University of Namur

MASTER IN COMPUTER SCIENCE

Modelling security during early requirements

contributions to and usage of a domain model for information system security risk management

Genon, Nicolas

Award date:
2007

Facultés Universitaires Notre-Dame de la Paix de Namur
Institut d'Informatique

Modelling Security during Early Requirements:

Contributions to and Usage of a Domain Model for Information System Security Risk Management

Nicolas Genon

Mémoire présenté en vue de l'obtention
du grade de Maître en Informatique

Année académique 2006-2007

Abstract

Security has become one of the most important concerns when designing **I**nformation **S**ystems (IS). In front of the increasing number of aspects of our life relying on network communication and on computerized processes, and the explosing number of Internet attacks, the industries have to protect their assets against such risk. Even if the need of practical approaches to ensure that security objectives are guaranteed is obvious, it seems that practitioners strain to take care about. The reason is certainly that they are faced with hundreds of security standards, frameworks and methods. Each of them focuses on different aspects of the security and uses different terminologies, leading to a situation without consensus between the practioners.

This lack of consensus and approaches to ensure security in IS motivates researchers to enrich actual **S**ecurity **R**isk **M**anagement (SRM) approaches and, in the same time, to investigate new methods to manage security in systems. This thesis comes within the scope of N. Mayer's work which investigates the **IS** **S**ecurity **R**isk **M**anagement (ISSRM). This thesis contributes to the elaboration of the ISSRM domain model by completing it with relationships between its concepts. Secondly, it gives proposals to enhance the ISSRM domain model from results obtained from its alignment with security modelling languages. Finally, it suggests how to improve these security modelling languages in order to support all the concepts from the ISSRM domain model.

Keywords: Security Risk Management, ISSRM domain model, KAOS, secure TROPOS, concepts alignment.

Résumé

La sécurité est devenue une des préoccupations essentielles lors de l'élaboration de **S**ystèmes d'*I*nformation (SI). Confrontées au nombre croissant d'aspects de notre vie qui reposent sur les réseaux de communication et les processus informatisés, ainsi que le nombre exponentiel d'attaques via Internet, les industries se doivent de protéger leurs assets. Même s'il est évident qu'ils ont besoin d'approches pratiques assurant que les objectifs de sécurité sont atteints, les practiciens peinent à s'y intéresser. La raison majeure est certainement qu'ils se retrouvent submergés par des centaines de standards, de frameworks et de méthodes traitant de la sécurité. Chacun d'entre eux se focalise sur un aspect différent de la sécurité en employant une terminologie distincte des autres. Ceci aboutit à une situation où aucun consensus ne semble possible parmi les practiciens.

Ce manque de consensus et d'approches pour assurer la sécurité des SI motive les chercheurs à enrichir les approches actuelles de gestion des risques de sécurité et, en même temps, à s'intéresser à de nouvelles méthodes. Ce mémoire s'inscrit dans le cadre du travail de N. Mayer qui étudie le **G**estion **d**es **R**isques de **S**écurité des **SI** (GdR SSI). Ce mémoire contribue à l'élaboration de l'ISSRM domain model en y intégrant des relations entre ses concepts. Deuxièmement, il fournit des propositions pour améliorer l'ISSRM domain model à partir des résultats de son alignement avec des langages de modélisation de la sécurité. Finalement, il suggère des perfectionnements à apporter à ces langages de modélisation de la sécurité afin qu'ils prennent en compte tous les concepts de l'ISSRM domain model.

Mots clés : Gestion des risques de sécurité, ISSRM domain model, KAOS, secure TROPOS, alignement de concepts.

Acknowledgments

I wish to thank Nicolas Mayer for his hospitality during my internship at the Centre de Recherche Public Henri Tudor in the Grand-Duchy of Luxembourg. Thanks to his guidance, his availability and his general help.

I would like to express my gratitude to Prof. Dr. Patrick Heymans and Dr. Raimundas Matulevičius, my supervisors at the University of Namur, for their support in determining the objectives of this work and helping me to reach them. I would also like to thank them for all their insightful remarks on the many drafts of this thesis.

I acknowledge the Centre de Recherche Public Henri Tudor and the University of Namur for their warm welcome and for the place they gave me to work.

A special thanks to Germain Saval, Jean-Christophe Trigaux, Alain Vagner and Luc Dehand for the very interesting discussions that we had.

I would also like to acknowledge Laurence Mathu who used her talents to correct my English and Jean-François Wauthy who improved the layout of this thesis.

Je tiens aussi à remercier Amélie et ma famille pour tout ce qu'ils ont fait pour moi durant la préparation de ce mémoire.

Contents

List of Figures

List of Tables

Foreword

My internship took place at the *Centre de Recherche Public Henri Tudor* in Luxembourg. We described briefly the organisational structure of the research center.

Centre de Recherche Public Henri Tudor

The **C**entre de **R**echerche **P**ublic **H**enri **T**udor (CRPHT), whose description is based on `http://www.tudor.lu` and logo is presented in Fig. 1, is a public organisation of private law situated in the Grand-Duchy of Luxembourg. It is upheld by a large industrial and institutional partnership. Founded in 1987, it aims to favour technological innovation in private and public sectors. It offers a set of services and activities : R&D projects, technologies transfer, technological assistance and guidance, high level training and qualification, thesis research, innovation networks animation, assistance in the creation of high-tech entreprises.

Figure 1: CRPHT Logo

As we can see on Fig. 2, the CRPHT is composed of eight departments :

- the “Laboratoire de Technologies Industrielles et Matériaux” (LTI);
- the “Centre d’Innovation par les Technologies de l’Information” (CITI);
- the “Centre de Ressources des Technologies pour l’Environnement” (CRTE);
- the “Centre de Ressources des Technologies pour la Santé” (CR SANTEC);
- the “Centre de Ressources des Technologies de l’Information pour le Bâtiment” (CRTI-B);
- the “Service de Formation continue de l’Ingénieur et du Cadre, SITec®”;
- the “Centre de Veille Technologique et Normative (CVT), l’incubateur d’entreprises de technologies innovantes ou Technoport”.

Departments are themselves composed by :

- the “plates-formes d’innovation” (PFI) that regroup around a national strategical aim, a pallet of partners and a lot of multidisciplinary research and transfer activities;
- the “unités scientifiques et technologiques (UST)” made up of specialized research, associated RDI and valuation activities;
- the “unités de services pour l’innovation”.

My training took place in the offices of the “centre d’innovation par les technologies de l’information” (CITI). Its aim is to lead, by research and guidance in the domain of innovation, the economy of Luxembourg to the most promising ICT applications. Direct support to partners and developing a network of the innovation actors of Luxembourg constitute the main objectives of this department. CITI counts more than a hundred R&D engineers shared out among five USTs and five PFIs as described in Fig. 3.

These five USTs are:

- Knowledge Economy and Innovation Management;
- Software Engineering;
- Innovation Project Management;
- Management of Organisations and IT Services;
- Reference Systems for Modelling and Certification.

The five PFIs are:

- Quality and Certification of Computer Services;
- Information Systems Security;
- Interoperability standards and E-Business;
- E-Learning, Knowledge Management and Networked Organisations;
- Statistical Studies and Prospective in the Knowledge Economy;

CITI’s competences are in the following domains :

- Information Systems Security, e-Business, e-Learning, Knowledge Management, Workflow;
- Innovation Strategies, Network of Innovation and learning organisations;
- Computerized exploitation models, Project Management, Improvement and Certification of computerized processes, Provider/Consumer relationship;
- Requirement Engineering, process modelling, software architecture and interoperability.

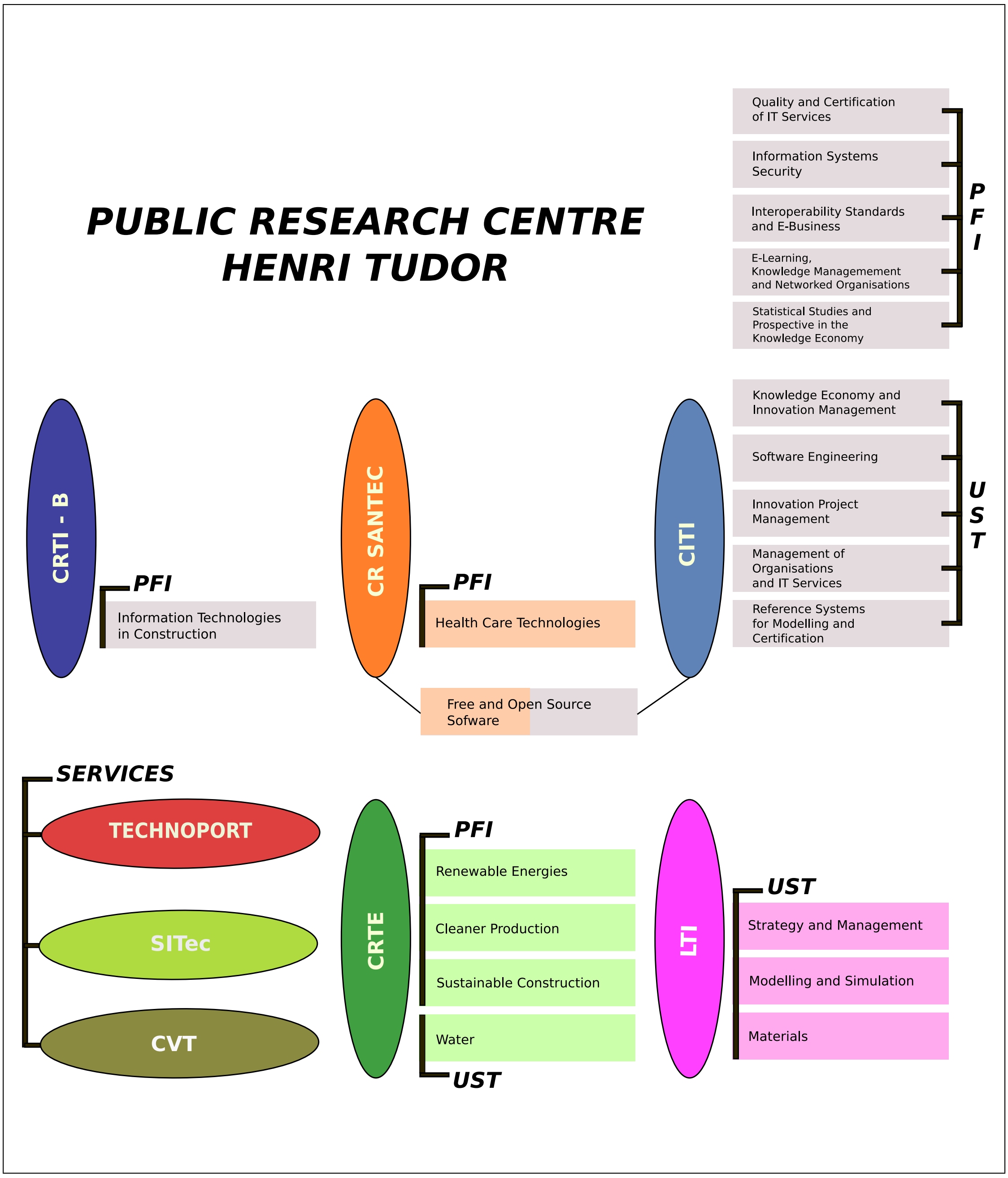

PUBLIC RESEARCH CENTRE
HENRI TUDOR
Quality and Certification of IT Services
Information Systems Security
Interoperability Standards and E-Business
E-Learning, Knowledge Managemement and Networked Organisations
Statistical Studies and Prospective in the Knowledge Economy
PFI
Knowledge Economy and Innovation Management
Software Engineering
Innovation Project Management
Management of Organisations and IT Services
Reference Systems for Modelling and Certification
UST
CRTI - B
PFI
Information Technologies in Construction
CR SANTEC
PFI
Health Care Technologies
CITI
Free and Open Source Sofware
SERVICES
TECHNOPORT
SITec
CVT
CRTE
PFI
Renewable Energies
Cleaner Production
Sustainable Construction
Water
UST
LTI
UST
Strategy and Management
Modelling and Simulation
Materials

Figure 2: Organisational structure of the CRPHT (adapted from `http://www.tudor.lu`)

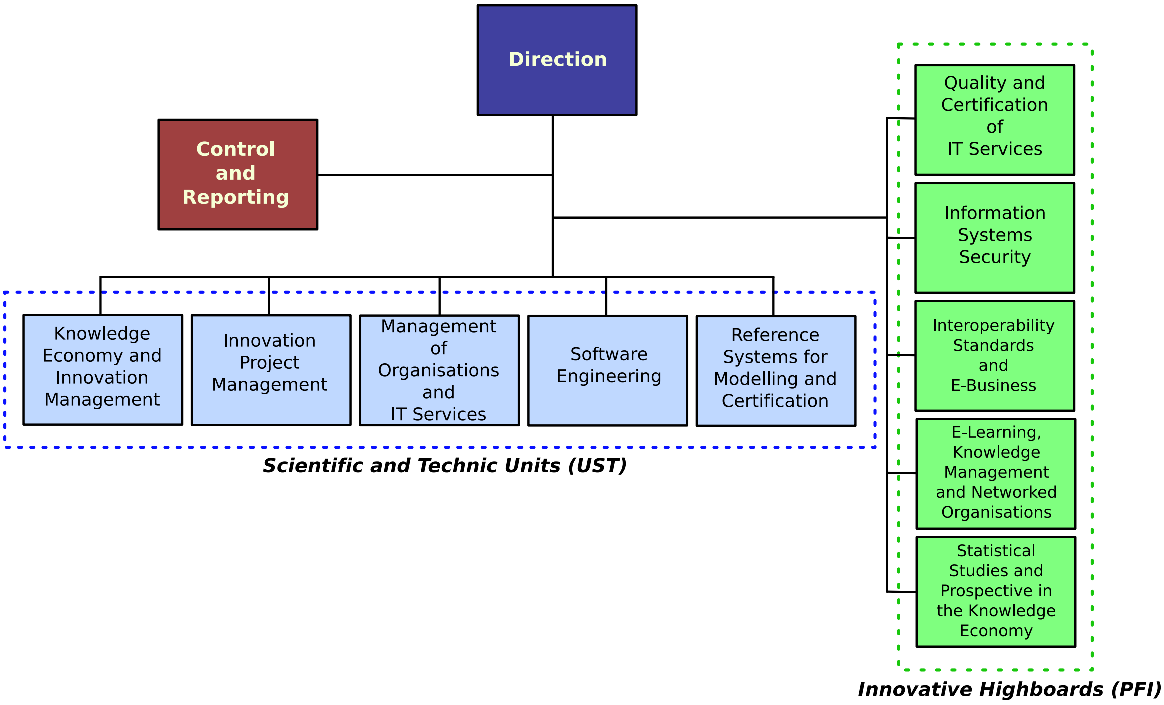

Direction
Control and Reporting
Knowledge Economy and Innovation Management
Innovation Project Management
Management of Organisations and IT Services
Software Engineering
Reference Systems for Modelling and Certification
Scientific and Technic Units (UST)
Quality and Certification of IT Services
Information Systems Security
Interoperability Standards and E-Business
E-Learning, Knowledge Management and Networked Organisations
Statistical Studies and Prospective in the Knowledge Economy
Innovative Highboards (PFI)

Figure 3: CITI Operating Chart (adapted from `http://www.tudor.lu`)

Chapter 1

Introduction

1.1 Security Concerns in Information Systems

Security. This word reflects the state of mind of our present-day society and not only in **I**nformation **T**echnology (IT) environments. Our society tries to take measures to insure security in all circumstances of our daily life: for example, highway code, evacuation planning, military training and security norms. Peculiarly after the extreme increase of Web attacks, IT specialists became aware of the significance of dealing with security in IT environments. From the beginning of computer sciences, failures in running systems were considered as inherent to IT systems, due to the fact that perfection is rather unreachable. Failures are bugs that are corrected during testing phases and when systems are already running. However, security concerns cannot be considered as failure in the running process of the software. They have to be viewed as system properties. For this reason, **R**isk **M**anagement (RM) has to be present in each step of the development of an **I**nformation **S**ystem (IS), from the earliest context analysis to the implementation and deployment of the IT solution.

However, making a new exhaustive elicitation of the potential risks is quite impossible. It is always possible to find a potential risk. Indeed, it is not rare to see that the solution adopted against a specific risk leads to the introduction of a new security risk. Computer analysts/specialists need methods to reveal potential security risks and they need knowledge, methods and tools to decide which are the revelant risks to consider.

The context around IS and their related risks is paramount. IS exist in business environments where making business is the first and the most important occupation. Risk assessment is constrained by financial means. Experts talk about **R**eturn **O**n **I**nvestments (ROI). The use of a method is always constrained by its envisioned ROI.

Another problem has to be addressed: *at which step(s) of IS development does RM have to be taken into account*? Usually, risk countermeasures are added as a layer over the business code of a software. This happens for mainly two reasons: first, legacy but still used systems have been designed without taking into account security concerns and thus security has to be implemented as a layer above the current business code. The second reason is that achieving a fully working system requires a large amount of time and adding security concerns also only increases it. IT specialists are under pressure to deliver running systems in time, and thus they often consider security as a possible extension to the software they are developing. Security is seen as a wish, not as a requirement. Adding a new security layer above running systems permits to increase its security level without having to redesign the entire system. But such a practice has also a recurrent problem: the chosen countermeasures are deployed with huge difficulties and are not always as suitable as possible to the corresponding risks. Software engineers have already drawn attention to the fact that the sooner a concept is investigated during software development, the cheaper the cost of its implementation will be.

Nowadays, according to [MH06], more than two hundred RM methods have been reported. Although a lot of these methods have been discarded for different reasons (no more supported, too abstract, too rigid, only guidelines), some are still supported by different organisations. This is a real proof of the current attention on security risks in IS. Unfortunately, these methods use some identical words with heterogeneous semantics and they do not tackle problems with the same process. This results in a large variety of RM methods in which practitioners try to make their way, taking some aspects from their favourite methods. But a RM method used in a given context cannot be easily reused in another. It is also difficult to migrate from a RM method (which is, for example, not anymore supported) to another one.

This special RM context shows the need for a **RM method** applicable from the *earliest IS engineering phases*. Such a method is currently being developed in the scope of a Ph.D. thesis by M. Nicolas Mayer. N. Mayer is a computer engineer working at the **C**entre de **R**echerche **P**ublic **H**enri **T**udor (CRPHT), where my internship took place. He conducts research on the development of an **I**nformation **S**ystem **S**ecurity **R**isk **M**anagement (ISSRM) domain model. Before asking *which are the different components of the ISSRM domain model*, we will gather all relevant reasons that justify the need of such a domain model. We have already introduced some of these reasons and we will complete this analysis.

1.1.1 The Rise of Security Concerns

All actions we undertake in our all days life rely more and more on network communications and computerized operations. In the same time, the number of attacks against Web components has an important increasing. Confronted to such a huge threat, industries and institutions became aware of the need to secure their assets. The threat is not new but its scale is a new contest concerning all analysts and developers of IS. The first response to the need of security and, in a broader view, to the need of RM was to add a security layer above existing systems. As we said before, this solution is quite imperfect and leads to **hard to maintain** systems. At the same time, the need of RM methods was identified and this domain was investigated. More than two hundred methods (see [MH06]) were created. However, from a situation in which IT specialists were disarmed, we reached another one in which these specialists are overflowed by the amount of different RM methods and, finally, no method succeeded to make itself the reference in RM domain. To explain the failure of finding a leader RM method, we need to investigate the RM phases covered by each of them. RM can be divided into three phases: a) *risk analysis*, b) *security requirements elicitation* and c) *implementation and management of countermeasures*. [Maytz] explicitly shows that only a very small subset of RM methods cover the entire RM process. The fact that methods that covered all phases did not become leader could be that a) they were too difficult to put into practice and b) they gathered no consensus among security specialists. So the main issues that a new RM method has to address are a) covering the entire RM process from risk analysis to implementation and management of countermeasures and b) being accepted by security and RM specialists.

