the BDI model). The BDI model has been initially introduced by Bratman [36] and then refined by Rao and Georgeff in [164, 165] for real implementation in agent based systems, such as Procedural Reasoning System (PRS) and distributed Multi Agent Reasoning System (dMARS). Currently, agent tools and frameworks (e.g., Jack [96], Jadex [163], and Jason [32]) use effectively the BDI model in their implementations. The deliberation process of an agent is supported by a meta-level reasoning, where the most appropriate plan for achieving a given goal is selected on the basis of specific criteria such as priority and belief. Unfortunately, these implementations do not consider uncertain events and, in particular, risks (i.e., uncertain events with negative impacts [109]) as an integral part of the meta-level reasoning. However, several approaches have been proposed to reason about uncertainty [88, 223], but often their complexity made almost impossible the implementation in real agent-based applications [44, 119].

We here adopt and adapt the GR Framework into a deliberation process of a BDI agent. Essentially, the GR framework consists of three basic concepts: goals capturing the desires of an agent, risks as uncertainty that can affect the satisfaction of goals, and treatments as special plans that are intended to mitigate the risks. These concepts are depicted in terms at a GR model (explained in Chapter 4) which is used to reason about uncertain events that might obstruct the achievement goals and plans and treatments that are necessary to mitigate the risks. Moreover, the GR framework provides an agent with the capability to choose a strategy (i.e., a combination of plans and treatments) to pursue a goal, whose

risk and cost are acceptable by the agent. We use the Unmanned Aerial Vehicle (UAV) case study to motivate our work as illustrated in Section 9.1. In Section 9.2, we briefly explain the implementation framework of Jadex [163] and in Section 9.3 how the basic concepts of GR framework is encoded into the Jadex platform is detailed. Finally, we mention some related work in Section 9.4 and draw some conclusions in Section 9.5.

9.1 Unmanned Aerial Vehicle

An Unmanned Aerial Vehicle (UAV) is an aircraft without pilot that either is controlled remotely or flies autonomously. UAVs are typically used in a number of critical missions, such as decoy, reconnaissance, combat, and even for research and civil purposes. In the early time, UAVs were called drones because they were not more than aircraft remotely controlled by human. Recently, several efforts have been made to apply intelligent agents to control aircraft [68, 120, 219]. Attempts to use intelligent agents in managing UAVs are addressed to avoid the risk of human life loss. In this setting, agents can respond to event occurrences autonomously without waiting for instructions from the ground control. This capability results to be essential in critical and dangerous missions (e.g., reconnaissance, decoy, and combat) where the response time has to be as short as possible. For instance, if a drone realises that it has been detected by the enemy, it informs the ground control about the danger. However, it will still proceed with the mission according to the previous plan until it receives new instructions from the ground control. It is possible that these new instructions are sent to the UAV too late for ensuring the success of the mission. The ambitious objective of the agent paradigm is to provide UAV with facilities to react autonomously and so to take countermeasures in the appropriate time.

There are still numerous open problems for completely replacing human with agents. For instance, a human pilot can learn from past experience such that it can adopt adequate measures when a new occurrence of events is detected. This work aims at improving software agents.

9.2 Jadex as an Agent Platform

There have been several agent infrastructures based on BDI concepts being developed in the last years, such as Jack [96], Jadex [163], and Jason [32]. Typically, agent platforms follow the Procedural Reasoning System (PRS) computational model [78] in implementing their reasoning engine. PRS-like systems are event-based architectures where events[1] are used to denote incoming messages, a new goal to be processed, or a change in the state of an existing goal. Once an event is generated, the agent dispatches the event to the deliberation mechanism (called meta-level reasoning) to choose the most appropriate plan to handle it. In some PRS-systems, like Jack and Jason, goals are represented as special type of event (i.e., goal-event). Thus, agents implemented in Jack or Jason platform are not aware which goals that they are currently pursuing. They execute the plan only as a response to the occurrence of an event.

On the contrary, the notion of goal plays a key role in the Jadex platform. This has spurred us to choose Jadex as platform to implement the GR framework. Jadex requires specifying agents' goals explicitly. These goals must be related to the plans that provide means to achieve them. Jadex represents agents' beliefs in terms of Java objects, and stores them in the beliefbase. Moreover, Jadex allows us to specify the plan that has to be executed when a belief is changed. Those BDI descriptions are represented in an XML file, called *Agent Definition File* (ADF). The Jadex reasoning engine starts the deliberation

[1]The notion of "event" supported by agent platforms is different from the one in the GR framework.

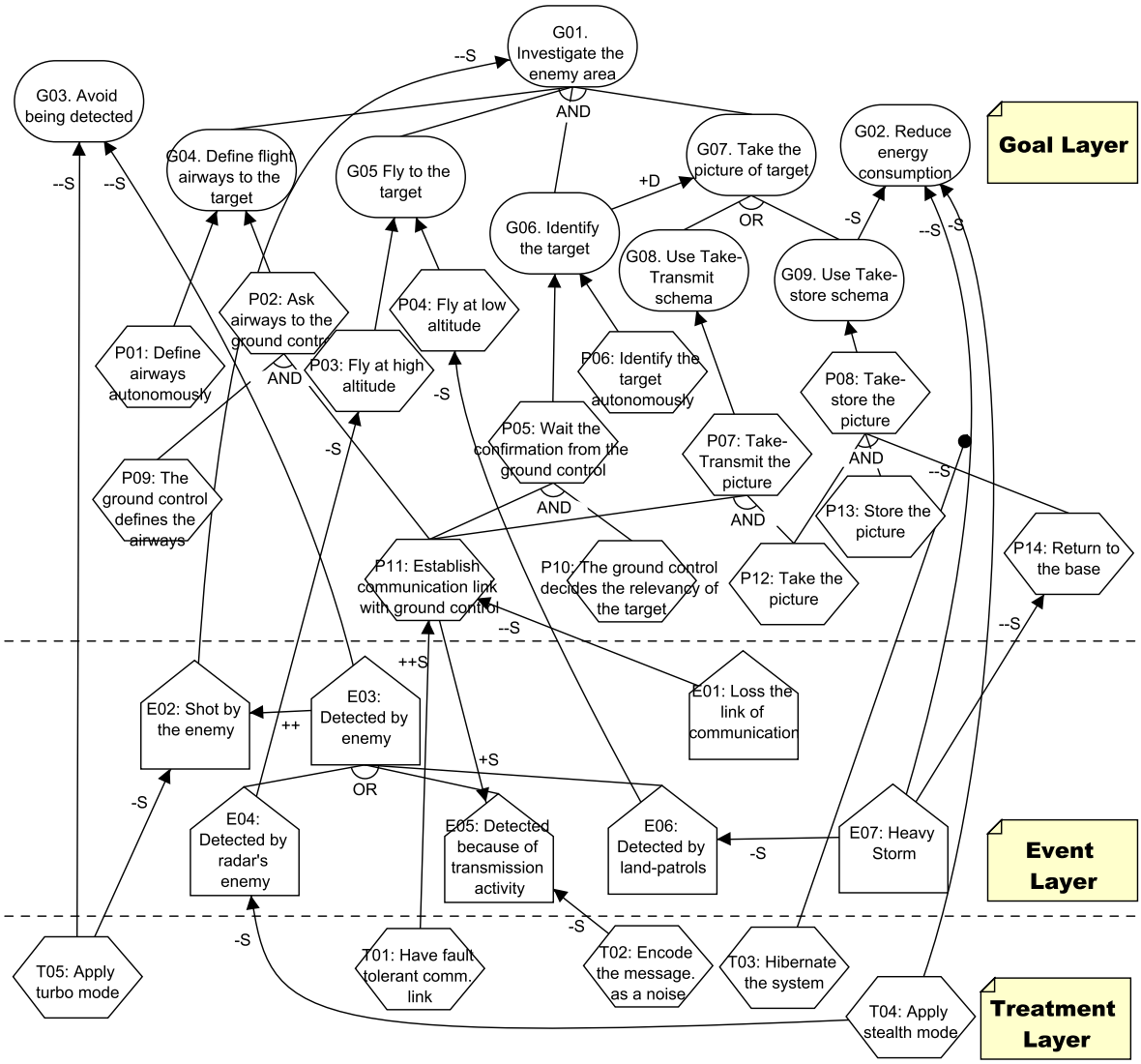


Figure 9.1: Goal-Risk model of the UAV Agent

process by considering the goal requested by an agent. The goal is then stored in the database (i.e., goalbase) that contains all adopted goals by the agent. To distinguish between just adopted and actively pursued goals, Jadex labels a goal into three states: option, active, and suspended. When a goal is adopted, it becomes an option goal in the agent's goalbase. Once the agent pursues the goal, it becomes active and a goal-event is triggered. The event is then dispatched to the meta-level reasoner in order to find the most appropriate plan to achieve the goal.

9.3 Framework Realisation

Though, the GR framework is developed adopting BDI concepts to model agent's mental states, some concepts may be not trivially mapped into Jadex. For instance, in the GR framework a goal can be

decomposes into several sub-goals, and the decomposition is ended when all leaf-goal have (at least) one plan to achieve it. Conversely, in Jadex such an operation is not possible. A goal needs always to be associated with some plan, and however a plan can introduction a new goal. Moreover, an agent must perform meta-level reasoning to choose a plan to achieve the goal.

Suppose by following the methodology (see Chapter 6) analysts obtain a GR model as depicted in Figure 9.1. The process to implement the GR models into the Jadex platform starts from defining the agent's beliefs (Figure 9.2(a)). The agent's beliefs include: the thresholds of risk level and cost (line 2-7) that are acceptable for the UAV agent; the current cost and risk level of adopted strategies are encoded in (line 8-10) and (line 11-13) respectively. The success rate for plans, the likelihood of events, and the success rate for treatments (i.e., depicted as GR evidence values/labels) are also included as the beliefs (line 17-32). We also provide the agent with knowledge (e.g., which events can obstruct one goal) represented in the GR model by implementing the model as a Java object and storing it as a belief (line 14-16). We also specify assess_risk plan (line 2-9 in Figure 9.2(b)) in case there is a change in these belief values. Essentially, assess_risk is the implementation of *Forward Reasoning* (presented in Algorithm 5.1 at Chapter 5) which calculates the risk level and cost of adopted strategy (i.e., plans and treatments).