1.1.2 Building a New RM Standard

Building a new RM standard requires to address the two issues previously discussed. This is what the ISSRM domain model developed by N. Mayer aims to do. To resolve the first issue a), N. Mayer elaborated a research method (discussed in detail in Chapter 4). This research method begins by analysing the most relevant and still supported RM standards (described in Chapter 5). After that, he gathers and aligns concepts from each of the retained RM standards in order to obtain the relevant RM concepts. Once these concepts and the relationships between them have been elicited, a new metamodel is designed, using these concepts.

The second issue, making our new RM standard accepted by RM specialists, is intrinsically linked with N. Mayer's research method. Indeed, as we will see in the sequel, intermediate versions of the RM standard will be repeatedly submitted to RM specialists, especially regarding to the alignment of concepts of the most current RM standards.

1.1.3 Objectives

My contribution to the research method tackles three different problems:

1. to investigate relationships between ISSRM domain model concepts that were elicited in previous works in order to complete and improve the ISSRM domain model;
2. to analyse how existing security modelling languages consider security concerns. We analyse what are the ISSRM domain model concepts that are treated in security modelling languages;
3. to elicit potential improvements of the ISSRM domain model – depending on the results of the analysis of security modelling languages – and also potential improvements to security modelling languages in order to make them support ISSRM concepts.

1.1.4 Overview

Chapter 2 aims to define foundational concepts for the rest of the thesis. These concepts are IS, **R**equirements **E**ngineering (RE), **S**oftware **D**esign **L**ifecycle (SDL) and Early Requirements. Chapter 3 focuses on the notions of security, Risk Management and RM process. The research method is presented in Chapter 4. Chapter 5 concerns the state of the art of the security and risk related sources, and some material resulting from initial steps of research method. Chapter 6 defines the ISSRM domain model. An introduction to security modelling languages is given in Chapter 7. Chapter 8 concerns the alignment of security modelling languages with the ISSRM domain model, and Chapter 9 validates the alignment with a case study. The validation and results obtained from the case study as discussed in Chapter 10. We finish the thesis with an investigation of future works.

Part I

Background

Chapter 2

Information System and Requirements Engineering

In this chapter, we introduce foundational notions for the rest of the thesis such as **Information System** and **Requirements Engineering** and other related terms.

2.1 Information System

Following [Hey] and [BP89] the term **I**nformation **S**ystem (IS) describes what the organisations conceive, realise and exploit to satisfy their need of information due to their organisational behaviours. An IS is composed of:

- information which is a partial representation of facts interesting for the organisation;
- information acquisition, memorization, transformation, search, presentation and communication processes;
- organisational rules that govern the execution of informational processing;
- human and technical resources required to make the IS work.

An IS **can** be computer-based, but does not need to be so. However, in the sequel of this thesis, we will use the term IS as a synonym of **computerized IS**. Indeed, the environments in which IS are currently developed are of **Software-Intensive Systems**. That is, systems composed of hardware and software but also human operators. The fact is that software on its own is useless because it needs to be executed on a hardware product. We call **Computer System** the set composed of the software and the hardware. However, Computer System is still, on its own, useless. It is only useful in an environment where it supports human activities. This thesis investigates the security of (computerized) IS.

To achieve a *good* IS – which is an IS that fully fulfills purposes it was built for – an engineering process is mandatory. Requirements Engineering, discussed in the next section, is an inevitable part of such a process.

2.2 Software Design Lifecycle

IS is thus composed of several facets. In this section we focus on the software system. The global process aiming to develop software facet of IS is called the **S**oftware **D**esign **L**ifecycle (SDL).

The most common system development activities are detailed in [Som01]. The common process includes four main stages:

1. **software specification**: consists in the definition of functionalities of the software and constraints on its operation;
2. **software development**: consists in the building of the software that meets specifications;
3. **software validation**: consists in the validation of the system by checking if the software makes what the customer needs;
4. **software evolution**: consists in the changes of the system to follow alterations in customers' needs.

Several decompositions of the four activities are possible and different development processes exist such as waterfall, spiral, prototyping, operational, transformational, the knowledge-based and domain-based models. As the purpose of the thesis is not focused on software engineering, we just illustrate the functioning of SDL by investigating the waterfall process.

Waterfall Development Process

The waterfall development process considers system development as successive stepwise transformations, from the description of the problem domain to a solution. Each step has to be successfully fulfilled before considering the next phase.However it is always possible to go back to the last step. The waterfall development process is depicted in Fig. 2.1. It contains five activities:

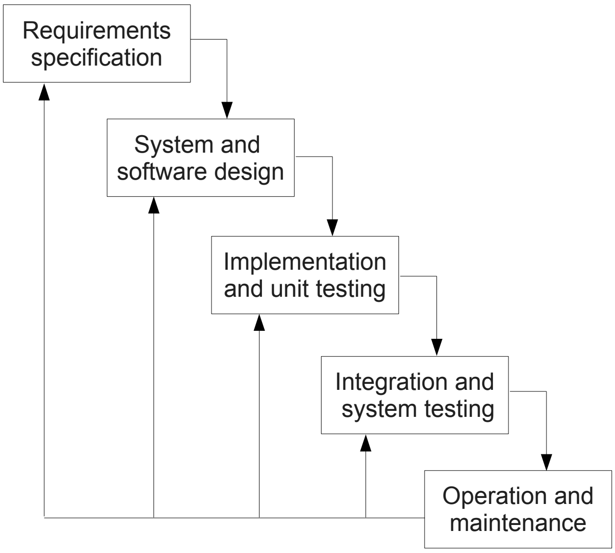

Requirements specification
System and software design
Implementation and unit testing
Integration and system testing
Operation and maintenance

Figure 2.1: Waterfall development process (adapted from [Som01])

1. **Requirements specification**: interactions with system users lead to the identification of system's services, constraints and goals. They are refined in order to be used as a specification;
2. **System and software design**: the system design process partitions the requirements to either hardware or software systems [Som01]. This activity draws the overall architecture of the system. "*Software design involves identifying and describing the fundamental software system abstractions and their relationships*" [Som01].

3. **Implementation and unit testing**: the software design is implemented as a set of programs. The unit testing involves verifying that each program corresponds to its specification;
4. **Integration and system testing**: all programs are gathered and a complete system test is run in order to ensure that the system meets its specification. The software system is then ready to be deployed in the customer's organisation;
5. **Operation and maintenance**: this activity focuses on the improvements of software system functionalities in order a) to remove bugs that were not discovered during unit and system testing and b) to align functionalities when customers' needs change.

2.3 Requirements Engineering

If building an IS is a quite difficult work, building an IS that really fulfills the purposes defined by all the stakeholders is an even harder one.

As said in [EN03], "*__R__equirements __E__ngineering (RE) is a* ***set of activities*** *concerned with* ***identifying and communicating*** *the* ***purpose*** *of a software-intensive system, and the* ***contexts*** *in which it will be used. Hence, RE acts as the bridge between the* ***real world needs*** *of users, customers, and other* ***constituencies*** *affected by a software system, and the* ***capabilities and opportunities*** *afforded by software intensive technologies*".

It emerges from this definition that:

1. RE is an **ongoing process** through the entire process, not only a stage or a phase;
2. **communication** is a really important part of the RE;
3. **understanding the purpose of the system** to build is at the heart of the success of the RE;
4. designers need to know **how and where** the system to build will be used;
5. on the one hand, requirements are partly about **what is needed**, and on the other hand about **what it is possible** to do;
6. **all stakeholders** need to be identified, not only users and customers.

In the context of IS development, security has to be considered as a requirement. It has to find its place in the global RE process.

2.4 Early Requirements

The present thesis focuses mainly on the Early Requirements phase.Early Requirements is concerned with the analysis of the operational environment where a software system will eventually function. For organizational software, this environment consists of stakeholders and their objectives, business processes, and inter-dependencies.

Early Requirements is often supported by modelling languages. The most frequent are:

- goal modelling languages such as KAOS [MMHD07] and i^* [BGG$^+$04];
- context and problem diagrams such as Problem Frames [Jac01]
- use cases [Jac92].

2.5 Summary

In this chapter, we defined foundamental concepts for the rest of the thesis. We first described the notions of **I**nformation **S**ystem (IS). We presented, in a general way, how the software component of an IS is elaborated and we focused on the waterfall development process. We also introduced the notion of Early Requirements because the thesis focuses essentially on this stage of the elaboration of an IS.

Chapter 3

Security and Risk Management

We define in this chapter the meaning of the terms *security* and ***R**isk **M**anagement* (RM).

3.1 Security Definition

Security is a word heavily used in all domains of our society. In this thesis, we focus on security in the context of IS. In [Pie07], security is described as *the **prevention** and the **detection** of **unauthorized actions** on **information**.* In this definition, the main element to be protected is the information. In [Fir03], Firesmith points out two main concepts making a system sure: a) the safety and b) the security of this system. The difference is that safety concerns accidental harm while security concerns malicious harm. Firesmith introduces the notion of *valuable asset* where Piessens focuses on information. In the sequel of the thesis, we will focus on the concept of **security**.

To summarize, we can say that security is about all harms, accidental or malicious, that threaten valuable assets of the IS. We will developed this definition in the sequel of the thesis.

3.2 Risk Management Definition

Risk **M**anagement (RM) is defined by ISO[1] in [ISO02] as the set of coordinated activities to direct and control an organisation with regard to risk. RM has three main objectives:

- to improve IS security;
- to justify the budget allocated to IS security;
- to prove the IS's credibility thanks to completed analyses.

This concept was born at the end of the fifties in the USA in financial domain in relationship to solving insurance questions as described in [Dub96]. Afterwards, this notion spread over other domains as environment, project management, marketing and computer security. RM tries to identify risks on the assets of an enterprise. The term **asset** must be taken in a broad sense: not only material things but also people, employees, knowledge of the enterprise. RM is an important process due to the fact that results regarding security reflect on the quality of the enterprise in general. Good results beget confidence of customers, partners, shareholders.

[1] International organisation for Standardization

Nowadays, we estimate that there are more than two hundred of **R**isk **M**anagement **M**ethods (RMM) (according to [MH06]). Such a multiplicity of methods leads to a great diversity of RM approaches. But we can define a set of basic common concepts to all RMMs.

3.3 Risk Management Process

We now describe the RM process at quite a high-level. The discussed process is usually found in one form or another in practical RMMs. On the other hand, the terminologies are not completely aligned within these different methods. The process can be broken down into six successive steps described below in Fig. 3.1. This description is inspired from [MH06].

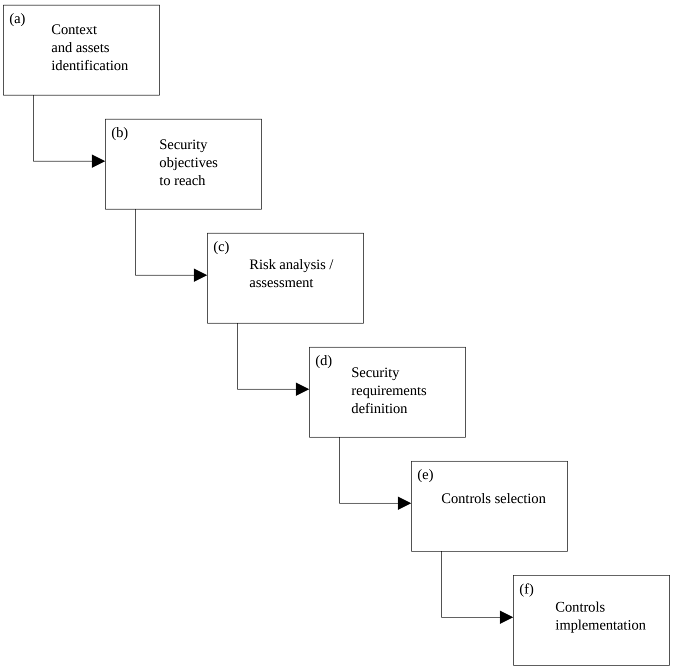

(a) Context and assets identification
(b) Security objectives to reach
(c) Risk analysis / assessment
(d) Security requirements definition
(e) Controls selection
(f) Controls implementation

Figure 3.1: Risk Management process (adapted from [MH06])

3.3.1 Context and Assets Identification

The first step (a) in Fig. 3.1 consists of the context and assets identification. During this phase, the analyst has to learn information about the organisation, focusing on the actors, manipulated data, assets, organisation's environment, organisation IS. After this learning, the first step continues by defining the system boundaries on which the risk management study will be realised. Once these boundaries have been identified, business assets – assets that constitute the organisation value – are identified. For each business asset, links are established with IS assets. These IS assets have to be identified and analysed, searching for risks from the technical and organisational points of view.

To illustrate this RM process, a case study described in [MH06] is analysed and, for each step of the RM process, a practical example is given.
Example: in the step (a), the context and assets identification teaches us that the case study is about an e-business website and this site has a database containing banking information about its customers.

3.3.2 Security Objectives to Reach

The second step (b) – security objectives to reach – intends to determine the security objectives in terms of availability, confidentiality and integrity, focusing on the business point of view. Other security criteria can be considered but the three discussed here are the most relevant in the practical risk management methods. Once the link between business assets and IS assets previously identified, this step defines security needs of the analysed system.
Example: by making links between business assets and IS assets, it appears that this database has relevant needs from the point of view of confidentiality due to the information's sensitivity.

3.3.3 Risk Analysis / Assessment

Risk analysis, also called risk assessment, is the third step of the RM process. Its purposes are the identification and the assessment of the three risk components: the threat, the exploited vulnerability (vulnerabilities) and the resulting impact(s). This leads to the risk assessment and to the estimation of its level in order to take the right measures. The assessment can be realised following two methods:

1. to make an audit of the system and of the different actors;
2. to consult a pre-defined knowledge database.

In theory, it is possible to quantify risk by statistical distributions on threats and on vulnerabilities and by estimating costs resulting from impacts. In practical situations, it is often impossible to give a precise value to estimate a risk. Risk levels are assessed and measured following a specific scale. Using such a scale, it is possible to situate the assessed risk value on a figure such as Fig. 3.2. Good practices use this division:

- **negligible area**: risks having low level of occurrence and low impact are inconsequential;
- **no existing risk**: risks having high level of occurrence and high impact are not acceptable. These risks need to be avoided. Otherwise, the organisation's activities have to be reconsidered if this situation occurs;
- **accepted risk**: risks having high level of occurrence but a low impact are accepted. Their cost is generally included in operational costs of the organisation;
- **risk to be transferred**: risks having low level of occurrence but a high impact have to be transferred. The responsibility is covered by an insurance or a third party;
- **risk to reduce**: other risks are considered individually and at the heart of the risk management. The aim is to mitigate these risks. That is illustrated on the following diagram Fig. 3.2 but bringing it near the axis origin.

This estimation makes possible to determine the most acceptable risk, before eliciting security requirements.
Example: risk analysis reveals several risk scenarios having an acceptable risk level for the customers' information in the database. One of them is the access to information by non-authorized actors through the network. In this case, data confidentiality could be broken. This risk needs to be treated by security controls.

3.3.4 Security Requirements Definition

The next step (d) is the one defining the security requirements that have to be taken into account to reduce the different risks. This step can lie on a knowledge database or can be exchanged by experts in system / in security. The complexity and the significance of the security requirements elicitation makes this process an incremental process. Starting with the definition of a high-level

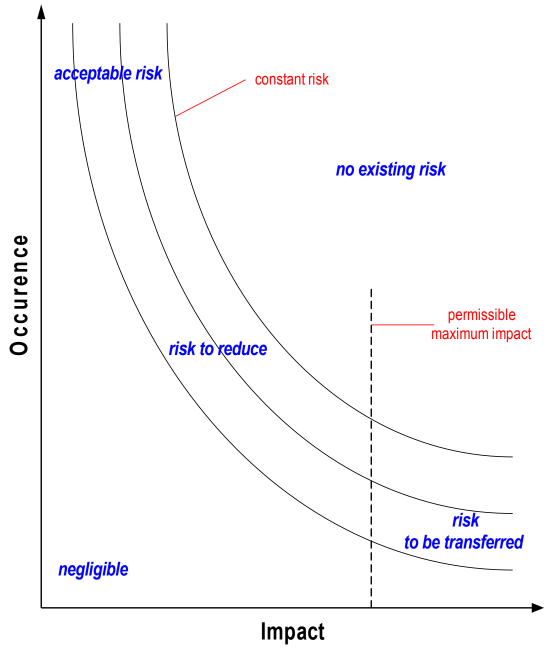


Figure 3.2: The different risk areas (adapted from [MH06])

risk (strategic level), searching for security requirements to counter it is a good way to start the risk management method. Indeed, this first high-level risk will be refined into more precise, practical risks of lower level (towards the operational level). These mitigation requirements are defined on the computerized system but also on its environment.
Example: the server, that hosts the customer database, needs to be protected from any kind of intrusions. At the same time, access control contrivance is placed on the database. A stronger remote authentication other than only a pair <login, password> is required. However, the cost and the simplicity of the use have to stay at an acceptable level.

3.3.5 Controls Selection

The last level of refinement is considered in the fifth step (e). This refinement results in control selection, also called countermeasures. Controls are the implementation of the lowest level of security requirements. The technical choices of the security solutions constrained by the established system are defined in this phase.
*Example: an **I**ntrusion **D**etection **S**ystem (IDS) is added to the current firewall. A authentication policy, stronger than <login,password> is chosen. This new policy combines a pair <login,password> with a code distributed to the customer during his registration. An authentication by card is considered to be too restricting.*

3.3.6 Controls Implementation

The last step (f) contains information about the control implementation in the IS. These controls can be evaluated and tested. The part of risks not covered by controls or introduced by the control implementation is called the residual risk.
Example: the computer service will put in place an IDS and the company that hosts the website will develop a new portal supporting the required authentication process.

3.4 Summary

We defined the notion of security and Risk Management (RM). We also analysed the RM process which is composed of six successive steps starting from context and assets identification and finishing with the implementation of controls. In this process, we discussed the different decisions that an enterprise can take when it is confronted to a risk. These decisions depend on the security strategy of the enterprise.

Chapter 4

Research Method for Designing ISSRM-compliant Languages

As the basic principles of Security Risk Management have been explained, we now describe the research method developed to address the question: *what are the concepts that should be present in a modelling language supporting ISSRM during the early stages of IS development?* The research method as described in [MHM07] consists of four steps. An overview is given in Fig. 4.1. It follows an iterative and incremental development pattern: during the work's progress, previous steps are frequently revisited to improve results of initial steps that are themselves used as inputs for further steps.

In order to provide stable and solid foundations for language support, a structured response to the previously enunciated question is formulated. A systematic analysis of ISSRM sources will produce the extraction of relevant concepts. Proceeding this way, concepts from existing security-oriented languages will be confronted with these extracted concepts and the coverage between them will be established. The resulting overlap will used in step 4. The research method gives a detailed description of this systematic analysis is described below and refers to Fig. 4.1.