To contain the risk level and cost, the UAV agent needs to introduce two additional goals: maintain_risks (line 2-9 in Figure 9.2(c)) and maintain_costs respectively. If such goals are denied, plan replanning (line 10-15 in Figure 9.2(b)) is executed to recover the desired conditions defined by those goals. Essentially, replanning is a JAVA implementation of *Find Treatments* (presented in Algorithm 7.2 at Chapter 7) which aims to synthesise a strategy on the basis of the current beliefs. We here assume that there is always a strategy for each replanning attempt.

We also need to represent goals, plans, and treatments of the GR model in the ADF, so that the agent behaves according to the GR model at runtime. Goals are declared in the ADF goalbase, while plans and treatments are depicted as plans section of the ADF. This transformation starts by declaring all goals, plans, and treatments occurring in the GR model as XML entries in the ADF as depicted in Figure 9.2(c) and Figure 9.2(b). Additional Jadex-goals and Jadex-plans need to be introduced to mimic the behaviours supported by the GR framework which are not supported by the Jadex platform.

For instance, goals, plans and treatments in a GR model can be AND/OR-decomposed into subgoals, while in Jadex a goal can only be AND-decomposed. The GR decompositions depict the knowledge of an agent about the relations between the upper level goal with subgoals. On the other hand, AND-decomposition in Jadex is done as a result of the execution of an applicable plan. For instance, G_1 is AND decomposed into G_6 and G_7 (left-side in Figure 9.3). To realise this behaviour in a Jadex agent, we need to design an additional plan P-G01 where it introduces (and later dispatches) subgoals G_6 and G_7, as depicted in the right-side of Figure 9.3. P-G01 is considered as a means to achieve G_1 (depicted by arrow line) which is represented in the ADF description as in Figure 9.2(b) (line 16-21). Both Tropos and Jadex AND decomposition have the same semantic meaning where the fulfilment of all AND-subgoals is required for fulfilling the upper-level goal. Moreover, AND-decompositions of plans/treatments are done with a similar intuition where some additional goals are introduced and associated each plan as a means to achieve each additional goal. For instance, plan P_8 will introduce additional goals (i.e., G-P12, G-P13, and G-P14) that later activate subplans P_{12}, P_{13}, and P_{14}. To mimic goal OR-decompositions, we introduce additional plans where each plan is used to activate a new subgoal. These plans represent alternative ways that an agent can adopt to achieve the upper-level goal. For instance, G_7 is OR-decomposed into G_8 and G_9 (Figure 9.3). This is represented in the ADF by introducing the additional plans P-G08 and P-G09 that respectively activate G_8 and G_8, as alternatives to achieve goal G_7. The agent will perform a meta-level reasoning on the meta-goal of G_7 (called

```
<beliefs>
<belief name="thres_cost"  class="double">
<fact>2000000</fact>
</belief>
<belief name="thres_risk"  class="char">
<fact>P</fact>
</belief>
<belief name="cost"  class="double">
<fact>1000000</fact>
</belief>
<belief name="risk"  class="char">
<fact>N</fact>
</belief>
<belief name="gr_model"  class="GRmodel">
<fact>new GRmodel("uav.grmodel")</fact>
</belief>
<beliefset name="goals" class="GRlabel">
<fact>new GRlabel(G1,N)</fact>
...
</beliefset>
<beliefset name="plans" class="GRlabel">
<fact>new GRlabel(P1,F)</fact>
...
</beliefset>
<beliefset name="events" class="GRlabel">
<fact>new GRlabel(E1,P)</fact>
...
</beliefset>
<beliefset name="treatments" class="GRlabel">
<fact>new GRlabel(T1,N)</fact>
...
</beliefset>
...
</beliefs>
```

(a) Beliefs

```
<plans>
<plan name="assess_risk">
<body>new RiskAssessment()</body>
<trigger>
<beliefsetchange ref="events"/>
<beliefsetchange ref="plans"/>
<beliefsetchange ref="treatments"/>
</trigger>
</plan>
<plan name="replanning">
<body>new Replanning()</body>
<trigger>
<goal ref="maintain_risk"/>
</trigger>
</plan>
<plan name="P-G01">
<body>..</body>
<trigger>
<goal ref="G01"/>
</trigger>
</plan>
<plan name="chooseG07">
<body>new RiskReasoning()</body>
<trigger>
<goal ref="choose_planG07"/>
</trigger>
</plan>
<plan name="P7">
<body>..</body>
<trigger>
<goal ref="G8"/>
</trigger>
</plan>
...
</plans>
```

(b) Plans

```
<goals>
<maintaingoal name="maintain_risk">
<maintaincondition>
\$beliefbase.risk < \$beliefbase.risk.thres_risk
</maintaincondition>
<targetcondition>
\$beliefbase.risk < \$beliefbase.risk.thres_risk
</targetcondition>
</maintaingoal>
<achievegoal name="G01">
...
</achievegoal>
...
<achievegoal name="G07">
...
</achievegoal>
<metagoal name="choose_planG07">
...
<trigger>
<goal ref="G07"/>
</trigger>
</metagoal>
<achievegoal name="G08">
...
</achievegoal>
...
</goals>
```

(c) Goals

Figure 9.2: UAV agent description in Jadex-ADF

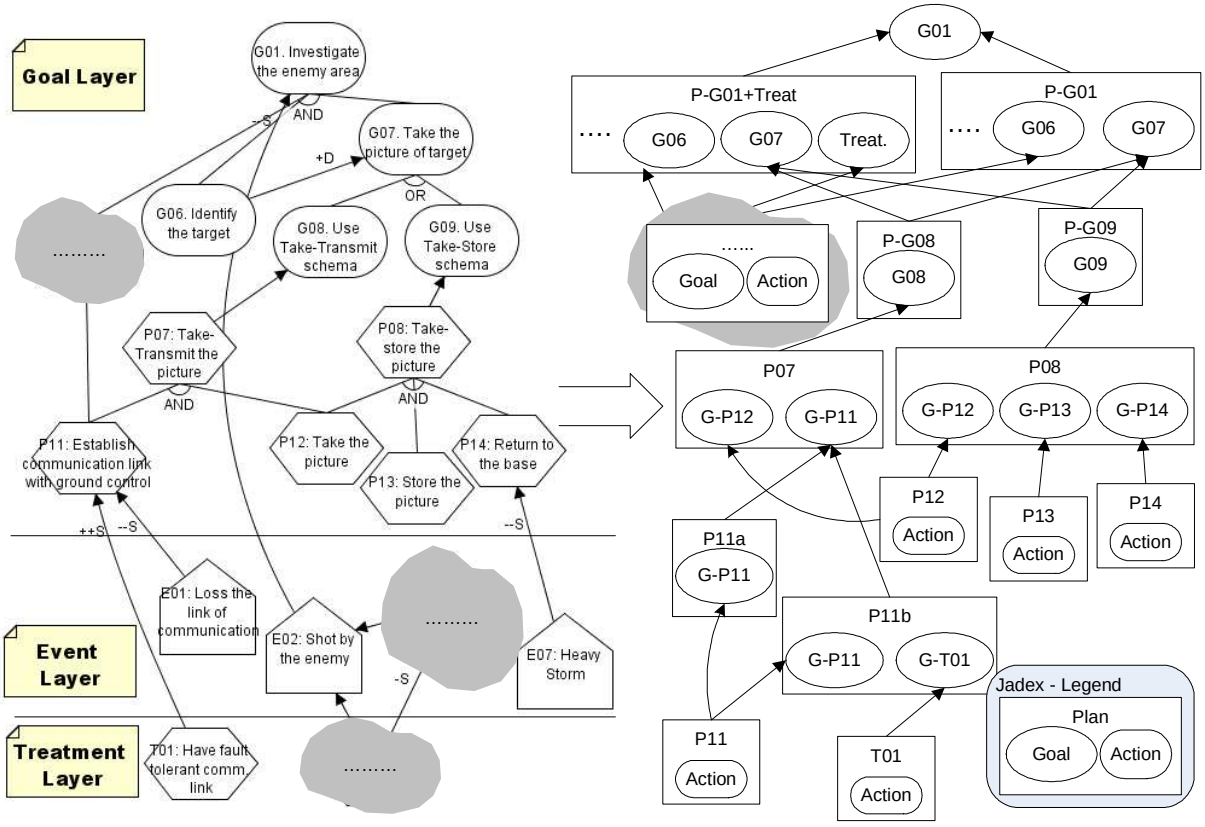


Figure 9.3: Goal-Risk model to Jadex

choose_planG07 in Figure 9.2(c) line 17-22) to decide which plan should be adopted. We override the Jadex meta-level reasoning by defining a plan (chooseG07 in Figure 9.2(c)) to always adopt the least risky alternative by means of *Forward Reasoning*. OR-decompositions of plans/treatments can be mimicked by defining a task that introduces an additional goal, and all subplans are defined as means to achieve the additional goal. Similarly to what we have done for goal OR-decomposition, the Jadex meta-level reasoning needs to be overridden with a plan similar with chooseG07.

Finally, we need to represent the GR means-end relations in a Jadex agent. Means-end relations correspond to the trigger mechanism in Jadex. For instance, $P_7 \longmapsto G_8$ is represented by adding G_8 as the trigger of P_7 in the ADF (line 30-32 in Figure 9.2(b)) or denote by relating P07 to G08 with an arrow line (right-side in Figure 9.3). This implementation of the GR means-end into the Jadex platform holds if the plan is not related with any treatments (i.e., direct or indirectly through the event layer). On the contrary, if the plan is related with treatments, the mapping schema is slightly different because several combinations of plans and treatments can be used to achieve the goal (e.g., it can adopt only the plan, the plan with a treatment, the plan with some treatments, or the plan and all treatments). For instance, the success of plan establish communication with ground control *(P_{11})* is obstructed by the risk due to loss the link of communication *(E_1)* (Figure 9.3). In this setting, the UAV agent can adopt treatment have fault tolerant communication link *(T_1)* for a retention measure. The agent may adopt only P_{11} (or P11a in Figure 9.3 (right-side)), whereas in others it needs to adopt P11b, which is the combination of P11 and T1, for ensuring the success of P_{11} execution. This implementation is getting more complicated when there are several available treatments. It is because the number of

combinations plans and treatments becomes exponential ($N(plan) \times 2^{N(treatment)}$). For instance, if G_1 is obstructed by E_2 directly and by E_3 indirectly and the agent knows there are one plan to achieve G_1 and three treatments (e.g., T_2, T_4, and T_5) to mitigate E_2 and E_3 Consequently, the agent has eight possible strategies to achieve goal G_1.

9.4 Related Work

In artificial intelligent community, a lot of effort has been devoted to develop reasoning mechanisms dealing with uncertainty [88, 127, 223]. Here, the focus was on determining how subtle uncertainties affect the reasoning mechanisms. Differently, the autonomous agent community is mainly focusing on how to implement agent platforms (e.g., Jack, Jason and Jadex) with reasoning mechanisms to deal with existing event and situation [37, 119, 219].

Halpern, in [88] proposed a framework to reasons about uncertainty from a single agent viewpoint and multi-agent environments. An environment (i.e., entities outside the agent) is treated as black boxes, with no structure. Therefore, the likelihood of event is viewed as a collection of random variables. In the multi-agent framework, the environment can be structured on the basis of the interactions among agents. The framework also supports modelling uncertainty as a function over time. Markov Decision Processes (MDPs) [223] is a mathematical framework to model a decision-making process where the outcomes are partly random and partly controlled by agents. MDPs address optimisation problems that are represented as a tuple $\langle S, A, P, R\rangle$, where S is a set of possible states, A is a set of actions that caused state transitions, P is a set of probabilities of action occurrences, and R is a set of rewards that are gained if an action is executed. This framework uses value iteration algorithms to determine the set of actions that minimise and maximise the reward. If risks are encoded as rewards, the agent can use such algorithms to identify the least risky strategy by searching the solution with minimal reward. Moreover, Simari and Parsons [185] investigated the relation between MDPs and BDI architectures which enable us to use MDPs as reasoning mechanisms in the BDI architecture. Another proposal is the Abstract Agent Programming Language (3APL)[2], a cognitive agent programming language that employs logical representation and reasoning features based on the BDI model. In this approach, the beliefs are assumed to be certain, either true or false. Kwisthout and Dastani [127] propose an extension of 3APL that allows an agent to reason about uncertain beliefs. This extension essentially uses the Dempster-Shafer theory [182] to model uncertainty in agent's beliefs. These reasoning frameworks are sufficient for reasoning about uncertainty, though sometime they appear too complex, but do not specify how an agent must react to deal with the effects of uncertainty. In other words, they are sophisticated to model uncertainties in the reasoning mechanism of an agent, but they lacks in representing some important aspect, such as: 1) the extent of uncertainty events (i.e., severity) affecting agents and 2) what the agents can do to anticipate the events (i.e., not react). These aspects are important because reasoning mechanisms should distinguish events by their effect in case they occur, besides only their likelihood/uncertainty and some events can be anticipated to occur (i.e., prevention measures).

In the autonomous agent community, there are some proposals [119, 219] that present the implementations of an UAV agent (called *Wingman*) in the Jack platform. The Wingman is provided with basic plans (e.g., flying quietly and at low altitude) in order to achieve its goals (e.g., taking picture of enemy installations). The Wingman is also provided with additional plans (e.g., flying as fast as possible) to deal with malicious events such as being detected by enemy. When a malicious event occurs, the Wingman reasons about its plans and diverts the adopted basic plan with another plan to guarantee the achievement

[2] http://www.cs.uu.nl/3apl/

its goals. However, the Wingman does not try to anticipate the occurrence of malicious events, but rather to react when they happen. Similar works have been done in [68, 215]. On the contrary, our approach allows the UAV agent to anticipate malicious events by employing treatments when the risk is unacceptable. Another implementation of reasoning in an autonomous agent has been presented in [37]. The authors propose an implementation of the *Cleaner World* agent in Jadex. Such an agent is designed to pick all wastes and throw them in trash bins. In this implementation, the agent moves to a location once its sensors detect the existence of wastes. However, it would be more efficient if agent movements are not only driven by the appearance of wastes, but they are anticipated by reasoning about the likelihood of having new wastes in a certain location. In other words, those agents do not predict future events and react accordingly; most of them just react in case a particular event happens.

Conversely, this work is in the middle between those two approaches. Our work adopts existing reasoning mechanisms in agent platforms (i.e., Jadex), and extends them with the Goal-Risk reasoning mechanism. This approach takes advantages of having the reasoning mechanism implementable in agent platforms and, especially, in the ones based on BDI. Though the proposed reasoning looks trivial compared with the ones proposed in AI community, we remark that the GR framework is not meant to assess the uncertainty precisely. Rather, it aims to support an agent in defining a strategy whose risk and cost are lower than a certain threshold. As consequence, it also allows agents to define a strategy to achieve their goals on the basis of probable events, and not only the current context condition.

9.5 Concluding Remark

In this work, we have presented a Jadex implementation of the Tropos Goal-Risk framework. The GR framework and Jadex differ in concepts and reasoning features and their integration was not straightforward. In particular, we have illustrated some limitations in Jadex to adopt all concepts supported by the GR framework, but we have also demonstrated that this mapping does not limit the expressiveness of the intended goal deliberation process.

The initial GR reasoning mechanisms allow agents to perform cost-benefit analysis on a strategy to be adopted, but require an exhaustive knowledge before the agent starts pursuing its goals. On the other hand, the Jadex approach results faster in choosing a strategy for a given goal and more adaptive to the current conditions of the agent. We have taken advances of combining GR and Jadex reasoning features by implementing complete plans and facilities for reasoning about them in the belief of the agent. The intuition is to use complete plans as a baseline for the Jadex reasoning. Then, during the pursuing of its goals, the agent evaluates the baseline by reasoning about risks in the current situation and modify the baseline if it is considered to be risky.

We plan to extend the risk reasoning to cope with multi-agent environments. In this setting, an agent should be able to reason about risks when it depends on other agents for the fulfilment of its goals. Yet, treatments may be introduced when risks are unacceptable or when the agent has no alternatives to achieve its goal besides depending on other agents.

Chapter 10

Conclusions

10.1 Summary of the Thesis

The last years we have seen a growing interest in considering risk during the development of information systems, particularly critical ones (CISs). However, the major limitation of current approaches is that they analyse a system as merely a technical artefact, and often overlook the organisational setting in which the system operates. In fact, many incidents are originated from some aspects beyond just technical failures, such as abuse of permissions, the distrust of other group members. Therefore, considering interactions between humans and technology in an organisation allows us to identify a wide range of risks in addition to those emerging from both aspects in isolation.

This thesis proposes an integrated framework, called the *Tropos Goal-Risk framework*, that supports analysts to assess risk along the analysis of stakeholders' needs/requirements. We proposed a modelling language tailored to capture requirements and risks in socio-technical systems. The modelling language is composed of three conceptual layers: first, the *value* layer, where stakeholders' needs are analysed and operationalised into a set of system requirements; second, the *event* layer, where events are captured, analysed, and quantified according to their impact to the value layer; and third, the *treatment* layer where additional countermeasures are introduced to mitigate risks (i.e., events with negative impacts).

This thesis also presents the methodological steps to assess risks alongside requirements analysis. The process starts from analysing stakeholders' needs/requirements, and refines them into finer structures, so that each need is satisfied either by the stakeholder itself, through the deployment of technical systems, or by other actors in the system. The process continues with analysing risks, which prevent the requirements from being satisfied. By using *forward analysis*, analysts can assess the risk level that the organisation is exposed to. If the risk is acceptable, then the process is finished; otherwise analysts need to introduce countermeasures to mitigate the risk (i.e., by reducing the likelihood of the risk or by alleviating the severity of the risk). Note that analysts must also analyse the side-effects of countermeasures to the initial requirements. The *backward analysis* was developed to assist analysts in evaluating all possible combinations of countermeasures so that the risk is acceptable. All the modelling and analysis steps are supported with a tool, namely SI* Tool, that has been realised as an Eclipse plug-in[1].

The Tropos Goal-Risk framework has been used in several national and European projects (e.g., TOCAI, SERENITY, and MASTER) where it is applied to a number of industrial case studies. Moreover, these applications have also demonstrated that the primitives of a GR model are adequate to capture most concepts related to risk and requirement in socio-technical systems. However, the proposed framework

[1] Available at `http://sesa.dit.unitn.it/sistar_tool/`

has some open challenges and reveals a number of pitfalls that need to be addressed in the future works. For instance, the GR framework has not addressed the issues related to requirement changes (i.e., driven by the modification of stakeholders' needs or by some changes in risk factors). In addition, there is no clear-cut steps explaining how to realise the requirements into the further phases of software development (i.e., design, implementation, deployment, maintenance), so that the integrity of the requirements is preserved.

We here have considered risk analysis as part of the requirement analysis, but this does not mean that we do not need to perform risk analysis in later phases of system development also (e.g., design, implementation, and deployment). This is because we might discover new risks or refine identified risks into more precise forms only when we have detailed designs of the system.

10.2 Pros and Cons of the Goal-Risk Framework

Starting with pros, we had positive experiences in communicating GR models to analysts and domain experts. This is an important strength for any requirements analysis technique because it empowers domain experts to understand and critique proposed models [3]. Moreover, the learning process for experts to understand and use a GR model is relatively short (approximately 2-3 months). This is partly due to the familiarity of experts with the concepts of goal and task. In [11], we report on the usage of the GR framework as part of the SI* modelling framework in identifying security and dependability patterns, and in [15] the GR framework was used to evaluate the possible business solutions in manufacturing Small-Medium Enterprises. This point is really significant for the success of a requirement analysis, because it allows the domain experts to understand and criticise directly the requirement model which might differ from their knowledge.

All modelling languages come with several choices of features and constructs. Essentially, *i** and SI* are considered as the underlying modelling frameworks for the GR framework. Modelling organisations was a key motivation for the original *i**, while SI* aims at modelling security and privacy concerns in organisations. However, the GR framework only include some constructs that are essential for analysing risk and requirement in a socio-technical system (e.g., role, agent, goal, task, resource, dependency, trust), and discard others (e.g., "position", "occupy", "monitoring", "delegation of permission"). We acknowledge that sometimes we stretched the English meaning of the relations but adding more constructs would have decreased the readability of the models and hindered their communication to other stakeholders outside the analysis team.