4.1 Step 1: Concept Alignment

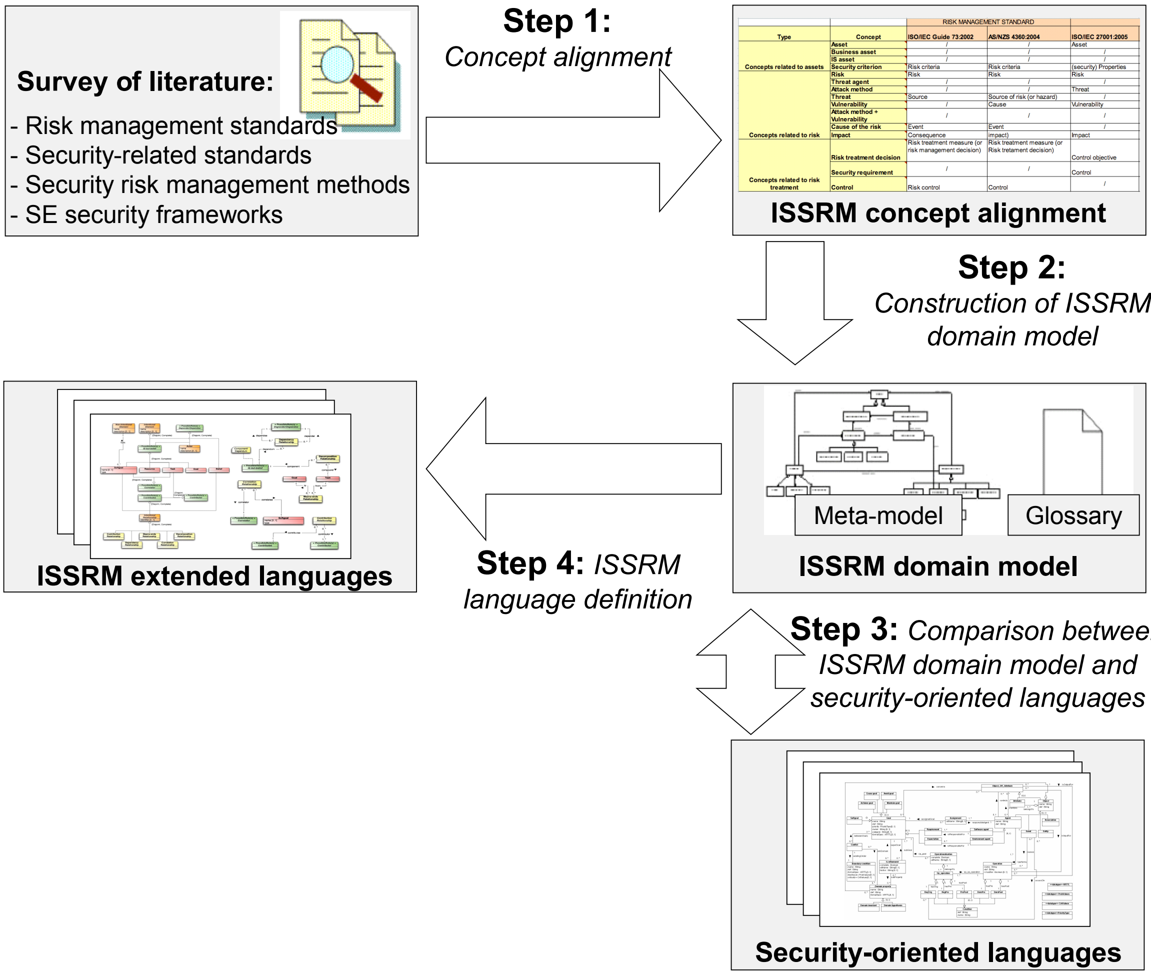
The first step is to collect information about the state of the art in ISSRM. The main objective of this step is the identification of key domain components and harmonization in terminology used in ISSRM sources.

Covered sources belong to one of these four categories: a) RM standards, b) security-related standards, c) security RM methods and d) **S**oftware **E**ngineering (SE) security frameworks. We elicit and detail them in Chapter 5.

Two documents are produced as outcome of this step (a detailed presentation of these documents will be given in the sequel):

- a grid that highlights the semantic similarities between terms as described in ISSRM sources;
- a glossary composed of definitions extracted from source documents.

Reading and extracting relevant information from source documents is the first part of the job made in this step, does not require a special manner to proceed. The second part, the harmonization of terminologies, could eventually be made following a quite formal method as UEML [A.L07, AG07]. Such quite formal methods have been used in [MPL07, MPL06]. Potential


Survey of literature:
- Risk management standards
- Security-related standards
- Security risk management methods
- SE security frameworks
Step 1:
Concept alignment
ISSRM concept alignment
Step 2:
Construction of ISSRM domain model
Meta-model
Glossary
ISSRM domain model
Step 3: Comparison between ISSRM domain model and security-oriented languages
Security-oriented languages
Step 4: ISSRM language definition
ISSRM extended languages

		RISK MANAGEMENT STANDARD		
Type	Concept	ISO/IEC Guide 73:2002	AS/NZS 4360:2004	ISO/IEC 27001:2005
Concepts related to assets	Asset	/	/	Asset
	Business asset	/	/	/
	IS asset	/	/	/
	Security criterion	Risk criteria	Risk criteria	(security) Properties
Concepts related to risk	Risk	Risk	Risk	Risk
	Threat agent	/	/	/
	Attack method	/	/	Threat
	Threat	Source	Source of risk (or hazard)	/
	Vulnerability	/	Cause	Vulnerability
	Attack method + Vulnerability	/	/	/
	Cause of the risk	Event	Event	/
	Impact	Consequence	impact)	Impact
Concepts related to risk treatment	Risk treatment decision	Risk treatment measure (or risk management decision)	Risk treatment measure (or Risk tretament decision)	Control objective
	Security requirement	/	/	Control
	Control	Risk control	Control	/

Figure 4.1: Global research method (adapted from [MHM07])

inputs are source documents, which are written in natural language. Such documents are not precisely defined, they can be understood differently by different people. So we need driven methods to analyse these documents. Moreover, the nature of these source documents raises problem. Indeed, they are scientific papers that might lack some concrete details about the provided content. The last and not the least, some concepts are described using different terminologies.

4.2 Step 2: Construction of ISSRM Domain Model

Using the grid as input, as a summary of key concepts used mainly in ISSRM sources, and the glossary as the semantic basis for key concepts, step 2 consists in the creation of a conceptual model of the ISSRM domain as a UML class diagram. This conceptual model is the foundation for the following step. Indeed, any RE security modelling language has to be designed to fit the conceptual domain model. The model is complemented with key components definitions glossary. These definitions are obtained by reusing, if needed improving, definitions discovered in source documents during step 1.

In order to produce a UML class diagram, the key concepts' elicitation process is reused to extract relevant relationships between these concepts. A new glossary [MHM07] is created in which relevant source documents excerpts are gathered in a way to prove the relationships' existence.

The new model follows the UML class diagram syntax. The choice of those which are, among key concepts obtained in step 1, relevant enough in the context of the new model being constructed, is made using common sense.

4.3 Step 3: Comparison between ISSRM Domain Model and Security-oriented Languages

"Is it necessary to reinvent the wheel?" This is one of the most relevant questions that anybody should ask himself before trying to build something new. The metaphor is suitable in the ISSRM domain context. This is why several security modelling languages as KAOS extended to security [vL04], Abuse frames [LNI$^+$03a], Misuse cases [LBD02], Abuse cases [MF99] and secure TROPOS [MJF06] are considered. Most of these languages are security extensions to existing modelling languages. For example, Misuse cases and Abuses cases are based on "regular" Use cases. The process unfolds like this: security modelling languages metamodels are investigated, looking for concepts in the ISSRM domain model which are supported and those which are missing. To help determining if a concept is present or not in the studied language, a UML class diagram is created. The class diagram contains key concepts that are supported by the language and relationships between them. At the end of this process, the class diagrams can be used to compare potential candidate languages for ISSRM modelling.

We use two methods to align concepts of security modelling languages with those of ISSRM domain model. The initial method was based on the definition of the different concepts and the metamodel of the languages in order to find synonyms in the ISSRM domain model. We used it to align security modelling languages such as KeS, Misuse cases, Abuse cases and Abuse frames (these languages are detailed in Chapter 7). However, it appeared that the precision of the outcomes of this alignment method was insufficient. One can find, in Appendix 3, the method and its outcomes. We then refined the method in order that it produces more accurate results. We described and used this method in Chapter 9.

4.4 Step 4: ISSRM Language Definition

Providing languages to support the ISSRM can be seen as the ultimate result of our research method. However, the introduction of a new language in the habits of an organisation which works with another language (or a combination of them or not at all) is a risky process. To ensure a smooth transition with respect to current organisational practices, the ISSRM domain model can be used as a framework outlining key concepts that should be taken into account when security concerns are tackled. So step 4 could result in ISSRM compliant versions of existing security modelling languages. Another aspect processed in Step 4 is the formal definition of syntax and semantic [HR04] of the ISSRM languages in order to support automated reasoning and avoid ambiguity. Softer characteristics as the appropriateness of graphical constructs and structuring mechanisms will also be taken into account [Moo06a, Moo06b].

Our contribution takes place at steps 2 and 3. In step 2, we elicited the ISSRM relationships (see Chapter 8) and, in step 3, we aligned several security modelling languages (described in Chapter 7 and discussed in Chapter 9).

4.5 Summary

In order to fulfill the objectives of this thesis, we performed parts of the research method that we detailed in this chapter. The research method can be decomposed into four successive steps. The performed parts belong to the step 2 – to achieve the first objective – and step 3 – to complete the other. The step 1 consists of the survey of security and risk related sources. The next chapter presents a state of the art of this literature.

Chapter 5

Analysis of Security and Risk related Sources

This chapter introduces the security and risk related sources that were used to elicit the relevant concepts that compose the ISSRM domain model. We begin by a state of the art of these sources. Then we present the documents produced as outcomes of the elicitation of the relevant concepts.

5.1 State of the Art of the Security and Risk related Sources

The research method, described in Chapter 4, begins with a literature survey, investigating four families of security and risk related sources. A description of each family and elements that constitute it is given, to present the literature covered by this research. Explanations about the four families are extracted from [MHM07] with some added information and presented in Table 5.1.

Family of sources	
1. Risk Management standards	ISO/IEC Guide 73 [ISO02] AS/NZS 4360 [AS/04]
2. IS/IT security standards	ISO/IEC 27001 [ISO05] ISO/IEC 13335-1 [ISO04] Common Criteria [Com05] NIST 800-27 Rev A / NIST 800-30 [SHF04]
3. Risk Management methods	EBIOS [DCS04] MEHARI [CLU04] OCTAVE [ADA01] CRAMM [Con03] CORAS [VML^{+}07]
4. Software engineering security frameworks	Haley *et al.* [HMLN06] Firesmith [Fir03]

Table 5.1: Four categories of security and risk related sources

5.1.1 Risk Management Standards

This first family gathers high-level references presenting general RM.

- **ISO/IEC Guide 73** [ISO02]: this guide defines the RM vocabulary and guidelines for use in ISO standards. It mainly focuses on terminology, which is of great interest with respect to our research method;
- **AS/NZS 4360** [AS/04]: this joint Australian/New-Zealand standard provides a generic guide for RM. The document proposes an overview of RM terminology and process.

5.1.2 (IS/IT) Security Standards

This category of documents often contains a section that concerns security-specific terminology. Sometimes some RM concepts are also mentioned.

- **ISO/IEC 27001** [ISO05]: the objective of this standard is to provide a model for establishing, implementing, operating, monitoring, reviewing, maintaining and improving an Information Security Management System (ISMS), that is the part of the overall management system of an organisation concerned by information security. The terminology related to an ISMS is provided in this reference;
- **ISO/IEC 13335-1** [ISO04]: this standard is the first of the ISO/IEC 13335 guidelines series that deals with the planning, management and implementation of IT security. This part is about concepts and models of IT security that may be applicable to different organisations;
- **Common Criteria** [Com05]: the Common Criteria (or ISO/IEC 15408) provides a common set of requirements for the security functions of IT products and systems and for assurance measures applied to them during a security evaluation. The first part entitled "Introduction and general model" is important with respect to our research scope;
- **NIST 800-27 Rev A** [SHF04] / **NIST 800-30 [7]** [SGF02]: within the series of publications proposed by the National Institute of Standards and Technology (NIST), the 800-series is about computer security. In this series of publications, NIST 800-27 and NIST 800-30 are the most relevant to the scope proposed this thesis. Terminology and concepts are provided by these standards and are compliant with each other.

5.1.3 Risk Management Methods

In 2004, a CLUSIF[1] study registered more than 200 security RM methods. In this work, only a representative subset of RM methods based on some recent conferences and studies, like the report "Inventory of risk assessment and risk management methods" [17] from ENISA have been retained. Most of the methods are supported by a software tool, but the focus in our research method stays only on the methodological part of each of them. As industry often works with "ready-to-use" products, these RM methods are well widespread in this environment and will be more retailed.

- **EBIOS** [DCS04]: the EBIOS method is developed and maintained by the DCSSI (Central Information Systems Security Division) in France. It was created in 1995, it is composed by five guidebooks (introduction, steps, risk assessment tools, risk treatment tools) and is supported by a software. The method aims to formalize security objectives suitable to (1) security needs revealed by audits and (2) the context of the organisation in which security objectives will be established. EBIOS decomposes the risk management into the three blocks previously described. As you can see on Fig. 5.1, the risk is constructed by taking into account the IS environment, its borders, the essential elements, information corresponding to business assets and, finally, entities which are IS assets. The second step of the EBIOS method follows a grid gathering the desired security criteria: confidentiality, integrity and availability. The risk tuned to the organisation is reinforced by taking into account vulnerabilities and threats judged as critical. Combining these two first steps, high-level security

[1] Club de la Sécurité de l'Information Français `http://www.clusif.asso.fr/en/clusif/present/`

requirements are defined, directly followed by low-level requirements named requirements. The last step leads to select right countermeasures suitable to organisation needs.

Due to the fact that EBIOS process does not treat the controls selection and controls implementation, EBIOS is considered “only” as a risk analysis method.

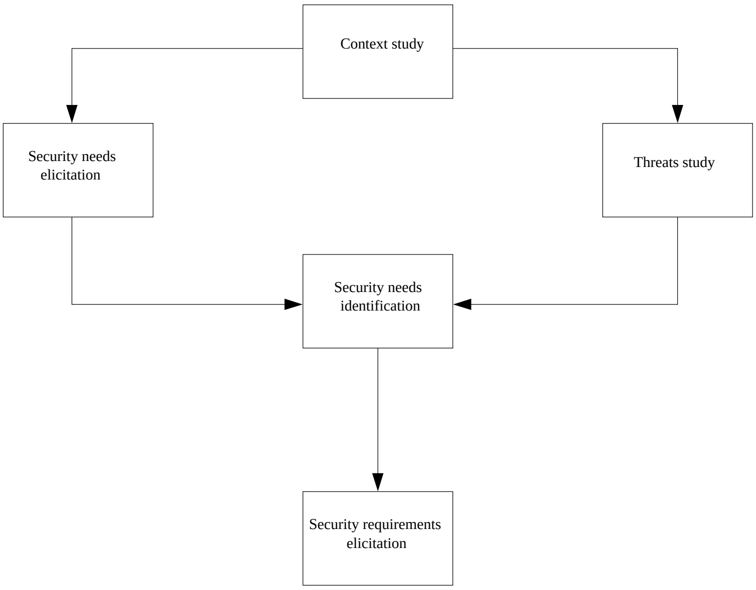

Context study
Security needs elicitation
Threats study
Security needs identification
Security requirements elicitation

Figure 5.1: Global EBIOS process (adapted from [MH06])

- **MEHARI** [CLU04]: MEHARI is a RM methodology developed by the CLUSIF and built on the top of two other RM methods: MARION [CLU98] and MELISA [Dir89] not maintained anymore. MEHARI remains one of the most used risk management methods. MEHARI can be viewed as a completed IS security toolbox, presented as a set of modules.

  The figure Fig. 5.2 displays a graphical of MEHARI process.

- **OCTAVE** [ADA01]: OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) is an approach to information security risk evaluations developed by SEI at the Carnegie Mellon University. The basis of this method is the capacity to realise internal audits. OCTAVE is suitable for large organisation. That is why OCTAVE-S has been proposed: to address risks in small structures. The two methods evaluate vulnerabilities and threats on operational assets. The OCTAVE (and OCTAVE-S) process is decomposed into three steps and represented in Fig. 5.3:

  1. **step 1** (organisational view): identification of relevant IT resources, their threats and the associated security requirements;
  2. **step 2** (technical view): identification of infrastructure vulnerabilities;
  3. **step 3** (protection and risk reduction design): security strategy development and its planning.

  As EBIOS, OCTAVE can be considered only as a risk analysis method.

- **CRAMM** [Con03]: CRAMM is a RM method from the UK originally developed by CCTA3 in 1985 and currently maintained by Insight Consulting.

- **CORAS** [VML$^+$07]: CORAS (Risk Assessment of Security Critical Systems) was a European project developing a tool-supported framework, exploiting methods for risk analysis and risk assessment of security critical systems. Theses modules permit:

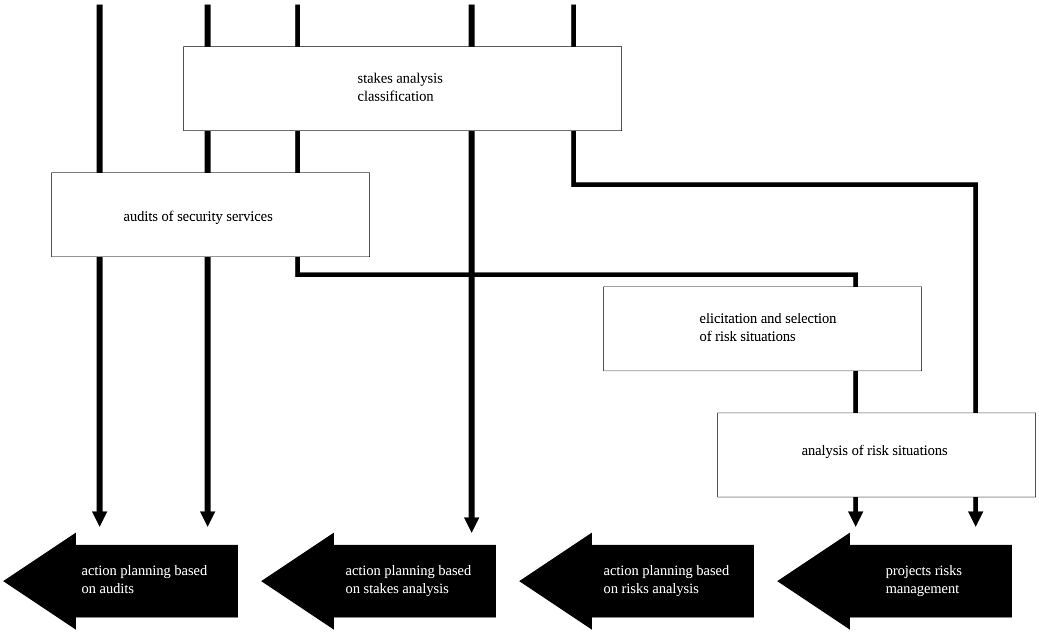


Figure 5.2: MEHARI process (adapted from [MH06])

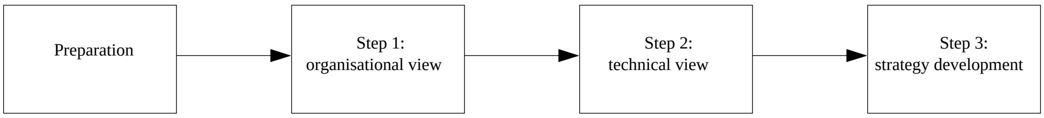


Figure 5.3: Main steps of OCTAVE process (adapted from [MH06])

– to analyse security stakes by describing kinds of dysfunctions and to classify assets depending on three security criteria (availability, integrity, confidentiality);

– to make audits on security services, testing their efficiency, their controls and to summarize vulnerabilities;

– to analyse risk circumstances (leading to evaluate their potentiality and their impacts) and risk attenuation factors, for, ultimately, assess risk level.

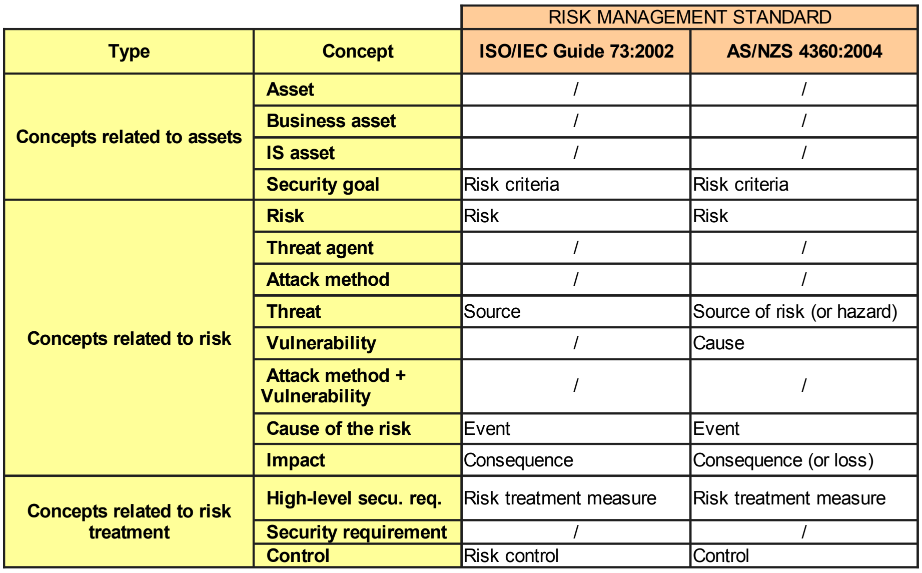
5.1.4 SE Security Frameworks

The publications in which **S**oftware **E**ngineering security frameworks are depicted, are extracted from SE and RE domain which are of great interest regarding the research scope. Those publications concern safety and security.

- **Haley et al.** [HMLN06]: this framework deals with security in RE;
- **Firesmith** [Fir03]: this framework is a set of related information models that provide a theoretical foundation underlying safety, security, and survivability engineering.

5.2 Grid of Concepts

To achieve the comparison of relevant concepts of the surveyed sources and to choose those which are necessary to construct ISSRM domain model, it is obviously important to have a clear view of supported concepts for each of these sources. A matrix was found to be an handy representation for this data. We present in Fig. 5.4 a excerpt of the complete grid of concepts.



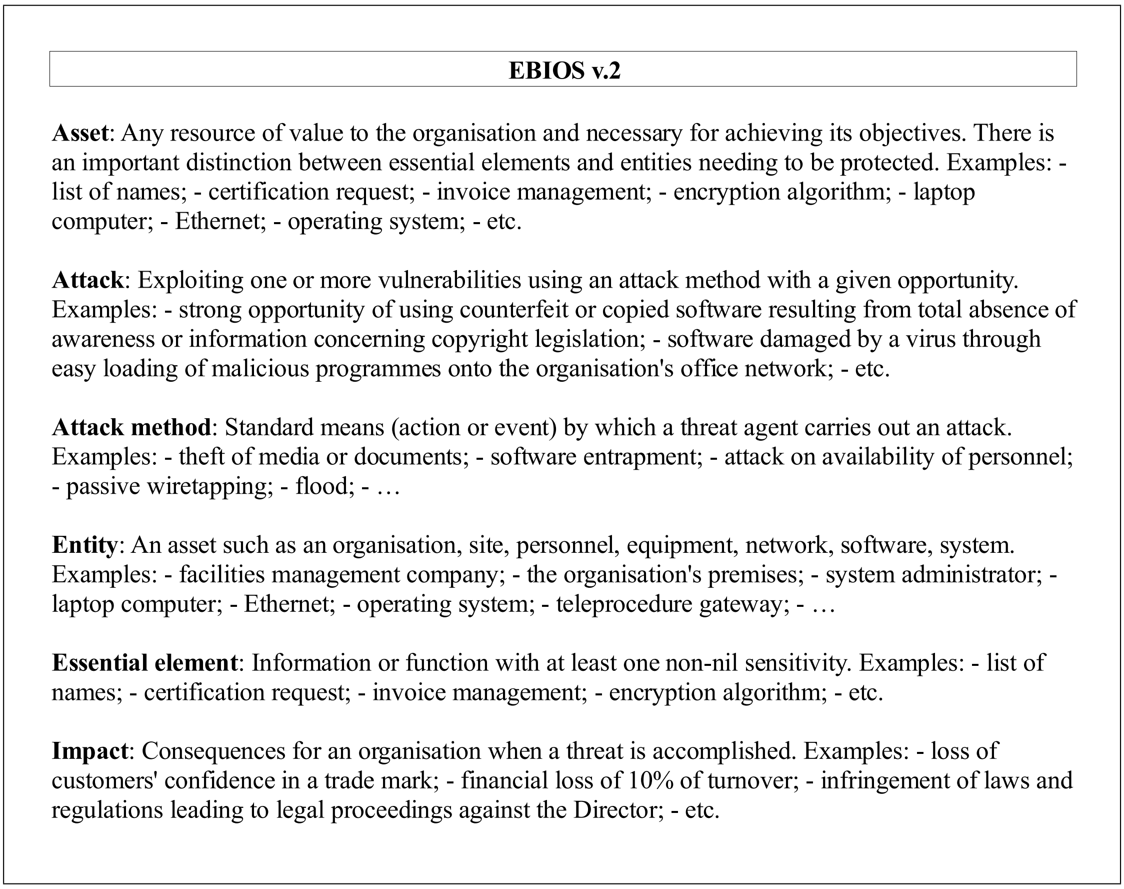
		RISK MANAGEMENT STANDARD	
Type	**Concept**	**ISO/IEC Guide 73:2002**	**AS/NZS 4360:2004**
Concepts related to assets	**Asset**	/	/
	Business asset	/	/
	IS asset	/	/
	Security goal	Risk criteria	Risk criteria
Concepts related to risk	**Risk**	Risk	Risk
	Threat agent	/	/
	Attack method	/	/
	Threat	Source	Source of risk (or hazard)
	Vulnerability	/	Cause
	Attack method + Vulnerability	/	/
	Cause of the risk	Event	Event
	Impact	Consequence	Consequence (or loss)
Concepts related to risk treatment	**High-level secu. req.**	Risk treatment measure	Risk treatment measure
	Security requirement	/	/
	Control	Risk control	Control

Figure 5.4: Excerpt of the grid of concepts

The grid is presented as a matrix in two dimensions. The first contains the name of the studied ISSRM sources, classified according to the family they belong to. The second suggests a label for the relevant concepts. The suggestion takes into account labels proposed in the security and risk related sources and the analysis performed in step 1 (Note that the excerpt of the grid only presents a small part of the complete grid. That is why the name of some concepts, such as security goal and impact, do not seems to the corresponding terms in the excerpt of the grid). The content of the grid is the outcome of the step 1. The concepts are grouped into three blocks. The first is composed of the concepts that are related to the notion of asset. The second group gathers concepts that composed the notion of risk. The last group concerns the risk treatment related concepts. At present, fourteen concepts have been selected which are defined in Chapter 6. They are gathered in the column named **Concept**. The complete grid can be found in Appendix 1.

5.3 Glossary of Key Concepts

While building the grid, a glossary of key concepts has to be elaborated to capture concept definitions extracted from ISSRM sources. The aim of the glossary is to condense in an unique document all the excerpts related to definitions on which step 2 is based. A difficulty is that a concept definition is often scattered in several parts of the document. Using a glossary centralizing the information facilitates the maintenance and navigation. We give an overview of the content of the glossary of concepts definitions in Fig. 5.5. This excerpt presents how EBIOS v2 described its fundamental terms. Note that all security and risk related sources do not give such definition for their concepts. As discussed previously, definition are sometimes scattered in several parts of the document. In this case, relevant sentences are gathered in the glossary and key terms are emphasized. One can see an example of this situation in the part of the glossary related to MEHARI. The full glossary can be found in [MHM07].



EBIOS v.2

Asset: Any resource of value to the organisation and necessary for achieving its objectives. There is an important distinction between essential elements and entities needing to be protected. Examples: - list of names; - certification request; - invoice management; - encryption algorithm; - laptop computer; - Ethernet; - operating system; - etc.

Attack: Exploiting one or more vulnerabilities using an attack method with a given opportunity. Examples: - strong opportunity of using counterfeit or copied software resulting from total absence of awareness or information concerning copyright legislation; - software damaged by a virus through easy loading of malicious programmes onto the organisation's office network; - etc.

Attack method: Standard means (action or event) by which a threat agent carries out an attack. Examples: - theft of media or documents; - software entrapment; - attack on availability of personnel; - passive wiretapping; - flood; - …

Entity: An asset such as an organisation, site, personnel, equipment, network, software, system. Examples: - facilities management company; - the organisation's premises; - system administrator; - laptop computer; - Ethernet; - operating system; - teleprocedure gateway; - …

Essential element: Information or function with at least one non-nil sensitivity. Examples: - list of names; - certification request; - invoice management; - encryption algorithm; - etc.

Impact: Consequences for an organisation when a threat is accomplished. Examples: - loss of customers' confidence in a trade mark; - financial loss of 10% of turnover; - infringement of laws and regulations leading to legal proceedings against the Director; - etc.

Figure 5.5: Excerpt of the glossary of concepts definitions

This unique glossary gathers information about all sources of the four families and it is divided into several sections. Each section corresponds to one the sources. In each section, concepts and their related definitions are classified in alphabetical order. When the definition of a concept contained in several sentences, keywords of the definition are written in bold font. Thus, understanding a concept is quite easier, just by looking for the right definition or for set of sentences and the words that are in bold font.

5.4 Summary

We started this chapter by a state of the art of security and risk related sources. This literature can be grouped into four categories: Risk Management standards, IS/IT security standards, Risk Management methods and Software engineering security frameworks. We briefly discussed the grid of concepts and the glossary that gathers concepts definitions of each source. Using these outcomes and relationships that we extracted – detailed in Chapter 8 –, the ISSRM domain was elaborated. We introduce it in the next chapter.

Chapter 6

ISSRM Domain Model

Based on relevant concepts obtained at the end of the step 2 of the research method, it is possible to design the ISSRM domain model. The ISSRM domain model is depicted in Fig. 6.1. As one can see, it can be decomposed into three groups of concepts:

- a) **assets-related concepts**: this block of concepts describe what are relevant assets of the system-to-be to protect. It also presents the criteria that guarantee the assets security. It gathers concepts that have a beige background colour;
- b) **risk-related concepts**: it presents how the risk itself is defined and what are principles that should be taken into account when eliciting potential risks. The background colour of these concepts is the orange;
- c) **risk-treatment decision**: it describes decisions, requirements and controls that should be chosen and implemented in order to mitigate risks. These concepts are painted in green.

In [MHM06], one can find explanations related to the selection of these concepts and the way their label were chosen.

6.1 Definition of ISSRM Domain Model Concepts

We give a definition for each concept of the ISSRM domain model. Concepts are gathered following the block they belong to as we explained before.

6.1.1 Asset-related Concepts

Asset: anything that has value to the organisation and necessary for achieving its objectives.
Examples: list of names; medical reimbursement process; operating system; laptop computer; Ethernet network; people encoding data; system administrator; air conditioning of server room.

Business asset: information or process inherent to the business of the organisation that has value to the organisation and necessary for achieving its objectives.
Examples: list of names; medical reimbursement process.

IS asset: a component or part of the IS that has value to the organisation and necessary for achieving its objectives and supporting business assets. An IS asset can be a component of the IT system, like hardware, software or network but also people or facilities playing a role in the IS and so in its security. Sometimes it is relevant for conducting a macroscopic analysis to define a system as an IS asset, composed of various IS assets belonging to other types described above.

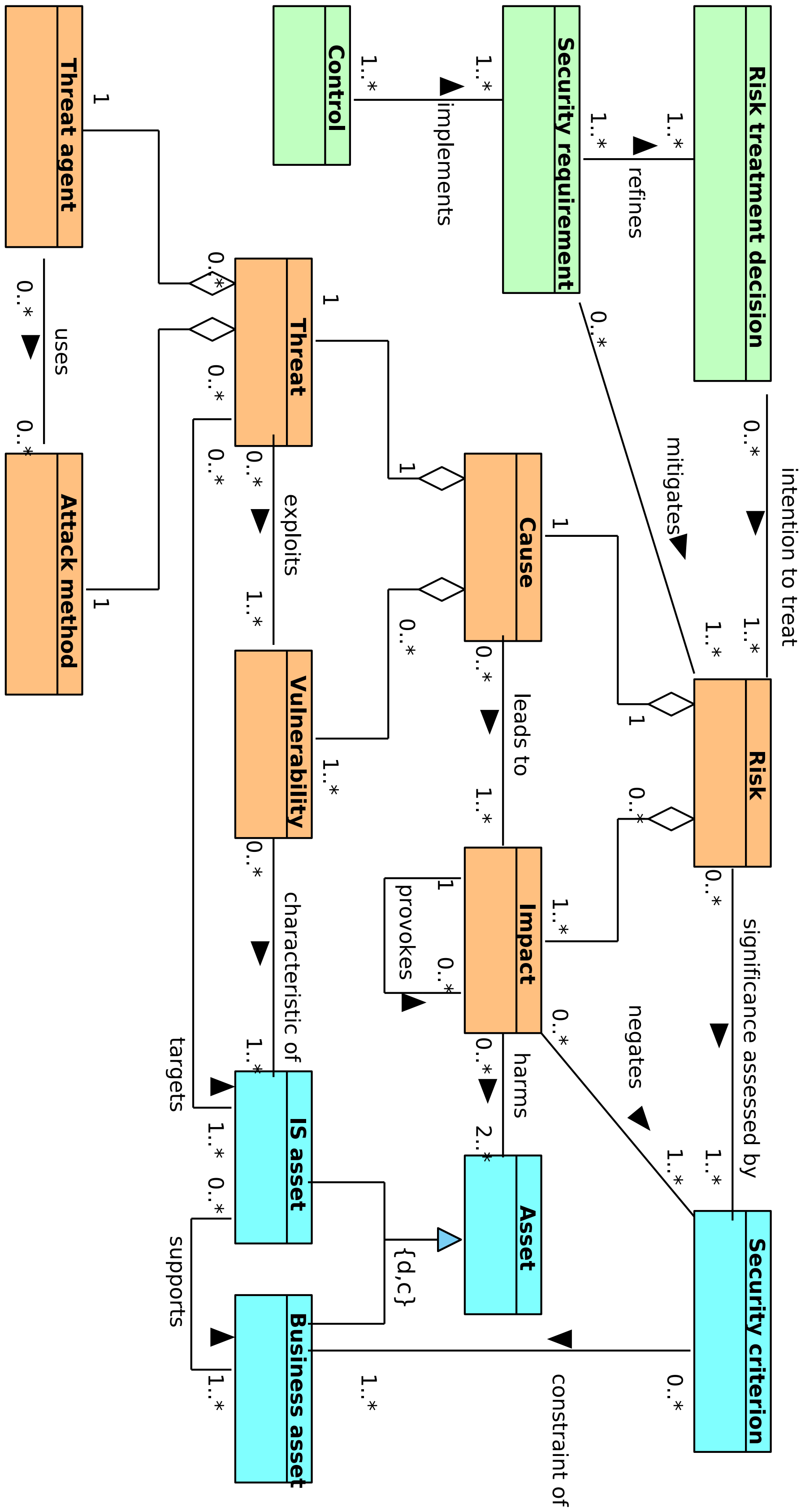


Figure 6.1: ISSRM domain model

Examples: operating system; laptop computer; Ethernet network; people encoding data; system administrator; air conditioning of server room.

Security criterion: – also called security properties – security criteria are properties or constraints on business assets characterising their security needs. Security criteria act as an indicator to assess significance of risk. Security criteria are most often confidentiality, integrity and availability, but sometimes, depending on the context, some other specific criteria might be added, like authenticity, non-repudiation or accountability. Security goals of an IS are defined by using security criteria on business assets.
Examples: confidentiality of information X; integrity of process Y.

6.1.2 Risk-related Concepts

Risk: a security risk is the combination of a threat with one or more vulnerabilities leading to a negative impact harming one or more of the assets. Threat and vulnerabilities are part of the cause of the risk and impact is the consequence of the risk.
Examples: a cracker using social engineering on a member of the organisation, because of weak awareness of the staff, leading to non-authorised access on personal computers and loss of confidentiality and integrity of sensitive information; a thief penetrating the organisation's building because of lack of physical access control, stealing documents containing sensitive information and so provoking loss of confidentiality.

Cause of the risk: the cause of a risk is the combination of a threat and one or more vulnerabilities.
Examples: Cracker using social engineering because of lack of awareness; a thief penetrating the organisation's building because of lack of physical access control.

Threat: potential attack or incident which, in combination with one or more vulnerabilities, targets one or more of the IS assets and that may lead to harm to assets. A threat is usually composed of a threat agent and an attack method. Note: We advocate that sometimes, a risk is more relevant to be described with a global threat, without refining into threat agent and attack method, like for a flood or a component failure.
Examples: Cracker using social engineering, flood, component failure.

Threat agent: a threat agent is an agent that can potentially cause harm to assets of the IS. A threat agent triggers a threat and is thus at the source of a risk. It can be characterised by its type (usually human or natural/environmental) and by the way in which it acts (accidental or deliberate). In the case of an accidental cause, it could also be characterised by exposure and available resources and in the case of a deliberate cause, it could also be characterised by expertise, available resources and motivation.
Examples: member of the personnel with little technical ability and time but possibly a strong motivation to carry out an attack; cracker with considerable technical ability, well equipped and strongly motivated by the money he could make; very wet climate for three months of the year; virus.

Attack method: standard means by which a threat agent carries out a threat.
Examples: system intrusion; theft of media or documents; social engineering; component failure; flood.

Vulnerability: characteristic of an IS asset or group of IS assets that can constitute a weakness or a flaw in terms of IS security. It could be accidentally or intentionally exploited by a threat.
Examples: weak awareness; lack of access control; lack of fire detection.

Impact: the impact is the potential negative consequence of a risk that may harm assets of a system or an organisation, when a threat (or the cause of a risk) is accomplished. The impact can be described at the level of IS asset (data destruction, failure of a component) or at the level of business assets, where it negates security criteria, like for example: loss of confidentiality, loss of integrity, unavailability. Impact can provoke chain reaction of impacts (or indirect impact), like for example a loss of confidentiality on sensitive information leads to a loss of customer confidence.
Examples: Password discovery (IS level); loss of confidentiality (business level).