The GR framework supports risk analysis since the very early phases of software development. Consequently, it reduces the risk of requirements changes, and therefore also the cost of development. Moreover, by capturing the business objects in the "value layer" (e.g., goals, tasks, resources) one can focus on any events that can prevent the attainment of these business objects (e.g., the anti-goals of competitors, the disruptions of business processes, or the unavailability of some resources). However, most existing risk frameworks only focus on risks that obstruct some physical assets required to achieve stakeholders' goals. However, an event may not be a risk for resources of an actor, but it still might obstruct the corresponding goal. For instance, the goal *increase sales* can be achieved by the means *advertise products* and *discount products*. Let's assume that both means are undisrupted. However, the event of *new competition* is still considered as a risk to the goal, though it does not disrupt any of the means to achieve the goal. Moreover, the GR framework enables us to assess risks in a multi-actor setting where the impact of risk can affect other actors following the dependency among them [15].

Many risk modelling frameworks only consider risks (i.e., events with negative impact) of the system-

to-be. However, events may also introduce positive impact. Within the GR framework, analysts can capture such situations, and consequently perform trade-off analysis to find mitigation that results in the acceptable risk level without pre-empting any opportunities. Moreover, the three layers of the GR model can work together with other frameworks, such as: FTA, FMECA, and the Attack-Graph, and the Markov models. For instance, analysts may decide to replace the event modelling with FMECA to analyse risks in the event layer of a GR model. However, some adjustments are required, in particular on the axiom of impact relations since the even likelihood in the GR framework is categorised into 3 qualitative levels while FMECA has 5 levels. In addition, instead of using the task modelling, one can use BPMN [156] to capture the process level of the value layer in a GR model,

Besides those pros, there are several weaknesses in the proposed framework. In the ATM models [3], and we believe in many others, the GR modelling framework misses some important constructs. For instance, the notion of *collective or group responsibility* could not be captured a convincing representation with *i** and its descendants. In such a setting, each member of a group must do its part, and all together they must ensure the fulfilment of the other members' parts. As an example, in a sector team in ATM, which is composed of an executive controller and a planning controller, each controller has its own obligations, but they need to support one to another so that the group can fulfil the task. For the moment, we capture such a situation by modelling the group as an agent (i.e., a big agent) and each member is depicted as agents that are part of the big agent. However, such representation ("part-of") is still in debate because it might have some overlap with "is-a" relation that has been introduced. Therefore, some mereology [213] studies are required to clarify their meaning and distinction. Moreover, the GR modelling framework (and perhaps other *i** descendants) suffers from a serious problem in terms of scalability. One might find some GR models are hardly readable when they capture a big and complex system/organisational-setting (see the Figure 7.3 and Figure 7.2 in Chapter 7).

Another weak point of the GR is that the notion of evidence (SAT and DEN) is not as well understood as *probability*, which is most often used for risk analysis. In response to this, we adjust the framework to be able to accept both inputs (e.g., evidence and probability) [15]. Moreover, one may argue that the GR framework's assessment lacks of precision in assessing the risk level. We remind to the readers that this framework was not proposed to assess risk precisely, but rather to provide guidance to analysts in eliciting requirements so that the requirements can fulfil the stakeholders' needs and expose to the acceptable risk. Such guidance is useful, particularly, for the software project with limited budget, our framework can point to risky requirements so that they get more priority. Of course, this analysis depends on value judgements made by the analysts in determining levels of likelihood and severity. To reduce subjectivity, these judgements are typically determined collectively by a group of analysts. For a more formal approach, one may adopt techniques such as the Delphi method [136] where an iterative process results in stable judgements for all participants.

10.3 Future Works

In some situations, we experienced where the GR modelling framework could not depict such situations in particular at an outsourcing scenario or a virtual organisation [89] in general. The dependency and the value layer of a GR model is considered to be less expressive comparing how an enterprise is captured in an enterprise engineering field [40, 125, 195]. In a GR model, the relation of how processes or artefacts deliver the value to the actors/organisations is captured by a means-end relation, which is considered too simplistic in an outsourcing scenario. The work in enterprise modelling (e.g., e3value [208], PRIDE-EEM [40]) captures better how an organisation structures its business and how IT systems supports the

business. They specify how some business services are composed, and in the case of outsourcing both organisations (i.e., provider and user) need to agree on some policies that cover quality of service, service licensing, compensation measures, etc. However, we need to keep in mind that adding more constructs might decrease the readability of the model and hinder its communication to the stakeholders. Another significant issue is that the GR framework lacks support in modelling the notion of time that is essential to model events and system behaviours.

Moreover, we are currently investigating some possibilities to deal with the model scalability issue. One possibility is organising a GR model into several layers of detail similar with the idea of level in the Data-Flow Diagram [225]. Another possibility is making a GR model of an organisation is similar to a map of a city. In this way, one can zoom-in to see the details of the model, or zoom-out to see the overview of the model and hide most of the details. Currently, we are still studying the trade-off of each possibility especially in the aspects of the use-cases of the model and the supporting tool.

Often requirements analysis is perceived as time and resource consuming efforts with less value by software projects. This view is understandable due to the fact that often the implementation might differ from what the analysts have been analysed; or often a requirement model is only useful up to the step where we are defining the system design, and afterwards the requirement model can be trashed out. One way to deal with such opinions is to increase the usages of a requirement model during system development. To realise such vision, it is necessary to have a mechanism that makes the transformation from requirements to the design and ultimately to the implementation of the system seamlessly. Moreover, the mechanism is useful to guarantee the integrity of the requirements and also to make sure the requirements can be traced from the implementation or vice versa. In fact, we are currently studying the possible usages of a GR model (and also a SI* model) during runtime. For instance, a GR model can be used to identify WHICH indicators to monitor (i.e., requirements as desired states-of-affairs), and if we can trace the implementation of such requirement, then we can define WHERE to monitor (i.e., the implementations of the requirements). Monitoring during a system operation is essential to guarantee the quality assurance of the system. We believe such works can also be beneficial to specify the management change or the deployment strategy.

Another direction under investigation involves the study of perceived risk in a multi-trust domain and even an outsourcing scenario. However, we believe that this area requires multidisciplinary studies (e.g., psychology, behavioural economic, sociology) and some empirical experiments to obtain reasonable results. In [9], we have identified some basic concepts that are relevant to assess perceived risk.

In this thesis, we have studied how "dependency" (i.e., delegation of execution) and "trust" affect the risk level that is perceived by an actor. We here assume that by depending on another actor, an actor becomes vulnerable to the performance of the other actor. For instance, Alice (director) depends on Bob (field manager) to *give the assessment of market condition*. In fact, we might assume Alice is at risk when Bob cannot fulfil his task. Such representation allows analysts to assess the availability and reliability of business services in an organisation. However, the GR framework still lacks of support in analysing integrity and confidentiality risks (i.e., other aspects of information security). For instance, Alice gives the permission of *signing the purchase order* to Charlie. In permission concern, the risk is not originated from the fact that Charlie could not sign the draft, but rather to the fact that how risky such permission is used for other purposes. In fact, giving Charlie permission might put Alice at risk, but often it is necessary otherwise the purchase order could not be processed while Alice is away. To deal with such risks, we need in-depth study of the notion of "entitlement" and "permission".

Bibliography

[1] John S. Anderson and Stephen Fickas. A proposed perspective shift: Viewing specification design as a planning problem. In *Proceedings of 5th International Workshop on Software Specification and Design (IWSSD '89)*, pages 177–184, 1989. ISBN 0-89791-305-1.

[2] Annie I. Anton. Goal-Based Requirements Analysis. In *Proceedings of the 2nd IEEE International Conference on Requirements Engineering (ICRE'96)*, page 136, Washington, DC, USA, 1996. IEEE Computer Society Press. ISBN 0-8186-7252-8.

[3] Y. Asnar, R. Bonato, V. Bryl, L. Campagna, K. Dolinar, P. Giorgini, S. Holtmanns, T. Klobucar, P. Lanzi, J. Latanicki, F. Massacci, V. Meduri, J. Porekar, C. Riccucci, A. Saidane, M. Seguran, A. Yautsiukhin, and N. Zannone. Security and Privacy Requirements at Organizational Level. Project Deliverable A1.D2.1, SERENITY consortium, November 2006. EU-IST-IP 6th Framework Programme - SERENITY 27587.

[4] Y. Asnar, M. V. Bryl, Fabiano Dalpiaz, P. El-Khoury, M. Felici, H. Halas, B. Krausova, C. Riccucci, A. Saidane, M. Sguran A. Yautsiukhin and, and K. Li. Extenderd Set of Security and Privacy Patterns at Organizational Level. Project Deliverable A1.D3.1.2, SERENITY consortium, 2008. EU-IST-IP 6th Framework Programme - SERENITY 27587.

[5] Yudistira Asnar and Paolo Giorgini. Modelling Risk and Identifying Countermeasures in Organizations. In *Proceedings of the 1st International Workshop on Critical Information Infrastructures Security*, volume 4347 of *Lecture Notes in Computer Science*, pages 55–66. Springer-Verlag, 2006.

[6] Yudistira Asnar and Paolo Giorgini. Risk Analysis as part of the Requirements Engineering Process. Technical Report DIT-07-014, DIT - University of Trento, March 2007.

[7] Yudistira Asnar and Paolo Giorgini. Analyzing Business Continuity through a Multi-Layers Modell. In *Proceedings of 6th International Conference on Business Process Management*, 2008.

[8] Yudistira Asnar and Paolo Giorgini. Analysing Risk-Countermeasure in Organizations: a Quantitative Approach. Technical Report DIT-07-047, DIT - University of Trento, July 2007.

[9] Yudistira Asnar and Nicola Zannone. Perceived Risk Assessment. In *Proceedings of the Fourth Workshop on Quality of Protection*. IEEE Computer Society Press, 2008.

[10] Yudistira Asnar, Volha Bryl, and Paolo Giorgini. Using Risk Analysis to Evaluate Design Alternatives. In *Proceedings of the Seventh International Workshop on Agent-Oriented Software Engineering*, volume 4405 of *Lecture Notes in Computer Science*. Springer-Verlag, 2006. Invited Paper.

[11] Yudistira Asnar, Roberto Bonato, Paolo Giorgini, Fabio Massacci, Valentino Meduri, Carlo Riccucci, and Ayda Saidane. Secure and Dependable Patterns in Organizations: An Empirical Approach. In *Proceedings of the 15th IEEE International Requirements Engineering Conference*. IEEE Computer Society Press, 2007.

[12] Yudistira Asnar, Paolo Giorgini, Fabio Massacci, and Nicola Zannone. From Trust to Dependability through Risk Analysis. In *Proceedings of the Second International Conference on Availability, Reliability and Security*. IEEE Press, 2007.