6.1.3 Risk Treatment-related Concepts

Risk treatment decision: expression of the intention to treat identified risks. It satisfies a security need, expressed in generic and functional terms which can be refined through security requirements. Risk treatment decisions can include:

- avoiding risk (risk avoidance decision): decision not to be involved in, or to withdraw from a risk;
- reducing risk (risk reduction decision): to take action to lessen the probability, negative consequences, or both, associated with a risk;
- transferring risk (risk transfer decision): sharing with another party the burden of loss for a risk;
- retaining risk (risk retention decision): accepting the burden of loss from a risk.

Examples: not to allow to reach the service outside of the internal network ; the rooms must be protected against lightning ; to take insurance for covering the loss of service; accept that the service could be unavailable for 1 hour.

Security requirement: the security requirements are the refinement of the risk treatment decision into a set of security requirements to mitigate the risk. Each security requirement contributes to cover one or more risk treatment decision for the target IS. Note: Most often, security requirements are used to refine risk reduction decisions.
Examples: the system must generate the encryption keys in compliance with a specified encryption key generation algorithm and with the specified sizes of encryption keys in compliance with specified standards; a physical access control must be performed; rooms must be protected against the start and spread of fire.

Control: a control is designed to improve security, specified by a security requirement and implemented to comply with it. Security controls can be processes, policies, devices, practices or other actions or components of the IS and its organisation that act to reduce risk. Synonyms are countermeasures or safeguards.
Examples: firewall; backup procedure; building guard.

6.2 Relationships and Multiplicities

We now present the relationships and their multiplicities that are defined between ISSRM domain model concepts (depicted in Fig. 6.1). We make our discussion following the three same groups as we did for concepts definition.

6.2.1 Asset-related Relationships and Multiplicities

Assets can be specialised in two different sub-concepts: Business assets and IS assets. This specification is *disjoint* and *complete*. An IS asset can *support* one or more Business assets, but a Business asset can have no support in the IS. On the other hand, a Business asset can be supported by several IS assets. A Vulnerability is a *characteristic* of an IS asset or a group of IS assets. An IS asset can have from zero to several Vulnerabilities. A Threat *targets* one or more IS assets and an IS asset can be targeted by several Threats. The term "can" means that IS assets might not be threatened at all. Business assets can – absence is envisaged in the case where the Business asset has no support in the IS – be *constrained* by Security criteria. A Security criterion can constrain several different Business assets. Impacts *harm* Assets, at the levels of Business and IS assets. An Asset can be harmed by zero (if no impact is considered as relevant) or several impacts and an impacts harms at least an IS asset and a Business asset, but more than two are often harmed by an impact. At the level of Business assets, an Impact *negates* one or more Security criteria and a security criterion can be negated by zero (if no relevant impact is concerned by this Security criterion) or several impacts. One or several Security criteria can be taken into account to *assess the significance* of a risk. But a Security criterion can be concerned by none of the risks if there is no relevant Impact for the criterion.

6.2.2 Risk-related Relationships and Multiplicities

A Risk is *composed* of a Cause and one or more Impacts. A specific Impact can be the related to several Risks. A given Cause *leads* to, at least, one or several Impacts. An Impact can results from several Causes. Sometimes a relevant Impact can be caused by no relevant Causes of the risk and thus contained in none of the Risks. To illustrate this situation, we take the example of an organisation that wants to avoid the disclosure of user personal information. But as the organisation gathers no data about users, no realistic Cause of the risk can lead to this relevant Impact. Impacts can *provoke* some other (indirect) Impacts. For instance, an Impact at the IS level of unauthorized access to a database provokes confidential information disclosure at the business level, leading to the loss of customer confidence and judicial penalties. The Cause of a risk is *composed* of a Threat and one or more Vulnerabilities. A given Threat can only be related to a specific Cause of the risk. The Threat *exploits* one or more Vulnerabilities. A Vulnerability can be exploited by many different Threats and therefore related to many different Causes of the risk, or not be exploited by any of them, if no relevant cause is found. A Threat is *defined* in terms of a Threat agent who *uses* an Attack method. Each Threat agent and Attack method identified as relevant can be involved in several Threats or sometimes in none of them, if no important corresponding (respectively) Attack method/Threat agent is found. A given Threat agent can use from zero to several Attack methods and an Attack method can be used by zero or more Threat agents.

6.2.3 Risk Treatment-related Relationships and Multiplicities

A Risk treatment decision expresses the *intention to treat* one or more Risks. We state that each Risk has a corresponding Risk treatment decision, even if the decision is to accept the risk. Risk treatment decisions are generally *refined* into one or more Security requirements. Note that the Risk treatment decision of acceptance or avoidance does not produce Security requirements. Each Security requirement refines one Risk treatment decision. A Security requirement *mitigates* one or more Risks and a Risk might not be mitigated by any Security requirement if the Risk is accepted for example. It can also be mitigated by several Security requirements if they are necessary to reach an acceptable level of Risk. Finally, a Control *implements* one or more Security requirements and a Security requirement can be implemented by more than one Control.

6.3 Summary

The IS Security Risk Management (ISSRM) domain model defines all revelant concepts to elaborate secure system, from the analysis of a existing system to the introduction of countermeasures to potential risks. The concepts of the ISSRM domain model can be grouped into three blocks: the asset block, the risk block and the risk treatment blocks. We defined relationships existing between the concepts and their multiplicities. We now introduce the security modelling languages that we will align with ISSRM domain model in Chapter 9.

Chapter 7

Security Modelling Languages

In this chapter, we introduces security modelling languages that we will use to perform step 3 of the research method (see Chapter 4). We provide a detailed description of KeS and secure TROPOS as they are analysed in this thesis. Other security modelling languages – Misuse cases, Abuse cases and Abuse frames – are briefly presented. Their analysis can be found in Appendix 3.

7.1 KAOS

The first security modelling language that we introduce is **K**nowledge **A**cquisition in aut**O**mated **S**pecification (KAOS). This language in itself has no security concerns but as we will describe below, it has been extended to security and it is then called **KAOS e**xtend to **S**ecurity (KeS). Note that name of KeS has been chosen by ourselves.

7.1.1 Introduction to KAOS

We base our introduction to KAOS on [MMHD07]. The definition of KAOS can be found in [vL03, Let01]. KAOS is a goal-oriented modelling language. Its main purpose it to identify root goals – high level goals – and to refine them until achieving precise requirements. Agents are then elicited and they are designated as responsible of the requirements. The successive refinements of goals until requirements modelled alternative solutions of the system-to-be that will be discussed in order to select the most satisfactory solution.

The KAOS approach consists of a modelling language, a method, and a software environment. We focus on the modelling language. A KAOS model consists of four types of diagrams (also called **models** in KAOS):

1. **the goal model**: identifies the root goals of the system-to-be and their refinements until obtaining requirements that are assigned to agents;

2. **the object model**: identifies the objects of the system-to-be that are used during operationalization. The object model is generally built in parallel with the goal model;

3. **the operation model**: identifies the operations required to fulfill a requirement elicited in the goal model. Operations have inputs and outputs and are performed by agents;

4. **the agent models**: identifies the agents of the system-to-be.

In the sequel of the thesis, we will mainly focus on the goal and the object models.

A KAOS metamodel is depicted in Fig. 7.1. We do not describe the metamodel in detail but we explain its main concepts.

- **Goal**: a "*Goal is a prescriptive assertion that captures an objective that the system-to-be should meet*" [MMHD07]. It can be categorized as one of these four patterns: *maintain*, *cease*, *achieve* and *avoid* (see Fig. 7.1). Each pattern defines a temporal behaviour categories [Let01];
- **G-refinement**: a Goal can be refined using a G-refinement which decompose the Goal into a set of sub-goals. This decomposition is an AND decomposition. Sub-goals, with possibly domain properties, contribute to the satisfaction of the decomposed Goal. The possible alternatives in the design of the system-to-be can be expressed by modelling alternative G-refinement for the same Goal;
- **Object**: "*an Object is a (kind of) thing of interest in the system*" [MMHD07]. Instances of an Object are recognizable and they can evolve from state to state.
- **Attribute**: Objects have Attributes that characterize the states of the system-to-be. Hence, a Goal refers to Objects and/or their Attributes [vL03];
- **Agent**: an Agent plays a role towards the satisfaction of a Goal by controlling the behaviour of the Objects concerned by this Goal;
- **Requirement**: a Goal refined until it is possible to find a software Agent who is responsible for it is called a Requirement. Requirements can be operationalized in the Operations model;
- **Expectation**: an Expectation is a terminal refinement of a Goal and it is assigned to an Agent of the environment, whereas a Requirement is assigned to a Software Agent;
- **Operation**: an Operation is an input-output relationship over Objects; Operations are determined textually by *domain* (DomPre and DomPost) and *required* (RegPre, ReqTrig and ReqPost) conditions;
- **Domain property**: a Domain property is a property that naturally holds in the environment.

The syntax of KAOS and KeS – as we will see in the sequel – is depicted in Fig. 7.2. Goal G1 refined with the G-refinement into sub-goal 1A and sub-goal 1B is an AND G-refinement. The Goal G2 is refined with two alternative G-refinements into sub-goal 1 and sub-goal 2.

7.1.2 Introduction to KeS

KeS is introduced in [vL04] and deals with security concerns by using security Goals that have a concern relationship with assets. The metamodel of KeS is exactly the same as the one depicted in Fig. 7.1. The only difference is the point of view of the analyst when eliciting the Goals from the malicious Agent. Thus, instead of having a Goal model, an Object model, an Agent model and a Operation model, these models are prefixed by the term "Anti-". New concepts (but represented with the same metamodel as KAOS) are:

- **Asset**: Assets of the system-to-be are modelled as Objects and are constrained by security criteria. These security criteria need to be protected against threats. Security criteria are represented as Objects attributes concerned by Anti-goal (Anti-goal is defined below);
- **Threat**: a Threat is a attacker's Goal also called Anti-goal;
- **Attacker**: an Attacker is represented as a malicious Agent in the environment.

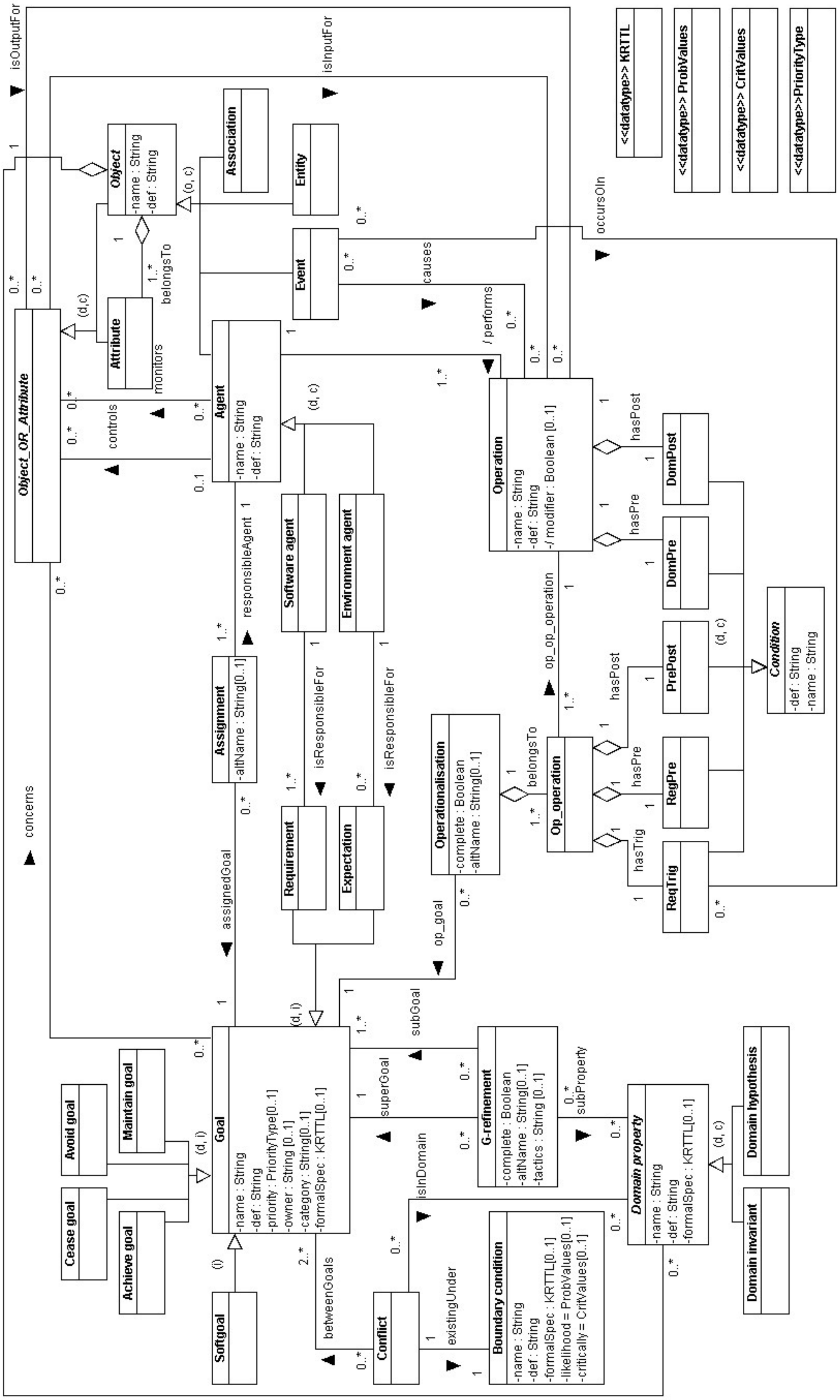


Figure 7.1: KAOS metamodel (adapted from [MMHD07])

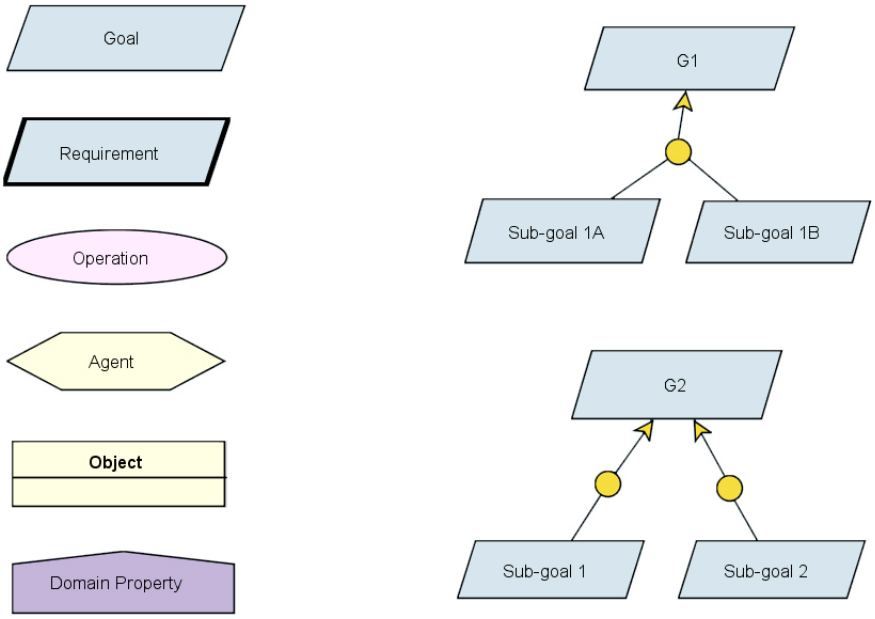


Figure 7.2: Graphical representation of KAOS and KeS concepts

The Anti-goal model is used to capture all possible Attackers, their Goals, the software Vulnerabilities and the Attack methods of the Attackers. When this Anti-goal model has been built, the analyst of the system-to-be has to consider countermeasures to the found Anti-goals and Vulnerabilities. As described in [MMHD07], countermeasures might include security treatment decisions, such as goal substitution, agent substitution, goal weakening, goal restoration, anti-goal mitigation, anti-goal prevention, vulnerability protection and vulnerability avoidance. Once the suitable decision taken in order to counter a specific threat, new security Goals are integrated in the latter models. As you can see, the KeS method is a iterative process as each time countermeasures are introduced in the design of the system-to-be, a new phase of assessment of the potential threat introduced by these new countermeasures has to be completed.

7.1.3 Illustration of KAOS and KeS in the Banking Services Example

To illustrate concepts of KAOS and KeS, we present the Banking Service example. Our aim is not to develop a complete case study at this point of the thesis. One can be found in Chapter 10, where we describe the Meeting scheduler system. The considered example is based on [vL04].

We depict in Fig. 7.3 an excerpt of the goal model of the Banking Services example. We start with the Goal `MoneyTakenFromTheBank` that is refined into a Domain property `ThereIsSomeMoneyInAccount`, a Requirement `Achieve [MoneyTakenOnlyByBankCustomer]` under the responsibility of the Agent `BankCustomer`. The last element of the refinement is the Goal `Avoid [MoneyStolenFromBankAccounts]`. As the labels of these concepts are self-explanatory, we do not explain them in further details. The Goal `Avoid [MoneyStolenFromBankAccounts]` itself can be refined into two alternative sub-goals `Avoid [PaymentMediumKnownByThief]` and `Avoid [MoneyTakenFromAttackedAccounts]`. We focus on the fulfillment of the Goal `Avoid [PaymentMediumKnownByThief]` that we decompose into three Requirements `Avoid [ThiefKnowsWhichBank]`, `Avoid [ThiefKnowsAccountStructure]` and `Avoid [AccountNumberAndPinKnownByThief]`. We choose, for sake of shortness, not to refine the Goal `Avoid [MoneyTakenFromAttackedAccounts]`.

On the Operation model of the Requirement `Achieve [MoneyTakenOnlyByBankCustomer]`

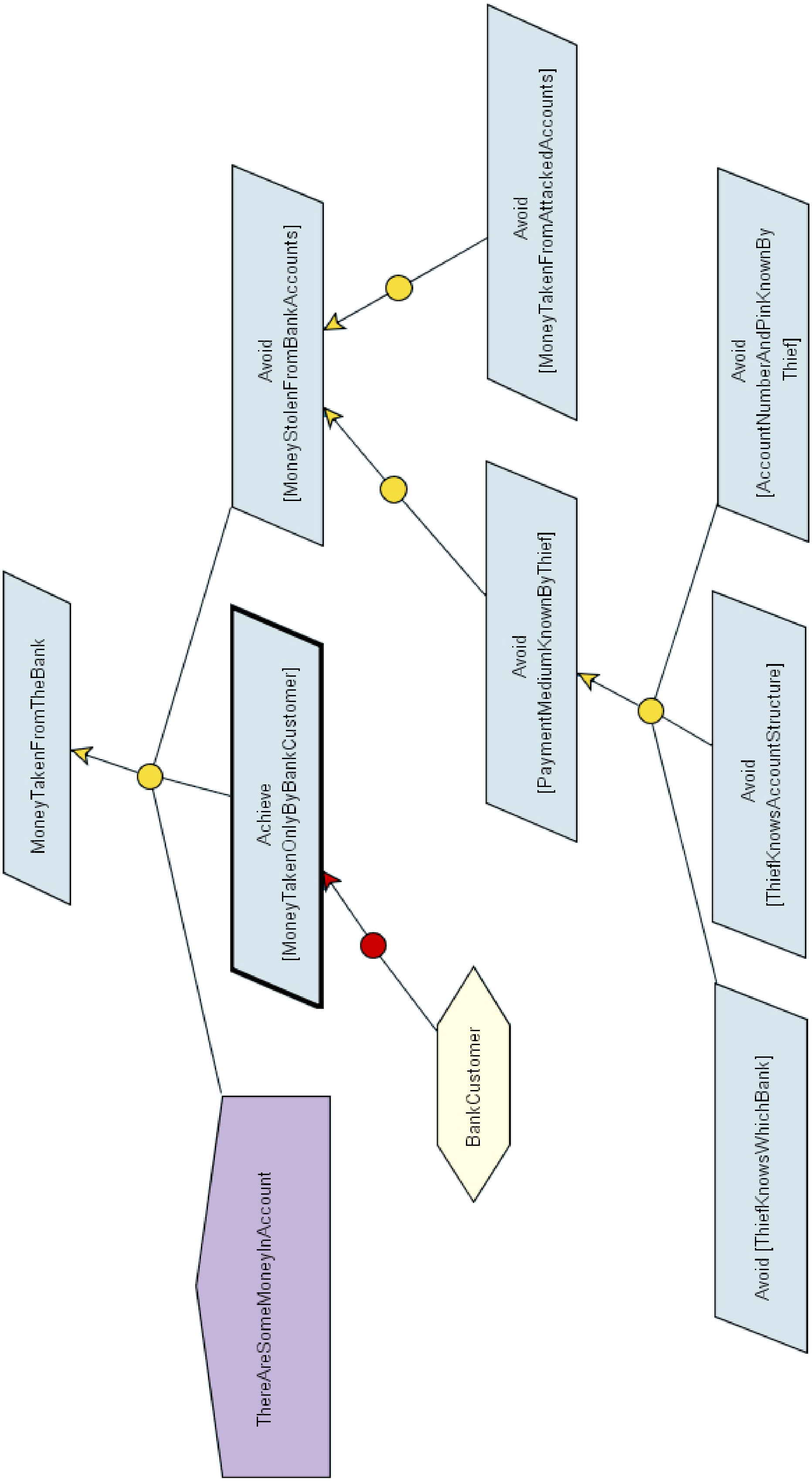

MoneyTakenFromTheBank
Avoid [MoneyStolenFromBankAccounts]
Avoid [MoneyTakenFromAttackedAccounts]
Achieve [MoneyTakenOnlyByBankCustomer]
Avoid [PaymentMediumKnownByThief]
Avoid [AccountNumberAndPinKnownBy Thief]
Avoid [ThiefKnowsAccountStructure]
Avoid [ThiefKnowsWhichBank]
BankCustomer
ThereAreSomeMoneyInAccount

Figure 7.3: Excerpt of the Goal model of the Banking Services (adapted from [MMHD07])

(depicted in Fig. 7.4), this Requirement is placed under the responsibility of the Agent BankCustomer and it is operationalized into three Operations: EnterPin, TakeMoney and EnterAccountNumber. The latter Operation uses the DatabaseAccountNumbers to perform its internal process.

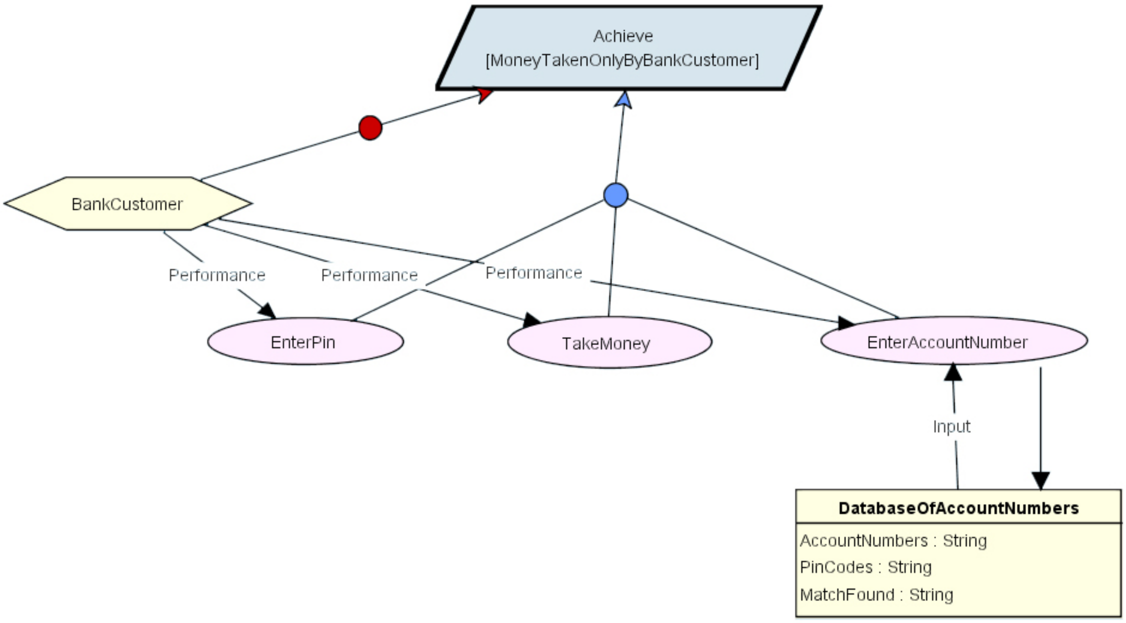


Figure 7.4: Operation model of the operationalization of the Requirement Achieve [MoneyTakenOnlyByBankCustomer] (adapted from [MMHD07])

Once the system-to-be is modelled, we can investigate the attacker's point of view (see Fig. 7.5). This is done by building the anti-goal model that negates one of the Goals of the goal model. The negated Goal is transformed into the Anti-goal Achieve [AccountNumberAndPinKnownByThief]. It is refined into two alternative sub-goals Achieve [PinKnownAndMatchingAccountFound] and Achieve [AccountKnownAndMatchingPinFound]. The second alternative is not further refined. The first leads by successive refinements to three Requirements Achieve [PinKnown], Achieve [AccountCheckedForPinMatch] and Achieve [CheckIteratedOnOtherAccountsIfNoMatch]. These three Requirements are assigned to the Agent called Attacker.

The attack method of the attacker is described in Fig. 7.6. The Requirement Achieve [CheckIteratedOnOtherAccountsIfNoMatch] is operationalized into two Operations InputNextAccountNumber and CheckIfAccountNumberMatch. The second Operation needs the DatabaseOfAccountNumbers as inputs and outputs of its internal process.

7.2 Secure TROPOS

The second security modelling language that we analyse in the next chapter is secure TROPOS.

7.2.1 Introduction to TROPOS

TROPOS, whose name is derived from two Greek words meaning respectively "*way of doing things*" and "*turn*" or "*change*", is a software development method founded on concepts of *actor* (which can be *agents*, *positions* or *roles*), *goal*, *plan* (also called *task*), *resource*, *capability*, *belief*

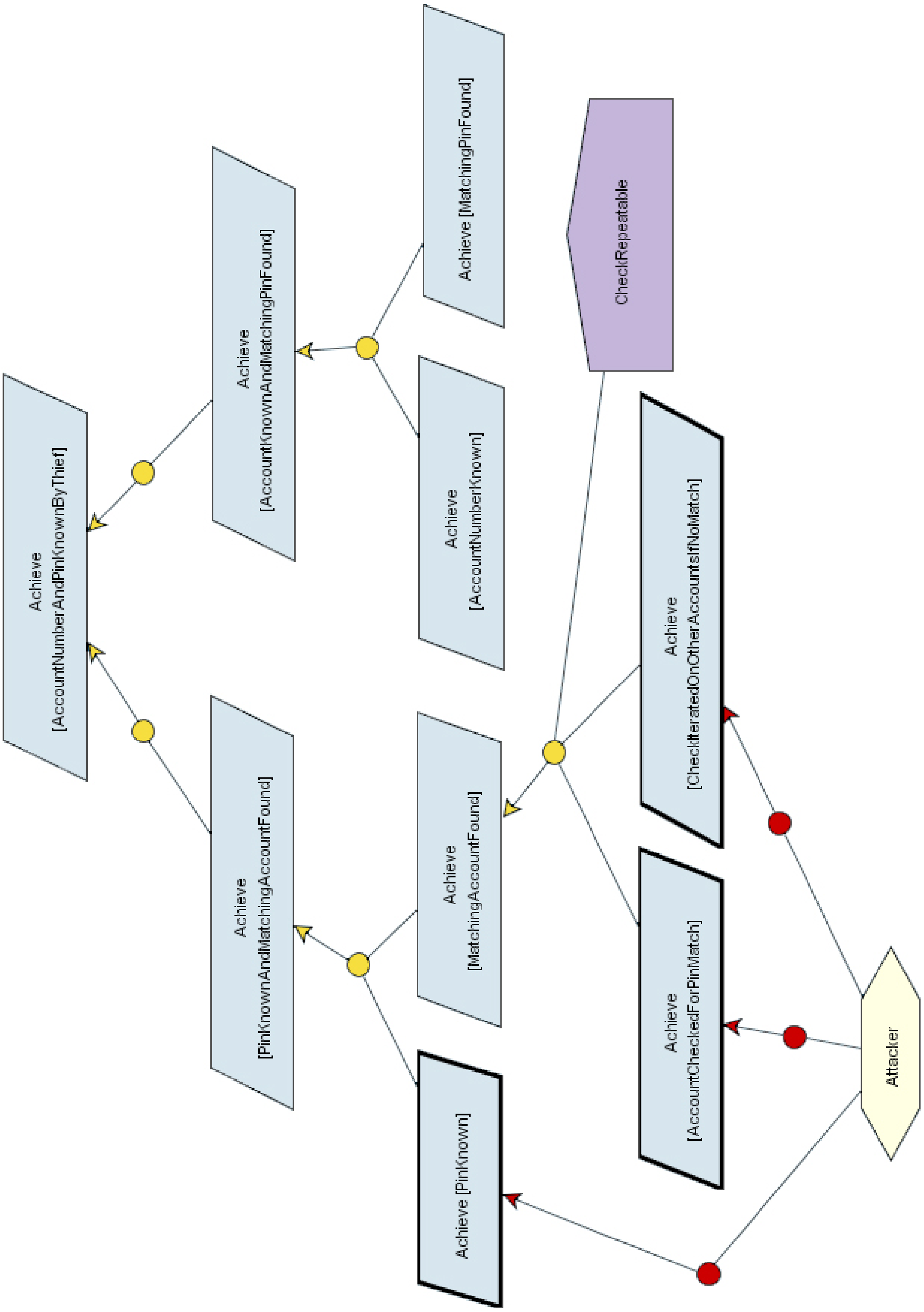

Achieve [AccountNumberAndPinKnownByThief]
Achieve [AccountKnownAndMatchingPinFound]
Achieve [MatchingPinFound]
Achieve [AccountNumberKnown]
CheckRepeatable
Achieve [PinKnownAndMatchingAccountFound]
Achieve [MatchingAccountFound]
Achieve [CheckIteratedOnOtherAccountsIfNoMatch]
Achieve [AccountCheckedForPinMatch]
Achieve [PinKnown]
Attacker

Figure 7.5: Anti-goal model of an attacker against the Banking Services (adapted from [MMHD07])

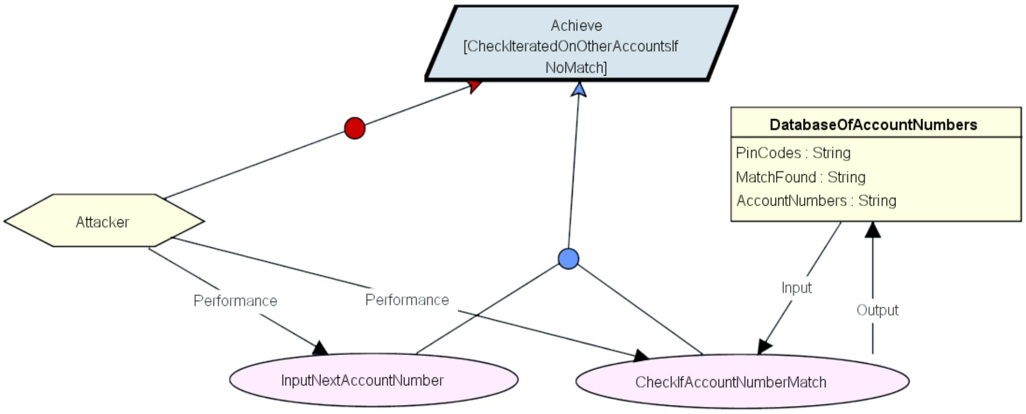


Figure 7.6: Operationalization of the Attacker's Requirement `Achieve [CheckIteratedOnOtherAccountsIfNoMatch]` (adapted from [MMHD07])

and *dependency*. These concepts, that are used to model requirements, have been introduced in i^* proposed by Eric Yu in [Yu95b]. However, the TROPOS methodology is intended to cover all analysis and design activities in the software development process, from early requirements down to the solution implementation. The main idea is to build a *model* of the system-to-be [BGG$^+$04] and its environments and to improve it incrementally by refinements and extensions. Hence, TROPOS provides a common interface to software development activities and a basis for evolution and documentation of the solution.

More precisely, the TROPOS methodology manipulates i^* concepts that we define just below. In TROPOS, several types of models are created but, as we focus on early requirements, we take into account only a) the **Strategic Dependency Model** and b) the **Strategic Rationale Model**.

The Strategic Dependency Model (SDM) is used to describe the network of relationships between actors of the system-to-be. These relationships capture the **intentional dependencies** between actors, and not the actors' internal goals. The concepts involved in this model are:

- **actor**: *entity that has strategic goal and intentionality within the system or the organisation setting. An actor represents a physical, a social or a software agent as well as a role or position [BGG$^+$04].* A *role* is defined as the behaviour of an actor in a given context. A *position* gathers a set of roles generally played by a specific agent. Following TROPOS definition, we say that an *agent* occupies a *position* and that a *position* covers a *role*;
- **goal**: represents one of the actors' strategic interests that is clearly defined, clearly bounded; When it is fulfilled, a goal is said to be **satisfied**;
- **softgoal**: represents one of the actors' strategic interests that that has no clear-cut definition and/or criteria for deciding whether it is satisfied or not [BGG$^+$04]. When it is achieved, a softgoal is said to be **satisficed**. This distinction of terminology with the fulfillment of a goal permits to distinguish between the goals and the softgoals;
- **plan**: also called *task* represents a way of doing things at a relatively abstract level;
- **resource**: represents an informational or physical entity;
- **capability**: is the term denoting *the ability of an actor of defining, choosing and executing a task for the fulfillment of a goal, given certain world conditions and in presence of a specific event [BGG$^+$04].* The elicitation of actors' capabilities is done at the end of the TROPOS process (described in the sequel of the thesis);

- **belief**: is the actor's knowledge of the world [BGG$^+$04];

- **dependency**: a dependency between two actors indicates that one actor (the dependee) depends for some reason on another one (the depender) in order to achieve a goal, to execute a plan, to deliver a resource. The object of the dependency is called the dependum. A dependum can be a *goal*, a *softgoal*, a *plan* or a *resource*.

Fig. 7.7 depicts the graphical syntax of the TROPOS concepts used in SDM. Fig. 7.8 presents an example of SDM where an actor – the Patient – depends on another actor – the eSAP system (a detail explanation of eSAP system is given in the sequel) – to achieve the goal `InformationProvided` and to obtained the resource `PatientPersonalInformation`. The goal and the resource are dependums of their respective dependency relationships. The direction of the **D** tag on dependency relationships indicates which actor depends on the other to achieve the dependum. In our example, the eSAP system depends on the Patient to fulfill the goal `InformationProvided` and to obtain the resource `PatientPersonalInformation`.

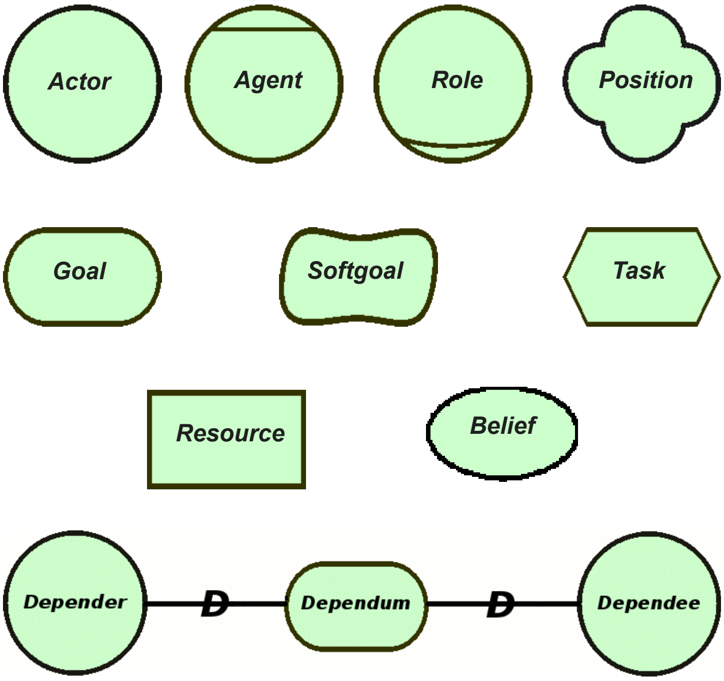


Figure 7.7: Graphical representation of TROPOS concepts used in SDM

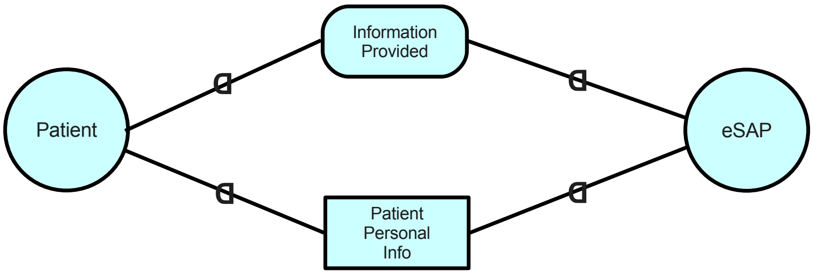


Figure 7.8: Example of SDM

The Strategic Rationale Model (SRM) While intentional dependencies among actors are modelled in SDM, internal goals of actors are captured in the SRM. The SRM offers a deeper understanding of a strategic actors' reasoning by *determining through means-ends analysis how goals (and softgoals) can be fulfilled through the contributions of others actors* [CKM02]. The SRM uses the concepts of *goal*, *softgoal*, *plan* and *resource* (previously described) and three kinds of relationships: a) the *means-ends* relationship, b) the *decomposition* relationship and c) the *contribution* relationship.

- a) **the *means-ends* relationship** links a *plan* with a *goal* it has the means to achieve. For convenience, TROPOS authorizes also *softgoals* and *resources* to be means to achieve a *goal*;
- b) **the *decomposition* relationship** further refines a *goal* by constructing a finer goals structure under an AND or an OR *decomposition*. Only a *plan* can be decomposed into *goals*, *softgoals*, *resources* and/or *plans*. TROPOS also allows the decomposition of a *goal* (see elaboration of secure TROPOS metamodel). An AND *decomposition* requires that all the sub-goals are achieved in order to fulfill the decomposed *goal*. In OR decomposition, sub-goals represent alternative ways to fulfill the decomposed *goal*. Hence only one sub-goal is needed to achieve the decomposed *goal*;
- c) **the *contribution* relationship** permits to *specify the goals or plans or resources that can contribute positively or negatively to its achievement* [BPSMa].