[13] Yudistira Asnar, Massimo Felici, Rizomiliotis Panagiotis, Alessandra Tedeschi, and Artiom Yautsiukhin. Extended Version of S&D Metrics. Project Deliverable A1.D5.2, SERENITY Consortium, 2008. EU-IST-IP 6th Framework Programme - SERENITY 27587.

[14] Yudistira Asnar, Paolo Giorgini, Paolo Ciancarini, Rocco Moretti, Maurizio Sebastianis, and Nicola Zannone. Evaluation of business solutions in manufacturing enterprises. *International Journal on Business Intelligence and Data Mining*, 3(3):305–329, 2008.

[15] Yudistira Asnar, Rocco Moretti, Maurizio Sebastianis, and Nicola Zannone. Risk as Dependability Metrics for the Evaluation of Business Solutions: A Model-driven Approach. In *Proceedings of the Third International Conference on Availability, Reliability and Security*, 2008.

[16] Australian Standard. Risk management. AS/NZS 4360, 1999.

[17] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl E. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on Dependable and Secure Computing*, 1(1):11–33, 2004.

[18] Lawrance S. Bale. Gregory bateson, cybernetics, and the social/behavioral sciences. *Cybernetics & Human Knowing. A journal of Second Order Cybernetics & Cyber-Semiotics*, 3(1):27–45, 1995.

[19] Basel Committee on Banking Supervision. Basel II: International Convergence of Capital Measurement and Capital Standards: a Revised Framework. http://www.bis.org/, June 2004.

[20] R.A. Bauer. Consumer Behavior as Risk Taking. In D. Cox, editor, *Risk Taking and Information Handling in Consumer Behavior*. Harvard University Press, 1967.

[21] Tim Bedford and Roger Cooke. *Probabilistic Risk Analysis: Foundations and Methods*. Cambridge University Press, 2001.

[22] I. Ben-Gal. Bayesian networks. In F. Ruggeri, R. Kenet, and F. Faltin, editors, *Encyclopedia of Statistics in Quality and Reliability*. John Wiley & Sons, 2007.

[23] Daniel Bernoulli. Exposition of a New Theory on the Measurement of Risk. *Econometrica*, 22: 23–36, 1954. ISSN 00129682. (original 1738).

[24] Peter L. Bernstein. *Against the Gods: The Remarkable Story of Risk*. Wiley, August 1998. ISBN 0471295639.

[25] James R. Bettman. Perceived risk and its components: A model and empirical test. *Journal of Marketing Research*, 10(2):184–190, 1973. ISSN 00222437.

[26] Moshiur Bhuiyan, M.M.Zahidul Islam, George Koliadis, Aneesh Krishna, and Aditya Ghose. Managing business process risk using rich organizational models. In *Computer Software and Applications Conference, 2007. COMPSAC 2007 - Vol. 2. 31st Annual International*, volume 2, pages 509–520, 2007. ISBN 0730-3157.

[27] Stefan Biffl, Aybüke Aurum, Barry Boehm, Hakan Erdogmus, and Paul Grünbacher, editors. *Value-Based Software Engineering*. Springer, 1 edition, October 2005. ISBN 3540259937.

[28] Barry Boehm. Value-based software engineering: reinventing. *ACM SIGSOFT Software Engineering Notes*, 28(2):3, 2003. ISSN 0163-5948.

[29] Barry W. Boehm. Software Risk Management: Principles and Practices. *IEEE Software*, 8(1): 32–41, 1991. ISSN 0740-7459.

[30] Barry W. Boehm and LiGuo Huang. Value-Based Software Engineering: A Case Study. *IEEE Computer*, 36(3):33–41, 2003.

[31] K. Boness, A. Finkelstein, and R. Harrison. A lightweight technique for assessing risks in requirements analysis. *Software, IET*, 2(1):46–57, 2008. ISSN 1751-8806.

[32] Rafael H. Bordini and Jomi Fred Hübner. BDI Agent Programming in AgentSpeak Using Jason. In *Proceedings of the 6th of International Workshop on Computational Logic in Multi-Agent Systems*, volume 3900 of *Lecture Notes in Computer Science*, pages 143–164, 2005.

[33] Borland. Mitigating risk with effective requirements engineering. White paper, Borland, April 2005.

[34] Robert P. Bostrom and J. Stephen Heinen. MIS Problems and Failures: A Socio-Technical Perspective. Part I: The Causes. *MIS Quarterly*, 1(3):17–32, September 1977.

[35] K. E. Boulding. General systems theory: The skeleton of science. *Management Science*, 2(3): 197–208, 1956.

[36] M. Bratman. *Intention, Plans, and Practical Reason*. Harvard University Press, 1987.

[37] Lars Braubach, Alexander Pokahr, Winfried Lamersdorf, and Daniel Moldt. Goal Representation for BDI Agent Systems. In Rafael H. Bordini, Mehdi Dastani, Jrgen Dix, and Amal El Fallah-Seghrouchni, editors, *Proceedings 2nd International Workshop on Programming Multiagent Systems: Languages and Tools*, volume 3346 of *Lecture Notes in Artificial Intelligence*, pages 9–20. Springer-Verlag, 7 2004.

[38] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004. ISSN 1387-2532.

[39] Paolo Giorgini Brian Henderson-Sellers, editor. *Agent-oriented Methodologies*. Idea Group Publishing, 2005. ISBN 1591405815, 9781591405818.

[40] M. Bryce and Associates. PRIDE - profitable information by design through phased planning and control. http://www.phmainstreet.com/mba/, 2008. access at 04.07.2008.

[41] Volha Bryl, Paolo Giorgini, and John Mylopoulos. Designing cooperative IS: Exploring and evaluating alternatives. In *Proceedings of the 14th International Conference on Cooperative Information Systems*, pages 533–550, 2006.

[42] Volha Bryl, Fabio Massacci, John Mylopoulos, and Nicola Zannone. Designing security requirements models through planning. In *Proceedings of the 18th Conference On Advanced Information Systems Engineering*, pages 33–47. Springer, 2006.

[43] BSI. Business Continuity Management. BSI 25999-1, 2006.

[44] Paolo Busetta and Kotagiri Ramamohanarao. An Architecture for Mobile BDI Agents. In *Proceedings of the 1998 ACM Symposium on Applied Computing*, pages 445–452. ACM Press, 1998. ISBN 0-89791-969-6.

[45] Shawn A. Butler. *Security Attribute Evaluation Method*. PhD thesis, School of Computer Science, Carnegie Mellon University, May 2003.

[46] Marvin J. Carr, Suresh L. Konda, Ira Monarch, F. Carol Ulrich, and Clay F. Walker. Taxonomy-Based Risk Identification. Technical Report CMU/SEI-93-TR-6, Software Engineering Institute, Carnegie Mellon University, June 1993.

[47] Cristiano Castelfranchi and Rino Falcone. Principles of Trust for MAS: Cognitive Anatomy, Social Importance and Quantification. In *Proceedings of 3rd International Conference on Multi-Agent Systems*, pages 72–79. IEEE Computer Society Press, 1998.

[48] L. Castillo, J. Fdez-Olivares, and A. Gonzalez. Integrating hierarchical and conditional planning techniques into a software design process for automated manufacturing. In *ICAPS 2003, Workshop on Planning under Uncertainty and Incomplete Information*, pages 28–39, 2003.

[49] Center for Chemical Process Safety (CCPS). *Guidelines for Engineering Design for Process Safety*. Wiley-AIChE, September 1993. ISBN 0816905657.

[50] EUROCONTROL Experimental Centre. Baseline integrated risk picture for air traffic management in europe. Technical Report EEC Note No. 15/05, EUROCONTROL, May 2005.

[51] Brian Chess and Jacob West. *Secure Programming with Static Analysis*. Addison-Wesley Professional, July 2007. ISBN 0321424778.

[52] L. K. Chung, B. A. Nixon, E. Yu, and J. Mylopoulos. *Non-Functional Requirements in Software Engineering*. Kluwer Academic Publishers, 2000.

[53] Robert T. Clemen and Robert L. Winkler. Combining probability distributions from experts in risk analysis. *Risk Analysis*, 19(2):187–203, April 1999.

[54] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, Shaker Heights, Ohio, United States, 1971. ACM.

[55] CORAS. CORAS: A Platform for Risk Analysis of Security Critical System. http://www.nr.no/coras/, 2005. accessed at September 2005.

[56] S.L. Cornford, M.S. Feather, and K.A. Hicks. DDP - A Tool for Life-Cycle Risk Management. In *Proceedings of the IEEE Aerospace Conference*, pages 441–451. IEEE Press, March 2001.

[57] Steven L. Cornford, Martin S. Feather, Vance A. Heron, and J. Steven Jenkins. Fusing Quantitative Requirements Analysis with Model-based Systems Engineering. In *Proceedings of the 14th IEEE International Requirements Engineering Conference*, pages 279–284, Los Alamitos, CA, USA, 2006. IEEE Computer Society Press.

[58] COSO. *Enterprise Risk Management - Integrated Framework*. Committee of Sponsoring Organizations of the Treadway Commission, September 2004.

[59] Susan Cramm. 8 things we hate about it. *BusinessWeek: Managing*, April 2008.

[60] R. Crook, D. Ince, L. Lin, and B. Nuseibeh. Security requirements engineering: When anti-requirements hit the fan. In *Proceedings of the 10th IEEE International Requirements Engineering Conference*, pages 203–205, 2002.

[61] Anne Dardenne, Axel van Lamsweerde, and Stephen Fickas. Goal-Directed Requirements Acquisition. *Science of Computer Programming*, 20(1-2):3–50, 1993.

[62] William Edwards Deming. *Out of the Crisis*. MIT Press, 2000.

[63] A.P. Dempster. The dempster-shafer calculus for statisticians. *International Journal of Approximate Reasoning*, 48(2):365–377, June 2008.

[64] F. den Braber, I. Hogganvik, M. Lund, K. Stølen, and F. Vraalsen. Model-based security analysis in seven steps - a guided tour to the coras method. *BT Technology Journal*, 25:101–117, 2007.

[65] Folker den Braber, Theo Dimitrakos, Bjørn Axel Gran, Mass Soldal Lund, Ketil Stølen, and Jan Øyvind Aagedal. The CORAS Methodology: Model-Based Risk Assessment using UML and UP. In *UML and the Unified Process*, pages 332–357. Idea Group Publishing, 2003.

[66] V. Dignum. *A model for organizational interaction: based on agents, founded in logic*. PhD thesis, Universiteit Utrecht, 2003.

[67] Didier Dubois and Henri Prade. Possibility Theory, Probability Theory and Multiple-Valued Logics: A Clarification. *Annals of Mathematics and Artificial Intelligence*, 32(1-4):35–66, 2001. ISSN 1012-2443.

[68] Jr. Dufrene, W.R. Approach for Autonomous Control of Unmanned Aerial Vehicle Using Intelligent Agents for Knowledge Creation. In *Proceedings of the 23rd of Digital Avionics Systems Conference*, volume 2, pages 1–9, 2004.

[69] Joanne Bechta Dugan and Tariq Said Assaf. Dynamic Fault Tree Analysis of a Reconfigurable Software System. In *Proceedings of the 19 th International System Safety Conference*, 2001.

[70] Stefan Edelkamp and Jorg Hoffmann. PDDL2.2: The language for the classical part of the 4th international planning competition. Technical Report 195, University of Freiburg, 2004.

[71] EEC - Eurocontrol Experimental Centre. Main Report for the 2005/2012 Integrated Risk Picture for Air Traffic Management in Europe. Technical Report EEC Note No. 05/06, EUROCONTROL, April 2006.

[72] Golnaz Elahi and Eric Yu. A Goal Oriented Approach for Modeling and Analyzing Security Trade-Offs. In *Proceedings of the 26th International Conference on Conceptual Modeling*, 2008.

[73] Rino Falcone and Cristiano Castelfranchi. Social Trust: A Cognitive Approach. In *Trust and Deception in Virtual Societies*, pages 55–90. Kluwer Academic Publishers, Norwell, MA, USA, 2001. ISBN 0-7923-6919-X.

[74] Martin S Feather. Towards a Unified Approach to the Representation of, and Reasoning with, Probabilistic Risk Information about Software and its System Interface. In *Proceedings of the 15th IEEE International Symposium on Software Software Reliability Engineering*, pages 391–402. IEEE Computer Society Press, November 2004.

[75] N. Fenton and Martin Neil. Managing risk in the modern world - application of bayesian networks, November 2007. London Mathematical Society.

[76] N. E. Fenton and M. Neil. Combining evidence in risk analysis using bayesian networks. White Paper W0704/01, Agena, 2004.

[77] Jose Figueira, Salvatore Greco, and Matthias Ehrgott, editors. *Multiple Criteria Decision Analysis:State of the Art Surveys*, volume 78 of *International Series in Operations Research & Management Science*. Springer, 1 edition, October 2005. ISBN 038723067X.

[78] M.P. Georgeff and A.L. Lansky. Reactive Reasoning and Planning. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 677–682. American Association for Artificial Intelligence, 1987.

[79] M. Ghallab, A. Howe, C. Knoblock, D. McDermott, A. Ram, M. Veloso, D. Weld, and D. Wilkins. PDDL – The Planning Domain Definition Language. In *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, 1998.

[80] Paolo Giorgini, John Mylopoulos, Eleonora Nicchiarelli, and Roberto Sebastiani. Formal Reasoning Techniques for Goal Models. *Journal of Data Semantics*, 1(1):1–20, October 2003.

[81] Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone. Modeling Security Requirements Through Ownership, Permission and Delegation. In *Proceedings of the 13th IEEE International Requirements Engineering Conference*, pages 167–176. IEEE Computer Society Press, 2005.

[82] Paolo Giorgini, Fabio Massacci, John Mylopoulos, and Nicola Zannone. Requirements Engineering for Trust Management: Model, Methodology, and Reasoning. *International Journal of Information Security*, 5(4):257–274, 2006.

[83] Martin Glinz. A risk-based, value-oriented approach to quality requirements. *Software, IEEE*, 25 (2):34–41, 2008. ISSN 0740-7459.

[84] Paul Goodman. *Software Metrics: Best Practices for Successful IT Management*. Rothstein Associates Inc., Publisher, 2004.

[85] Bjrn Axel Gran, Nikos Stathiakis, Gustav Dahll, Rune Fredriksen, Atoosa P-J. Thunem, Eva Henriksen, Eva Skipenes, Mass Soldal Lund, Ketil Stlen, Siv Hilde Houmb, Eirik Mork-Knudsen, and Erik Dagfinn Wislff. The coras methodology for model-based risk assessment. Technical Report D2.4, CORAS Consortium, June 2003.

[86] Charles Haley, Robin Laney, Jonathan Moffett, and Bashar Nuseibeh. Using trust assumptions with security requirements. *Requirements Engineering Journal*, 11(2):138–151, April 2006.

[87] Charles Haley, Robin Laney, Jonathan Moffett, and Bashar Nuseibeh. Security requirements engineering: A framework for representation and analysis. *IEEE Transactions on Software Engineering*, 34(1):133–153, 2008. ISSN 0098-5589.

[88] Joseph Y. Halpern. *Reasoning About Uncertainty*. The MIT Press, 2003.

[89] Charles Handy. Trust and the virtual organization. *Harvard Business Review*, 73, 1995.

[90] Robert Hanmer. *Patterns for Fault Tolerant Software*. Wiley, December 2007. ISBN 0470319798.

[91] K.M. Hansen, A.P. Ravn, and V. Stavridou. From safety analysis to software requirements. *IEEE Transactions on Software Engineering*, 24(7):573–584, 1998. ISSN 0098-5589.

[92] G. Helmer, J. Wong, M. Slagell, V. Honavar, L. Miller, and R. Lutz. A Software Fault Tree Approach to Requirements Analysis of an Intrusion Detection System. *Requirements Engineering Journal*, 7(4):207–220, 2002.

[93] Frederick S. Hillier and Gerald J. Lieberman. *Introduction to Operations Research*. McGraw-Hill, 2005.

[94] Glyn A. Holton. Defining Risk. *Financial Analyst Journal*, 60(6):19–25, 2004.

[95] S.H. Houmb, G. Georg, R. France, J. Bieman, and J. Jurjens. Cost-benefit trade-off analysis using bbn for aspect-oriented risk-driven development. In *Proceedings of 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS '05)*, pages 195–204, 2005.

[96] N. Howden, R. Ronnquist, A. Hodgson, and A. Lucas. JACK Intelligent Agents-Summary of an Agent Infrastructure. In *Proceedings of the 5th International Conference on Autonomous Agents*. ACM Press, 2001.

[97] Douglas W. Hubbard. *How to Measure Anything: Finding the Value of "Intangibles" in Business*. Wiley, 1 edition, August 2007. ISBN 0470110120.

[98] Marnie L. Hutcheson. *Software Testing Fundamentals: Methods and Metrics*. John Wiley & Sons, 2003.

[99] ICAO. Aircraft accident and incident investigation. ICAO Annex 13, July 2001.

[100] IEC. Fault Tree Analysis (FTA). IEC 61025, 1990.

[101] Project Management Institute. *A Guide to the Project Management Body of Knowledge, Third Edition*. Project Management Institute, 3 edition, November 2004. ISBN 193069945X.

[102] IPC-4 Homepage. International Planning Competition 2004. http://ls5-www.cs.uni-dortmund.de/ edelkamp/ipc-4/, 2006. accessed at November 2006.

[103] ISO. Information technology - security techniques - code of practice for information security management. ISO 17799, 2000.

[104] ISO/IEC. Systems and software engineering - life cycle processes - risk management. ISO/IEC 16085, 2006.

[105] ISO/IEC. Information Technology - Security Techniques - Information Security Management Systems - Requirements. ISO/IEC 27001, 2005.

[106] ISO/IEC. Information Technology - Security Techniques - Code of Practice for Information Security Management. ISO/IEC 27002, 2005.

[107] ISO/IEC. Information Technology - Security Techniques - Information Security Risk Management. ISO/IEC 27005, 2008.

[108] ISO/IEC. Corporate Governance of Information Technology. ISO/IEC 38500, 2008.

[109] ISO/IEC. Risk Management-Vocabulary-Guidelines for Use in Standards. ISO/IEC Guide 73, 2002.

[110] Michael Jackson. *Problem Frames: Analysing & Structuring Software Development Problems*. Addison-Wesley Professional, 2000. ISBN 020159627X.

[111] Sushil Jajodia, Pierangela Samarati, V. S. Subrahmanian, and Eliza Bertino. A unified framework for enforcing multiple access control policies. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, pages 474–485, Tucson, Arizona, United States, 1997. ACM Press. ISBN 0-89791-911-4.

[112] Andrew Jaquith. *Security Metrics: Replacing Fear, Uncertainty, and Doubt*. Addison Wesley, 2007.

[113] Adam Jolly. *Managing Business Risk: A Practical Guide to Protecting Your Business*. Kogan Page Business Books, 2003.

[114] A. Jøsang and S.L. Presti. Analysing the Relationship Between Risk and Trust. In *Proceedings of the Second International Conference on Trust Management*, volume 2995 of *Lecture Notes in Computer Science*, pages 135–145. Springer-Verlag, 2004.

[115] Audun Jøsang, Daniel Bradley, and Svein J. Knapskog. Belief-Based Risk Analysis. In *Proceedings of the 2nd Workshop on Australasian Information Security, Data Mining and Web Intelligence, and Software Internationalisation*, pages 63–68, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[116] Jan Jürjens. *Secure Systems Development With UML*. Springer, 2005.

[117] D. Kahneman and A. Tversky. Prospect theory: an analysis of decision under risk. *Econometrica*, 47(2):263–291, 1979.

[118] R. S. Kaplan and D. P. Norton. The balanced scorecard: Measures that drive performance. *Harvard Business Review*, 83:172, 1992.

[119] S. Karim, C. Heinze, and S. Dunn. Agent-Based Mission Management for a UAV. In *Proceedings of the 2004 of Intelligent Sensors, Sensor Networks and Information Processing Conference*, pages 481–486. IEEE Press, 2004.

[120] Samin Karim and Clint Heinze. Experiences with the design and implementation of an agent-based autonomous uav controller. In *Proceedings of 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 19–26, New York, NY, USA, 2005. ACM Press. ISBN 1-59593-093-0.

[121] Nadzeya Kiyavitskaya, Rocco Moretti, Maurizio Sebastianis, and Nicola Zannone. Project Report on the Initial Analysis of (Early) Requirements of Domain 1. TOCAI Deliverable D2.1, DIT - University of Trento, 2007.