Fig. 7.9 depicts the graphical syntax of TROPOS relationships used in SRM. SRM concepts have already been defined previously. Fig. 7.10 gives an overview of a SRM. It represents the internal structure of the eSAP actor. We have chosen to refine the goal `InformationProvided`. This goal was introduced in Fig. 7.8 as a dependum of the dependency relationship. In order to achieve this goal, the eSAP system uses the plan `CollectInformation`. It is a means to fulfill the root goal and it is modelled with a means-ends relationship. To perform the plan, the eSAP has to `ManageCarePlan` of the Patients and it needs the `PatientPersonalInformation`. The sub-plan and the resource are required in order to fully performed the plan `CollectInformation`.

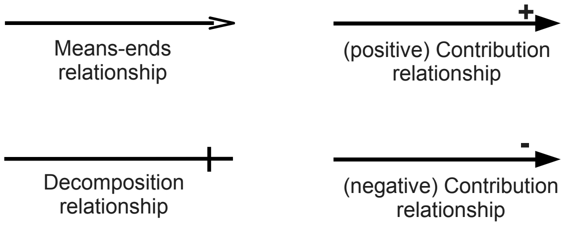

Means-ends relationship
+
(positive) Contribution relationship
Decomposition relationship
-
(negative) Contribution relationship

Figure 7.9: Graphical representation of TROPOS relationships used in SRM

The method consists of on four development phases (briefly described below):

1. **early requirements analysis** consists of identifying and analyzing the different stakeholders, in the environment of the system-to-be, and their intentions. Stakeholders are represented by actors depending on one another for achieving goals, for performing plans and for furnishing resources. Their intentions are modeled as goals that can be refined into finer goals;
2. **late requirements analysis** aims to define the system-to-be within its operating environment, considering its relevant functions and qualities. The system-to-be is represented as an actor linked to other actors of the organisation via dependency relationships. These dependencies are the functional and non-functional requirements of the system;
3. **architectural design** focuses on the elicitation of sub-systems (considered as actors) connected through data and control flows (considered as dependencies). This design leads to the system's global architecture;

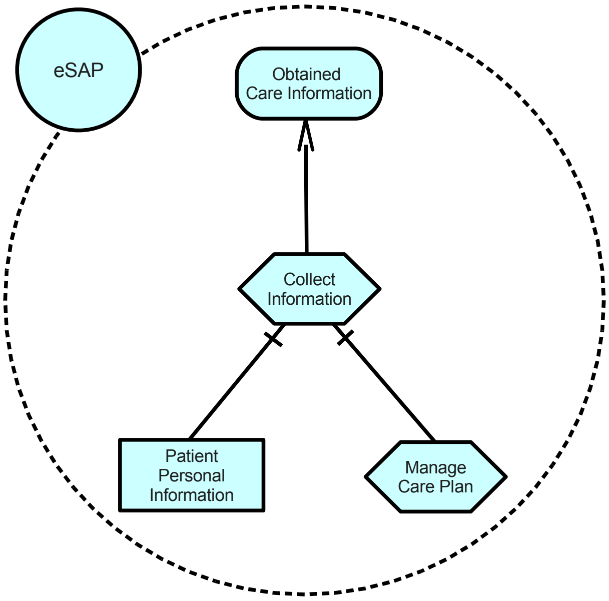


Figure 7.10: Example of SRM

4. **detailed design** goes into details of the specification of the *agents' micro level* (see [Yu95a]). This phase takes into account the development platform and the programming language used to elaborated the system.

However, TROPOS method has no specific concepts related to security. Two main extensions have been proposed, both named *secure TROPOS*. One is developed by H. Mouratidis *et al.* [MG04] at the University of East London. Their method adds concepts related to **security**. The second *secure TROPOS* [BMMZ06], proposed by N. Zannone from University of Trento, focuses on the concept of **trust**. At this point of our work, we do not focus on the trust concept, so this second extension of TROPOS will not be investigated.

7.2.2 Secure TROPOS : Security Extension

Secure TROPOS designed by H. Mouratidis *et al.* is a security extension of TROPOS that "*introduces security related concepts to the TROPOS methodology*" [MJF06]. This extension allows developers to consider security issues during IS development. Secure TROPOS supplies nine new concepts: a) *secure goal*, b) *secure constraint*, c) *secure plan*, d) *secure dependency*, e) *secure resource*, f) *secure capability*, g) *threat*, h) *security diagram* and i) *security features*. These concepts are interconnected as shown on the secure TROPOS metamodel in Fig. 9.7 [1]. However, we are going further by considering in our analysis the improvement of secure TROPOS as described in [MG04]. Secure TROPOS is completed with three sub-activities. 1) The system's architectural style selection following its security requirements; 2) the transformation of the security requirements to a design meeting these requirements and 3) the attack testing of the system under development. We will explain in the sequel of the thesis how these three sub-activities are integrated to the secure TROPOS process. But, first, let us have a look at the definition of the new security concepts.

- **secure goal**: is a (hard)goal but with respect to security. A *secure goal* contributes to attain *security constraints* that are imposed to an *actor* or exist in the system (see on secure

[1] Note that *secure capability* and *threat* are not represented in this metamodel because they are concepts but not constructs of secure TROPOS.

TROPOS metamodel in Fig. 9.7). But a secure goal does not give a precise definition of how the secure constraint can be achieved, seeing that there are several possible alternatives;

- **security constraint**: is a restriction related to security interests (such as availability, integrity and privacy) which can influence the analysis and the design solutions, by conflicting with some of the requirements of the system, or by refining some of the system's objectives [Mou04];
- **secure plan**: is *a way for satisfying* (fulfilling as expressed on Fig. 9.7) *a secure goal* [BGG$^+$04] and is executed by an actor. In other words, a secure plan is a mechanism to assure protection to achieve security objectives. It may be noticed that a secure plan can have a **positive** contribution (what is expected) but also a **negative** contribution to a security objective;
- **secure dependency**: *introduces security constraint(s) that must be fulfilled for the dependency to be satisfied* [MGM03]. The depender as well as the dependee must agree on the fulfilment of the *security constraint* in order to make the secure dependency valid. As underlined in [MG04], "*that means the depender expects from the dependee to satisfy the security constraint(s) and also that the dependee will make effort to deliver the dependum by satisfying the security constraint(s)*";
- **secure resource**: informational or physical entity related to security;
- **secure capability**: ability of an actor to achieve a *secure goal*, carry out a *secure plan*, and deliver a *secure resource*;
- **threat**: "*represents circumstances that have the potential to cause loss or problems that can put in danger the security features of the system*" [MGGP02];
- **security features**: also named protection properties, are features related to security that the system-to-be must have. A security feature is represented using softgoal syntax on the security diagram;
- **security diagram**: a new diagram is added to the diagrams built by TROPOS method, the ***S**ecurity **D**iagram* (SD) (an example is presented in [MGGP02]). This diagram is constructed to gather the security requirements previously elicited during the secure TROPOS process (we will describe this part in the sequel of this section). As presented in Fig. 7.11, SDs are built using the concepts of *security features*, *secure goal*, *secure plan* and *threat*. The positive or negative contributions of threats and secure goals are analysed, helping the analyst to have an overview of the security requirements of the system-to-be.

The graphical syntax of TROPOS is extended with four new shapes as represented in Fig. 7.11. The secure TROPOS process aims to achieve three objectives:

1. to identify the security requirements of the system-to-be;
2. to build a design that meets the specified security requirements elicited;
3. to validate the system-to-be with respect to security concerns.

The process is composed of four stages: a) environment security analysis, b) system security analysis, c) system design and d) validation. Each stage gets closer to an operational representation of the system.

Environment Security Analysis

From this first phase, three main outcomes are expected: a) the comprehension of the environment, b) the identification of security concerns imposed by the environment and c) the realisation of a balance analysis. This phase thus correspond to the early requirements analysis phase described in the TROPOS method. During this phase, the SD is built but also the *actor diagram*

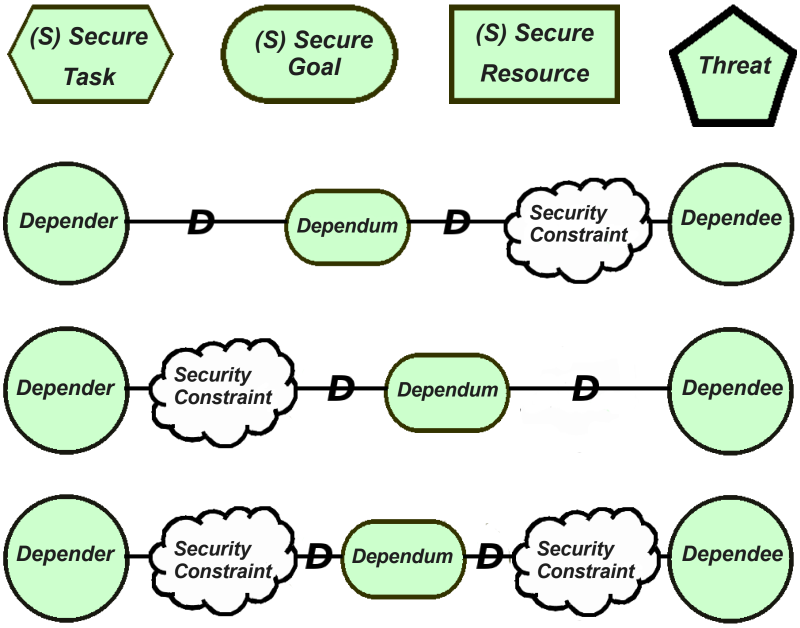


Figure 7.11: The secure TROPOS graphical syntax (adapted from [MG04])

with the inclusion of security constraints. In addition, the security constraints imposed to actors are further investigated by identifying which goals of the actors are restricted by these *security constraints*. The assignment of a *security constraint* to a goal is depicted on the *means-ends diagram* by the use of a constraint link having the "restricts" tag.

System Security Analysis

This phase corresponds to the late requirements analysis from the TROPOS method. In this phase, the system-to-be is under further investigation. System security requirements are elicited and the SD is used to add security concerns on the *means-ends diagram* of the system, leading to the introduction of *secure goals* that are themselves detailed in terms of *security plans*.

System Design

System design gathers the *architectural design* and *detailed design phases* and goes further by introducing testing of the developed design. The architectural design defines the system's global architecture. The global architectural organisation is defined by selecting among alternative architectural solutions using, as criteria, the non-functional requirements of the system. This corresponds to the first of the three sub-activities introduced in [MG04], *the system's architectural style selection following its security requirements*. To help the analyst in measuring these non-functional requirements, it is proposed to evaluate the concept of *satisfiability* discussed by Giorgini *et al* in [GMNS02]. A weight is used to express the value of the *satisfiability*. The analysis involves the refinement of the previous non-functional requirements to obtain more specific ones. Note that selecting between alternative architectural styles is achieved by evaluating the secure plans. Once the weights have been defined, the non-functional requirements have to be classified according to the importance to the system and the identification of the architectural style that best satisfies the most important of these non-functional requirements. More details about how to attain this classification can be found in [GMNS02]. When an architectural style is chosen, *the transformation of the security requirements to a design meeting these requirements* can begin (this stage corresponds to the second sub-activity explained in [MG04]). New actors and their dependencies have to be introduced as well as decomposing the existing actors into sub-actors and the delegation of security responsibilities from the actors to the sub-actors. The last stage of the *architectural design phase* concerns the identification of capabilities for each actors by "*taking into account dependency relationships of the actors*" [GMZ06].

The *detailed design phase*, the previously elicited components of the system are further specified. The actor's capabilities and the interactions related to security are described in detail.

A final phase takes place: *the attack testing of the system under development* (that is the third sub-activity in [MG04]). The main aims is to let the developer see how his solution copes with potential attacks on the newly built system. This process is based on Security **A**ttack **S**cenarios (SAS). A SAS is defined in [MG04] as "*an attack situation describing the agents of a multiagent system and their secure capabilities as well as possible attackers and their goals, and it identifies how the secure capabilities of the system prevent (if they prevent) the satisfaction of the attackers' goals*". A scenario provides enough information about the system and its environment to allow validation of the security requirements. An attacker is defined in [MG04] as "*an agent who aims to break the security of the system*". *Attacker* intentions are modelled as goals and plans and are elicited following the same reasoning as the one used in the TROPOS method in the goals and plans analysis. The *attacks* are graphically represented by dashed links, named *attack links* and identified by a "attacks" tag. The origin of an *attack link* is the attacker's goal and at its end is the *attacked resource.* Secure capabilities of agent that can help to prevent identified attacks are represented by helps dashed links. More explanations about the SAS building process is available in [MG04]. Hence, a SAS helps to understand not only the method used by an attacker to potentially harm the system but also **why** he wants to do so. Thus leads to the redefinition of new *secure capabilities* and/or *secure entities* to protect the system from these attacks.

Validation

The last phase of secure TROPOS method is the *validation phase.* Two types of validation are allowed:

- **the model validation**: involves the validation of the developed models with respect to a set of validation rules:
  - **inter-model rules**: which help to check the consistency inside a model;
  - **outer-model rules**: which help to check the consistency between the different models of a process.
- **the design validation**: corresponds to the validation of the developed solution with respect to the security constraints of the system.

The three next modelling languages are not so well described as they are not directly presented in the thesis. However, as they were investigated during the master thesis and as they are relevant for security modelling during the early requirement engineering, we give an overview of them just below.

7.3 Misuse Cases

Misuse cases, as described in [SO04], are an extension of the UML Use cases diagrams that permits to express behaviour not wanted in the system-to-be in order to elicit security requirements. Misuse cases are composed of a graphical model that is described textually as for Use cases. We focus on the graphical representation that is the same as for Use cases diagrams but using inverted colors as depicted in Fig. 7.12. As can be observed, Misuse cases are easily understandable without a long speech. New concepts introduced in Misuse cases are:

- **Misuse case**: "*a sequence of actions, including variants, that a system or other entity can perform, interacting with misusers of the entity and causing harm to some stakeholder if the sequence is allowed to complete*" [SO04];
- **Misuser**: "*an actor that initiates misuse cases, either intentionally or inadvertently*" [SO04];
- **Mitigate relationship**: some Use cases can be defined as countermeasures against Misuse cases. In this case, the target that is mitigated by a Use case is represented with a Mitigation relationship. You can see an example of this relationship in Fig. 7.12;
- **Threaten relationship**: conversely, the targeted Use case that a Misuse case wants to harm is indicated with a Threaten relationship, as depicted in Fig. 7.12.

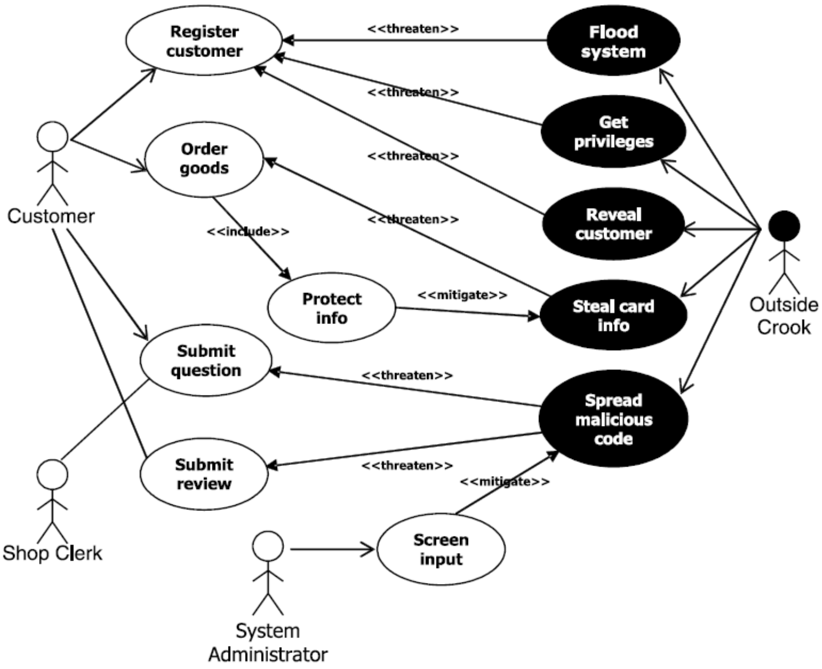


Figure 7.12: Misuse case example (from [SO04])

7.4 Abuse Cases

In [LNI+03a], J. McDermott and C. Fox present Abuse cases. They are another, less widespread, security extension of UML Use cases. Abuse cases are described as "*a specification of a type of complete interaction between a system and one or more actors, where the results of the interaction are harmful to the system, one of the actors, or one of the stakeholders in the system*". Abuse cases focus on the abused privileges and describe a range of privileges that might be used to accomplish the abuse. Harms caused by Abuse cases are textually explained.

Abuse cases do not add new syntax in the existing Use case syntax. They are represented with the same constructs but seen from another point of view – in this case, the attacker's point of view. To let the analysts make differentiate them, Abuse cases are not modelled in the same diagram as Use cases.

As explained in [SO04], Abuse cases and Misuse cases are not two competing languages used to model security risk in Use cases models. They are complementary to each other. The Abuse cases focus specifically on security requirements and their relation to design and testing, whereas the Misuse cases take into consideration the elicitation of security requirements in relation to other requirements. Fig. 7.13 depicts an example of Abuse cases diagram (based on [MF99]). This example presents Abuse cases of an Internet-Based Information Security Laboratory.

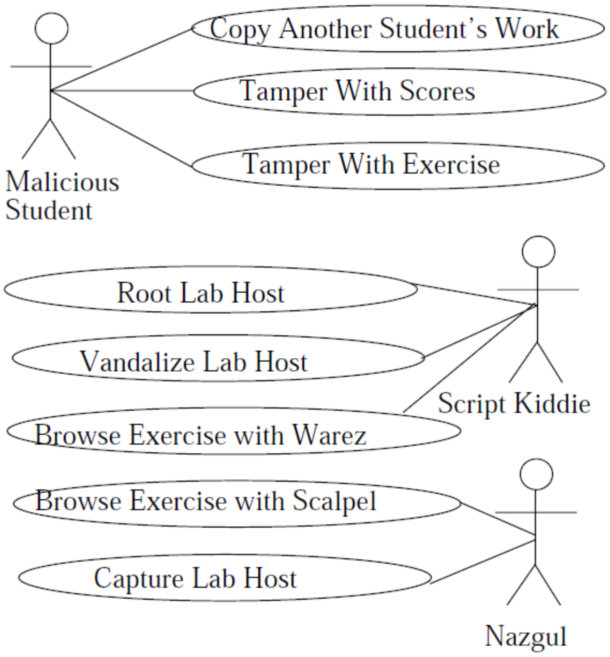


Figure 7.13: Abuse case example (from [MF99])

7.5 Abuse Frames

Abuse frames, described in [LNI$^+$03b], are an extension of Jackson's Problem Frames. When we analyse a problem, we have to identify the problem and to make the descriptions needed to solve it. These two **levels** of analysis are not sufficient to handle most of the realistic problems due to their complexity and their size. Another level is needed: structuring the problem as a collection of interacting sub-problems that are smaller and simpler than the initial problem. This is exactly what Problem Frames aim to.