[122] Trevor A Kletz. HAZOP - Past and Future. *Reliability Engineering and System Safety*, 55(3): 263–266, March 1997.

[123] P. Kocher, J. Jaffe, and B. Jun. Introduction to Differential Power Analysis and Related Attacks. http://www.cryptography.com/technology/dpa/DPATechnicalInfo.PDF, 1998.

[124] A. N. Kolmogorov. *Foundations of the Theory of Probability*. Chelsea Publishing Company, 2 edition, 1956. ISBN 0828400237.

[125] K. Kosanke, F. Vernadat, and M. Zelm. Cimosa: enterprise engineering and integration. *Computers in Industry*, 40(2-3):83–97, November 1999.

[126] Sanjeev Kumar and Philip R. Cohen. Towards a Fault-Tolerant Multi-Agent System Architecture. In *Proceedings of the 4th International Conference on Autonomous Agents*, pages 459–466, New York, NY, USA, 2000. ACM Press. ISBN 1-58113-230-1.

[127] J. Kwisthout and M. Dastani. Modelling uncertainty in agent programming. In *Proceedings of the Third International Workshop on Declarative Agent Languages and Technologies*, volume 3904 of *Lecture Notes in Computer Science*, pages 17–32. Springer-Verlag, 2005.

[128] H. Lacoheea, A.D. Phippenb, and S.M. Furnell. Risk and Restitution: Assessing How Users Establish Online Trust. *Computers & Security*, 25(7):286–293, October 2006.

[129] Carl E. Landwehr, Alan R. Bull, John P. McDermott, and William S. Choi. A Taxonomy of Computer Program Security Flaws. *ACM Computing Surveys*, 26(3):211–254, 1994. ISSN 0360-0300.

[130] J. Lauber, C. Steger, and R. Weiss. Autonomous Agents for Online Diagnosis of a Safety-critical System Based on Probabilistic Causal Reasoning. In *Proceedings of the 4th International Symposium on Autonomous Decentralized Systems*, pages 213–219, Washington, DC, USA, 1999. IEEE Computer Society Press. ISBN 0-7695-0137-0.

[131] Seok-Won Lee, Robin Gandhi, Divya Muthurajan, Deepak Yavagal, and Gail-Joon Ahn. Building Problem Domain Ontology from Security Requirements in Regulatory Documents. In *Proceedings of the 2006 Workshop on Software Engineering for Secure Systems Building Trustworthy Applications*, pages 43–50, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-411-1.

[132] SW Lee, RA Gandhi, and G. Ahn. Security Requirements Driven Risk Assessment for Critical Infrastructure Information Systems. In *Proceedings of the 3rd Symposium on Requirements Engineering for Information Security*, 2005.

[133] Luncheng Lin, Bashar Nuseibeh, Daniel Ince, Michael Jackson, and Jonathan Moffett. Analysing security threats and vulnerabilities using abuse frames. Technical report, The Open University, 2003.

[134] Luncheng Lin, B. Nuseibeh, D. Ince, and M. Jackson. Using abuse frames to bound the scope of security problems. In *Proceedings of the 12th IEEE International Requirements Engineering Conference*, pages 354–355, 2004. ISBN 1090-705X.

[135] Dennis Vector Lindley. *Understanding Uncertainty*. John Wiley & Sons, 1923.

[136] H.A. Linstone and M. Turoff. *The Delphi Method: Techniques and Applications*. Addison-Wesley Pub. Co, 1975.

[137] Lin Liu, Eric S. K. Yu, and John Mylopoulos. Security and Privacy Requirements Analysis within a Social Setting. In *Proceedings of the 11th IEEE International Requirements Engineering Conference*, pages 151–161, 2003.

[138] Torsten Lodderstedt, David Basin, and Jürgen Doser. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In *Proceedings of the 5th International Conference on the Unified Modeling Language – the Language and its Applications*, volume 2460 of *Lecture Notes in Computer Science*, pages 426–441. Springer-Verlag, 2002.

[139] LPG Homepage. LPG-td Planner. http://zeus.ing.unibs.it/lpg/, 2006. accessed at 2006.

[140] Fabio Massacci and Nicola Zannone. Detecting Conflicts between Functional and Security Requirements with Secure Tropos: John Rusnak and the Allied Irish Bank. In *Social Modeling for Requirements Engineering*. MIT Press, 2007. To appear.

[141] Nicholas Mayer, Eugine Dobuis, and Andre Rifaut. Requirements Engineering for Improving Business/IT Alignment in Security Risk Management Methods. In *Proceedings of the 3rd International Conference Interoperability for Enterprise Software and Applications*, 2007.

[142] Deborah G. Mayo and Rachelle D. Hollander. *Acceptable Evidence: Science and Values in Risk Management*. Oxford University Press US, 1991.

[143] John McCarthy. Ascribing mental qualities to machines. Technical Report Tech. Rept. Memo 326, Stanford AI Lab, Stanford, 1979.

[144] J. McDermott and C. Fox. Using Abuse Case Models for Security Requirements Analysis. In *Proceedings of 15th Annual Computer Security Applications Conference*, pages 55–64, Phoenix, AZ, USA, 1999.

[145] J. D. Meier, Alex Mackman, Srinath Casireddy, Michael Dunner, Ray Escamilla, and Anandha Murukan. *Improving Web Application Security: Threats and Countermeasures*. Microsoft, 2003.

[146] R. Miles and K. Hamilton. *Learning UML 2.0*. O'Reilly, 2006.

[147] Ali Mosleh, E. Richard Hilton, and Peter S. Browne. Bayesian probabilistic risk analysis. *SIGMETRICS Perform. Eval. Rev.*, 13(1):5–12, 1985. ISSN 0163-5999.

[148] H. Mouratidis and P. Giorgini. Secure Tropos: Dealing effectively with Security Requirements in the development of Multiagent Systems. In *Safety and Security in Multiagent Systems*, Lecture Notes in Computer Science. Springer-Verlag, 2006.

[149] Haralambos Mouratidis and Paolo Giorgini. Security tropos: A security-oriented extension of the tropos methodology. *International Journal of Software Engineering and Knowledge Engineering*, 17(2):2085–309, 2007.

[150] John Mylopoulos, Lawrence Chung, and Brian Nixon. Representing and using nonfunctional requirements: a process-oriented approach. *IEEE Transactions on Software Engineering*, 18(6): 488–497, 1992. ISSN 0098-5589.

[151] NASA. Presidential commission report on space shuttle challenger accident. Presidential commission report, NASA, 1986. available at `http://science.ksc.nasa.gov/shuttle/missions/51-l/docs/rogers-commission/table-of-contents.html`.

[152] Peter G. Neumann. RISKS-LIST: RISKS-FORUM Digest. http://catless.ncl.ac.uk/Risks/, 2008. accessed at 2008-05-27.

[153] Bashar Nuseibeh. Weaving together requirements and architectures. *Computer*, 34(3):115–119, 2001. ISSN 0018-9162.

[154] OGC. *Information Technology Infrastructure Library (ITIL)*. Office of Government Commerce, 2007. Version 3.

[155] A. O'Hagan, C. E. Buck, A. Daneshkhah, J. R. Eiser, P. H. Garthwaite, D. J. Jenkinson, J. E. Oakley, and T. Rakow. *Uncertain Judgements: Eliciting Experts' Probabilities*. Wiley, 2006.

[156] OMG. *Business Process Modeling Notation Specification*. Object Management Group, v 1.0 edition, February 2006.

[157] OMG. OMG Unified Modeling Language (OMG UML), Superstructure. http://www.omg.org/spec/UML/2.1.2/, 2007. UML 2.1.2.

[158] Sylvia Osborn. Mandatory access control and role-based access control revisited. In *Proceedings of the 2nd ACM Workshop on Role-Based Access Control*, pages 31–40, New York, NY, USA, 1997. ACM Press. ISBN 0-89791-985-8.

[159] Fiona D. Patterson and Kevin Neailey. A risk register database system to aid the management of project risk. *International Journal of Project Management*, 20(5):365–374, July 2002.

[160] Judea Pearl. *Causality - Models, Reasoning, and Inference*. Cambridge University Press, 2001. ISBN 0521773628, 9780521773621.

[161] Joachim Peer. Web Service Composition as AI Planning – a Survey. Technical report, University of St. Gallen, 2005.

[162] Charles P. Pfleeger and Shari Lawrence Pfleeger. *Security in Computing*. Prentice-Hall, 4th edition, 2006.

[163] Alexander Pokahr, Lars Braubach, and Winfried Lamersdorf. Jadex: A BDI Reasoning Engine. In *Multi-Agent Programming: Languages, Platforms and Applications*, pages 149–174. Springer Science, September 2005.

[164] Anand S. Rao and Michael P. Georgeff. Modeling Rational Agents within a BDI-Architecture. In James Allen, Richard Fikes, and Erik Sandewall, editors, *Proceedings of 2nd International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484. Morgan Kaufmann publishers Inc.: San Mateo, CA, USA, 1991. ISBN 1-55860-165-1.

[165] Anand S. Rao and Michael P. Georgeff. BDI Agents: From Theory to Practice. In *Proceedings of 1st International Conference on Multi-Agent Systems*, pages 312–319, 1995.

[166] G. Reniers, W.Dullaert, and K. Soudan. A Domino Effect Evaluation Model. Working papers, University of Antwerp, Faculty of Applied Economics, December 2004.

[167] David Rice. *Geekonomics: The Real Cost of Insecure Software*. Addison-Wesley Professional, 1 edition, December 2007. ISBN 0321477898.

[168] David F. Rico. *ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers*. J. Ross Publishing, 2004.

[169] Stephen P. Robbins. *Organizational Behavior, Concepts, Controversies, Applications - Seventh Edition*. Prentice-Hall, 7th edition, 1996. ISBN 0132285118.

[170] G. Ropohl. Philosophy of socio-technical systems. *Society for Philosophy and Technology*, 4(3): 59–71, 1999.

[171] Sheldon M. Ross. *Introduction to Probability Models*. Academic Press, 1997.

[172] G. G. Roy and T. L. Woodings. A Framework for Risk Analysis in Software Engineering. In *Proceedings of the Seventh Asia-Pacific Software Engineering Conference (APSEC '00)*, page 441, Washington, DC, USA, 2000. IEEE Computer Society Press. ISBN 0-7695-0915-0.

[173] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 2nd edition, 2003.

[174] T L Saaty. How to Make a Decision: The Analytic Hierarchy Process. *European Journal of Operational Research*, 48(1):9–26, 1990.

[175] T.L. Saaty and G. Hu. Ranking by Eigenvector Versus Other Methods in the Analytic Hierarchy Process. *Applied Mathematics Letters*, 11(4):121–125, July 1998.

[176] Wesely C. Salomon. Probabilistic Causality. In Ernest Sosa and Michael Tooley, editors, *Causation*, Oxford Readings in Philosophy. Oxford Press, 1993.

[177] Paul Sarbanes and Michael G. Oxley. Public company accounting reform and investor protection act. Washington, DC: Government Printing Office, 2002.

[178] B. Schneier. Attack Trees: Modeling Security Threats. *Dr. Dobbs Journal*, 12(24):21–29, 1999.

[179] Bruce Schneier. *Beyond Fear: Thinking Sensibly about Security in an Uncertain World*. Springer, 2003.

[180] Markus Schumacher, Eduardo Fernandez-Buglioni, Duane Hybertson, Frank Buschmann, and Peter Sommerlad. *Security Patterns: Integrating Security and Systems Engineering*. Wiley, 1 edition, March 2006. ISBN 0470858842.

[181] Roberto Sebastiani, Paolo Giorgini, and John Mylopoulos. Simple and Minimum-Cost Satisfiability for Goal Models. In *Proceedings of the 16th Conference On Advanced Information Systems Engineering*, volume 3084 of *Lecture Notes in Computer Science*, pages 20–33. Springer-Verlag Heidelberg, June 2004.

[182] Glenn Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ, 1976.

[183] D. Shapiro and R. Shachter. User-Agent Value Alignment. In *Proceedings of The Eighteenth National Conference on Artificial Intelligence*. American Association for Artificial Intelligence, 2002.

[184] Yoav Shoham. Agent-Oriented Programming. *Artificial Intelligence*, 60(1):51–92, 1993. ISSN 0004-3702.

[185] Gerardo I. Simari and Simon Parsons. On the relationship between mdps and the bdi architecture. In *Proceedings of 5th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1041–1048, New York, NY, USA, 2006. ACM Press. ISBN 1-59593-303-4.

[186] Guttorm Sindre and Andreas L. Opdahl. Eliciting Security Requirements With Misuse Cases. *Requirements Engineering Journal*, 10(1):34–44, January 2005.

[187] Paul Slovic. Perceived Risk, Trust, and Democracy. *Risk Analysis*, 13(6):675–682, 1993.

[188] Philippe Smets. What is dempster-shafer's model? In *Advances in the Dempster-Shafer theory of evidence*, pages 5–34. John Wiley & Sons, Inc., 1994. ISBN 0-471-55248-8.

[189] Adam Smith. *An Inquiry into the Nature and Causes of the Wealth of Nations*. www.elecbook.com, 1956. ISBN 1593775415, 9781593775414.

[190] M.J. Smithson. *Ignorance and Uncertainty: Emerging Paradigms*. Springer-Verlag, 1989.

[191] Bjørnar Solhaug, Dag Elgesem, and Ketil Stølen. Why Trust is not Proportional to Risk. In *Proceedings of the Second International Conference on Availability, Reliability and Security*. IEEE Press, 2007.

[192] Ian Sommerville. *Software Engineering*. Addison Wesley, 7th edition, May 2004.

[193] SRC - Safety Regulation Commission. ATM Contribution to Aircraft Accidents/Incidents: Review and Analysis of Historical Data. Technical Report SRC DOC 2 (4.0), EUROCONTROL, May 2005.

[194] SRC - Safety Regulation Commission. Annual Safety Report 2006. Technical report, EUROCONTROL, November 2006.

[195] Jussi Stader. Results of the enterprise project. In *Proceedings of the 16th Annual Conference of the British Computer Society Specialist Group on Expert Systems*, pages 888–893, 1996.

[196] Michael Stamatelatos, William Vesely, Joanne Dugan, Joseph Fragola, Joseph Minarick, and Jan Railsback. *Fault Tree Handbook with Aerospace Applications*. NASA, 2002.

[197] V. Stanford. Using pervasive computing to deliver elder care. *Pervasive Computing, IEEE*, 1: 10–13, 2002. ISSN 1536-1268.

[198] James Surowiecki. *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations*. Doubleday, May 2004. ISBN 0385503865.

[199] Alistair Sutcliffe and A.G. Sutcliffe. *The Domain Theory: Patterns for Knowledge and Software Reuse*. CRC, 1 edition, 2002. ISBN 0805839518.

[200] The IT Governance Institute. *CoBIT - Framework Control Objectives Management Guidelines Maturity Models*. The IT Governance Institute, 4.1 edition, 2007.

[201] Paolo Tonella, Paolo Avesani, and Angelo Susi. Using the case-based ranking methodology for test case prioritization. In *Proceedings of the 22nd IEEE International Conference on Software Maintenance*, pages 123–133. IEEE Computer Society, 2006. ISBN 0-7695-2354-4.

[202] E. Trist. The evolution of socio-technical systems. *Occasional Paper*, 2, 1981.

[203] US-DoD. Work Breakdown Structure. MIL-HDBK-881, 1998.

[204] US-DoD. *Department of Defense Information Technolgoy Security Certification and Accreditation Process (DITSCAP) Application Manual*. US-Department of Defense, July 2000.

[205] US-DoD. Military Standard, Procedures for Performing a Failure Mode, Effects, and Critical Analysis. MIL-STD-1629A, 1980.

[206] U.S. NCSA - NHTSA. Fatality Analysis Reporting System General Estimates System - 2006 Data Summary. http://www-nrd.nhtsa.dot.gov/CMSWeb/, 2008. last access 04.08.2008.

[207] U.S. NTSB. Aviation Accident Statistics. http://www.ntsb.gov/aviation/Table2.htm, 2008. last access 04.08.2008.

[208] Pascal van Eck, Jaap Gordijn, and Roel Wieringa. Risk-Driven Conceptual Modeling of Outsourcing Decisions. In Paolo Atzeni, editor, *Proceedings of the 23rd International Conference on Conceptual Modeling*, volume 3288 of *Lecture Notes in Computer Science*, pages 709–723. Springer-Verlag, 2004. `http://www.cs.utwente.nl/~patveck/redirect.php?p=ER04`.

[209] A. van Lamsweerde, S. Brohez, R. De Landtsheer, and D. Janssens. From System Goals to Intruder Anti-Goals: Attack Generation and Resolution for Security Requirements Engineering. In *Proceedings of the 2nd International Workshop on Requirements for High Assurance Systems*, 2003.

[210] Axel van Lamsweerde and Emmanuel Letier. Handling Obstacles in Goal-Oriented Requirements Engineering. *IEEE Transactions on Software Engineering*, 26(10):978–1005, 2000. ISSN 0098-5589.

[211] Axel van Lamsweerde, Emmanual Letier, and Robert Darimont. Managing Conflicts in Goal-Driven Requirements Engineering. *IEEE Transactions on Software Engineering*, 24(11):908–926, 1998. ISSN 0098-5589.

[212] Martin van Staveren. *Uncertainty and Ground Conditions: A Risk Management Approach*. Elsevier, 2006. ISBN 0080462677.

[213] Achille Varzi. *Stanford Encyclopedia of Philosophy*, chapter Mereology. Metaphysics Research Lab, CSLI, Stanford University, 2003. http://plato.stanford.edu/entries/mereology/.

[214] D. Verdon and G. McGraw. Risk analysis in software design. *IEEE Security and Privacy*, 2(4): 79–84, 2004. ISSN 1540-7993.

[215] B. Vidolov, J. De Miras, and S. Bonnet. AURYON - A Mechatronic UAV Project Focus on Control Experimentations. In *Proceedings of the International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC '05)*, volume 1, pages 1072–1078. Washington, DC, USA, IEEE Computer Society Press, 2005.

[216] L. von Bertalanffy. The theory of open systems in physics and biology. *Science*, 111(2872):23–29, 1950.

[217] Ludwig von Bertalanffy. An outline of general system theory. *The British Journal for the Philosophy of Science*, 1(2):134–165, August 1950. ISSN 00070882.

[218] D. Vose. *Risk Analysis: A Quantitative Guide*. Wiley, 2000.

[219] P. Wallis, R. Ronnquist, D. Jarvis, and A. Lucas. The Automated Wingman - Using JACK Intelligent Agents for Unmanned Autonomous Vehicles. In *Proceedings of the IEEE Aerospace Conference*, volume 5, pages 5–2615–5–2622 vol.5. IEEE Press, 2002.

[220] E.U. Weber, A.R. Blais, and N.E. Betz. A Domain-Specific Risk-Attitude Scale: Measuring Risk Perceptions and Risk Behaviors. *Journal of Behavioral Decision Making*, 15(4):263–290, 2002.

[221] Daniel S. Weld. Recent Advances in AI Planning. *AI Magazine*, 20(2):93–123, 1999.

[222] Mathias Weske. *Business Process Management: Concepts, Languages, Architectures*. Springer-Verlag New York, 2007. ISBN 3540735216.

[223] D. J. White. *Markov Decision Processes*. John Wiley & Sons, 1993.

[224] Weihang Wu and Tim Kelly. Combining bayesian belief networks and the goal structuring notation to support architectural reasoning about safety. In *Computer Safety, Reliability, and Security*, 2007.

[225] Edward Yourdon. *Just Enough Structured Analysis*. Yourdon, 2006.

[226] Eric Yu. *Modelling Strategic Relationships for Process Engineering*. PhD thesis, University of Toronto, Department of Computer Science, 1995.

[227] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, 100 (Suplement 1):9–34, 1999.

[228] Emmanuele Zambon, Damiano Bolzoni, Sandro Etalle, and Marco Salvato. A Model Supporting Business Continuity Auditing and Planning in Information Systems. In *Proceedings of the Second International Conference on Internet Monitoring and Protection*, page 33, 2007.

[229] Franco Zambonelli, Nicholas R. Jennings, and Michael Wooldridge. Developing multiagent systems: The gaia methodology. *ACM Transactions on Software Engineering and Methodology*, 12 (3):317–370, 2003.

[230] Nicola Zannone. *A Requirements Engineering Methodology for Trust, Security and Privacy*. PhD thesis, International Doctorate School in Information and Communication Technologies, University of Trento, 2007.