Abuse frames allow to deals with security threat in the early requirements phase. Abuse frames represent security threats against the system-to-be and use the same notation as Problem frames. But each Domain is associated with a different meaning:

- the **Machine domain** contains the vulnerabilities exploited by malicious users in order to achieve an attack;
- the **victim domain** identifies the asset that are under attack;
- the **malicious user domain** and an anti-requirement define the threat agent.

We define more precisely the concept of anti-requirement: an anti-requirement indicates the intention of a malicious user that subverts an existing requirement of the system-to-be. It "*defines a set of undesirable phenomena imposed by the malicious user that will ultimately cause the system to reach a state that is inconsistent with the system requirements.*" [LNI$^+$03b].

Fig. 7.14 represents a generic Abuse frame. We can see the Machine domain, the Malicious user and the Anti-requirements as well as the Assets (Victim domain) that are threatened.

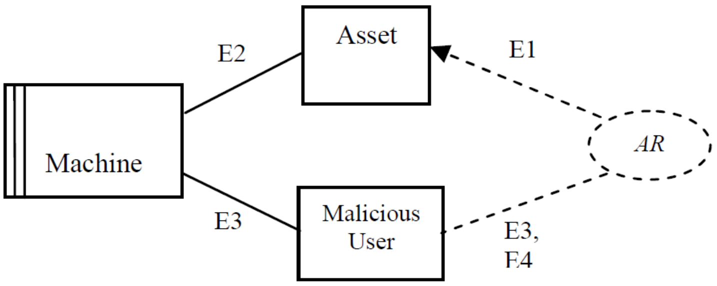


Figure 7.14: A Generic Abuse frame (from [LNI$^+$03b])

7.6 Summary

We give of detailed description of two goal modelling languages – KeS and secure TROPOS – that we will analyse in detail in Chapter 9. KAOS extended to Security (KeS) is an security extension of the KAOS language and secure TROPOS is an security extension of the TROPOS language. KeS was depicted through an example. We do not illustrate secure TROPOS at this stage of the analysis because its concepts and their relationships need to be defined in more details. An example will be given in the sequel. We also gave a short introduction to the Misuse cases, the Abuse case and the Abuse frames. They are respectively security extensions of Use cases (for Misuse and Abuse cases) and Problem frames.

Part II

Contribution

Chapter 8

Eliciting ISSRM Relationships

Relationship elicitation is the first part of our contribution to the building of the ISSRM domain model. It contributes to the step 2 of the research method (see Chapter 4). The elicitation was performed of each of the security and risk related sources (discussed in Chapter 5). Outcomes are gathered in Appendix 2. In this section, we discuss why relationships between concepts contained in the ISSRM sources need to be elicited and we present, as example, the results of elicitation for EBIOS v2.

8.1 Improve Relevant Concepts Comprehension

After having extracted relevant concepts in several ISSRM sources and having tried to harmonize terminologies, a straightforward observation unveils itself. It is not possible to harmonize terminologies without further information. Indeed, several imprecise elements in concept definitions introduce too many doubts to confirm synonymy between concepts of the different ISSRM sources. For example, security and risk related sources use some identical terms but with a completely different semantic, what is called a threat in a source is presented as a risk in another one. Moreover, in the context of these sources described in Chapter 5, there are also discrepancies or too shadowy descriptions making concepts sense fuzzy. Faced with this situation, achieving an ISSRM domain model can seem impossible. So we decided to create, for each ISSRM source, a graphical representation of its relevant concepts in order to improve their comprehension. A UML Class diagram is suitable to this kind of representation as what we will design is a kind of metamodel of the concepts described in each source and that metamodels are usually modelled using an UML Class diagram. Each concept is translated into a UML Class and is decorated with an UML tagged value that corresponds to the ISSRM concept considered as equivalent to this concept. However, a key concept of UML Class diagram is still missing: relationships between concepts. In order to complete Class diagram, ISSRM source documents have to been analysed, focusing on elements in the sentences that concern relationships elicitation.

At the end of this work, outcomes are:

1. an improved validation of the concepts alignment in the grid. Relationships between concepts can increase our comprehension of the concepts and so help to validate the result of the alignment;

2. a metamodel of each ISSRM source. The metamodel gives an illustration of concepts and relationships identified, and it is easier to validate by external people than the list of concepts and relationships. It can also act as a quick guide to the source which is often a large document;

3. a summary of the relationships between ISSRM concepts. This will be the reference work for defining relationships between concepts in the ISSRM metamodel.

8.2 Relationships Extraction

The process used to extract relationships between concepts from ISSRM sources is the same as the one applied for the elicitation of concept definitions. As for the associated definitions document, a document about relationships is also created. It contains excerpts of sentences from ISSRM sources in which a proof or, at least, a clue of the existence of a relationship can be found. In order to have a more efficient reading of this document, a special notation described below is used.

8.3 Structure of the Relationship Explanation Document

As said before, models are represented following the UML Class Diagram syntax. The three most common types of relationships have been retained: *association*, *aggregation* and *inheritance* (generalization / specialization).

Explanations about these relationships can be found in [Hey05]:

- an **association**, in the context of UML Class diagrams, is defined as a link between two or more related concepts, which are represented by classes. Each end of an association has a role. This role consists of a name, a visibility (optional) and a multiplicity. Multiplicities can be of many types, for example, one-to-one, many-to-many and one-to-many and many-to-one. In this step, only associations linking two and only two classes are under investigation. The reason is explained in the sequel. An illustration of an association can be found in Fig. 8.1;
- an **aggregation**, in the context of UML Class diagrams, is a special kind of association used when instances of a class are aggregates of instances of another one.
- an **inheritance**, a global name chosen to enclose generalization and specialization, is a hierarchical relationship, allowing to manage complexity by representing, in the same diagram, several levels of abstraction, several levels of granularity.

Notational Convention

An **association** relationship is represented by a UML association between two concepts (represented by UML classes). It respects the UML notation as depicted in Fig. 8.1.
Example: ***Threat.exploits - Vulnerability.exploits*** *can be read: a* ***Threat*** *exploits a* ***Vulnerability****. (NB : spaces in role names are replaced by underscores). Multiplicities are not added because in most cases no clues were given in the documentation.*

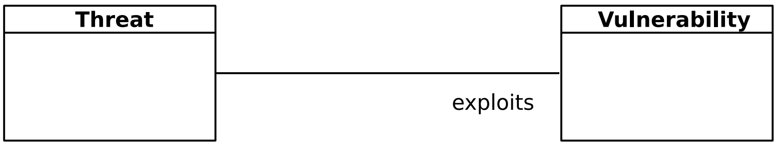


Figure 8.1: UML association relationship (from EBIOS v2 Class diagram)

An aggregation relationship is represented by a UML aggregation. It respects the UML notation as depicted in Fig. 8.2.

Example: ***Aggregation: Threat - Attack*** *means that the class* ***Threat*** *is composed of the class* ***Attack****.*

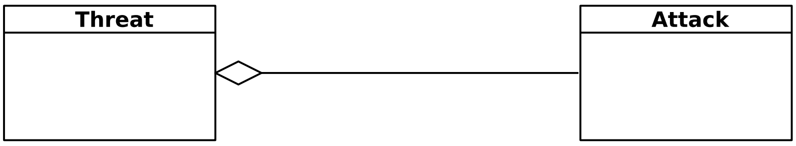


Figure 8.2: UML aggregation relationship (from EBIOS v2 Class diagram)

An inheritance relationship is represented by a UML Generalization/Specialization. It respects the UML notation as depicted in Fig. 8.3.
Example: ***Inheritance: Asset - Entity*** *can be read: a* ***Entity*** *is a kind of an* ***Asset****. So the class* ***Asset*** *is a supertype of the class* ***Entity****.*

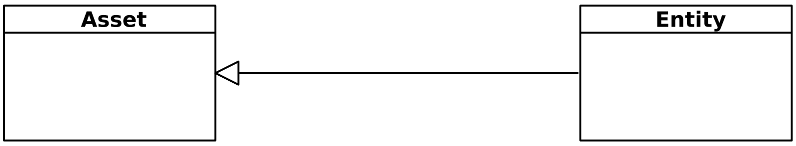
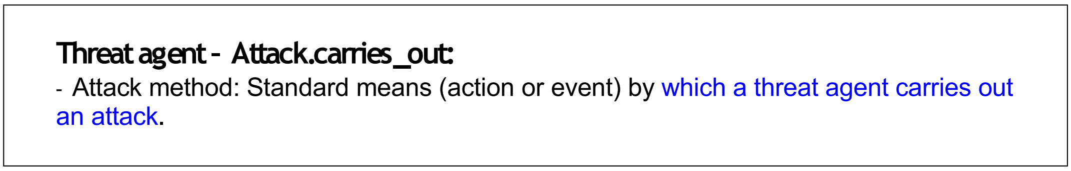


Figure 8.3: UML inheritance relationship (from EBIOS v2 Class diagram)

Each relationship (association, aggregation, inheritance, ...) is explained by

- either a single paragraph (or more but with similar terms or semantics) with essential keywords. They are strong proofs (they do not require any kind of interpretation) of the existence of the relationship. These keywords are in blue italic font in the document as depicted in Fig. 8.4;

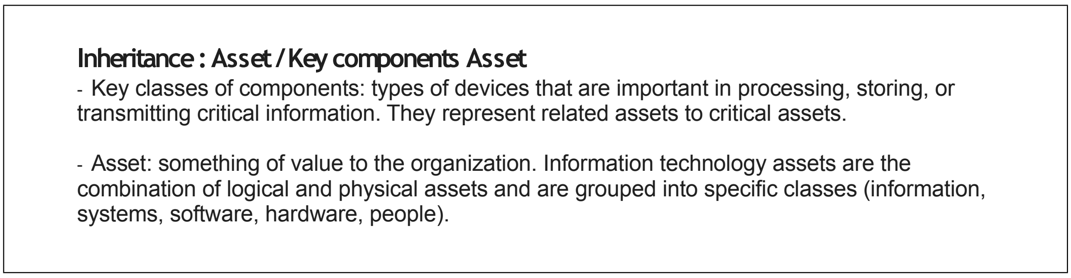
- either by one or more paragraphs in which users have to do some interpretations to understand the relationship. In this case, no keywords are directly given. An example is presented in Fig. 8.5.



Threat agent - Attack.carries_out:

- Attack method: Standard means (action or event) by which a threat agent carries out an attack.

Figure 8.4: Direct elicitation of the existence of the relationship in source document (excerpt of EBIOS v2 glossary)



Inheritance : Asset / Key components Asset

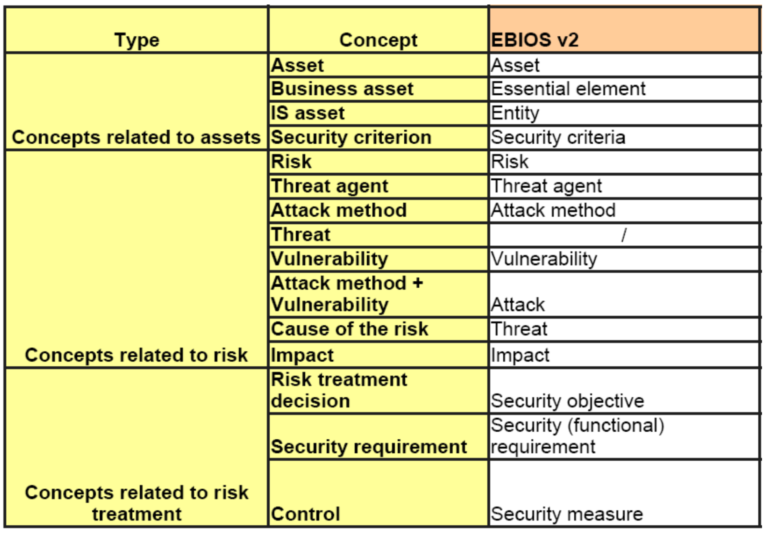
- Key classes of components: types of devices that are important in processing, storing, or transmitting critical information. They represent related assets to critical assets.

- Asset: something of value to the organization. Information technology assets are the combination of logical and physical assets and are grouped into specific classes (information, systems, software, hardware, people).

Figure 8.5: Concept that require interpretation to understand the relationship in source document (excerpt of OCTAVE 2.0 glossary)

8.4 From Theory to Practice

We apply the process we just described, to the case of EBIOS v2 [DCS04]. This security risk management method has been defined in Chapter 5. Just before the extraction of EBIOS relationships, the previous step had produced the grid presented in Fig. 8.6.



Type	Concept	EBIOS v2
Concepts related to assets	Asset	Asset
	Business asset	Essential element
	IS asset	Entity
	Security criterion	Security criteria
Concepts related to risk	Risk	Risk
	Threat agent	Threat agent
	Attack method	Attack method
	Threat	/
	Vulnerability	Vulnerability
	Attack method + Vulnerability	Attack
	Cause of the risk	Threat
	Impact	Impact
Concepts related to risk treatment	Risk treatment decision	Security objective
	Security requirement	Security (functional) requirement
	Control	Security measure

Figure 8.6: EBIOS v2 terminology summarized in the concept grid

The symbol "/" means that the ISSRM source has no concept equivalent to the one described in the column "*Concept*" of the grid. The associated definitions related to these relevant concepts are described in Appendix 2.

At the end of step 1 of the research method (see Chapter 4), a first version of a UML Class diagram for EBIOS before relationships elicitation is as depicted in Fig. 8.7 where EBIOS concepts are not related to each other.

8.4.1 Elicitation of Relationships

As we said before, for each concept, a tagged value, whose name correspond to the suitable ISSRM concept, has been added. Moreover, the relative positioning and the background colour of each class on the Class diagram depends of the membership of this class to one of the three categories defined in Chapter 6:

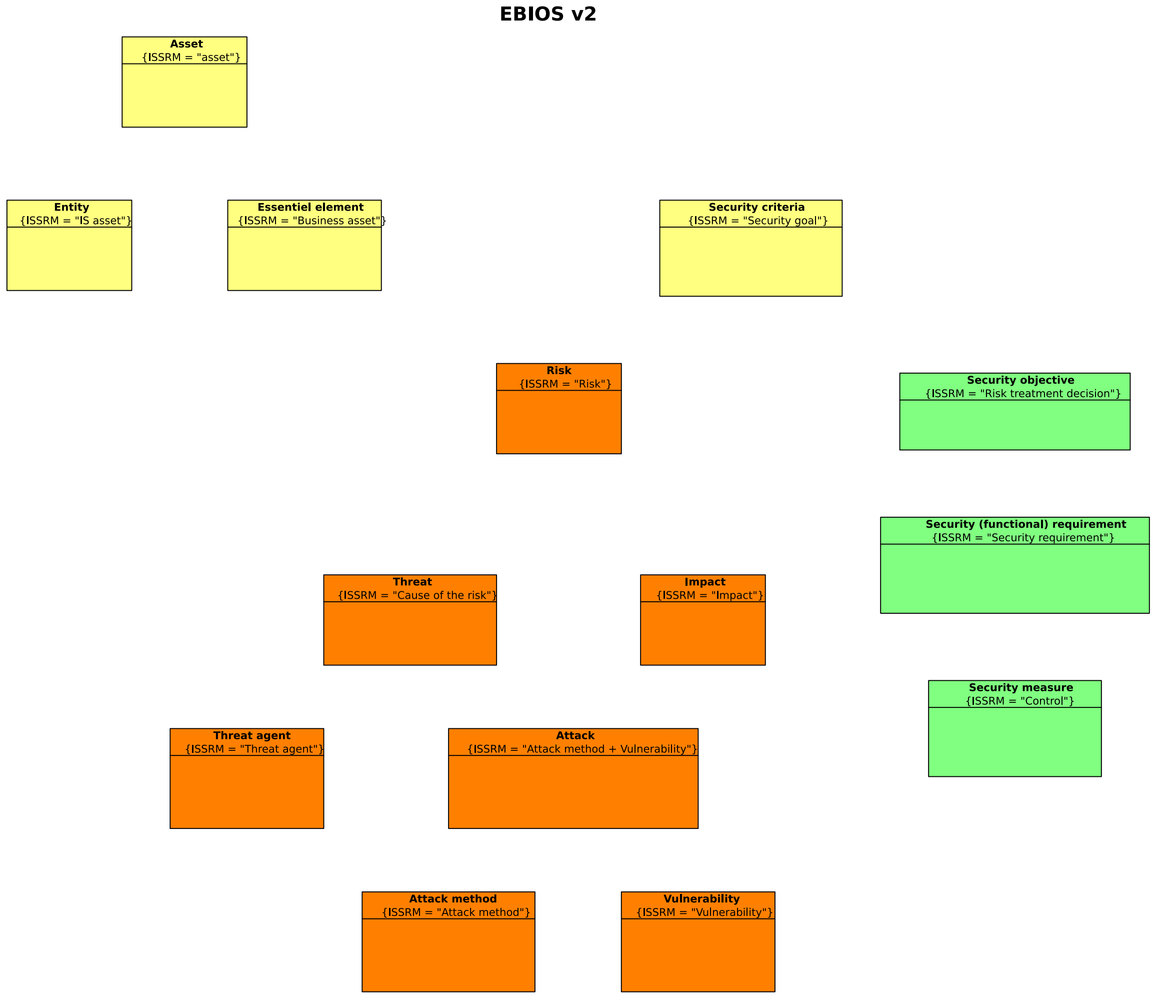

EBIOS v2
Asset
{ISSRM = "asset"}
Entity
{ISSRM = "IS asset"}
Essentiel element
{ISSRM = "Business asset"}
Security criteria
{ISSRM = "Security goal"}
Risk
{ISSRM = "Risk"}
Security objective
{ISSRM = "Risk treatment decision"}
Security (functional) requirement
{ISSRM = "Security requirement"}
Threat
{ISSRM = "Cause of the risk"}
Impact
{ISSRM = "Impact"}
Security measure
{ISSRM = "Control"}
Threat agent
{ISSRM = "Threat agent"}
Attack
{ISSRM = "Attack method + Vulnerability"}
Attack method
{ISSRM = "Attack method"}
Vulnerability
{ISSRM = "Vulnerability"}

Figure 8.7: UML Class diagram of EBIOS v2 before relationships elicitation

- Asset block: beige colour;
- Risk block: orange colour;
- Countermeasure block: green colour.

The next task is the elicitation of the relationships between concepts presented on the Class diagram. The elicitation is made using common sense, i.e. by reading, comparing and discussing with experts. Using EBIOS v2 sources [MHM07] and [DCS04], several relationships are extracted and are summarized below. Relationships between concepts of the same groups are firstly considered and, afterwards, relationships between concepts of different groups. Groups are analysed in the order: asset, risk, countermeasure as detailed in Tables 8.2, 8.1, 8.3, 8.4, 8.5.

We detail the elicitation of relationships for the Aggregation link between a Risk and a Threat. We can find this definition of a Risk in EBIOS sources [DCS04]: *Risk: Combination of a threat and the losses it can cause.* The term **Combination of** can be translated as an aggregation relation between the Risk – the term being described – and the Threat – the term designed as a component of the Risk.

The existence of the inheritance relationship between the concepts of **Asset** and **Key component asset** requires a part of interpretation (see Fig. 8.5). In this case, two important ideas can be found in the description of the concept of **Asset**: (a) something that has value for the organisation and (b) information technology asset (logical or physical assets). The description of the **Key component asset** contains the following notions: (a) types of devices that are important in processing, storing, or transmitting critical information. and (b) they represent assets related to critical assets. So, in the **Key component asset**, we can found the notion of **value** contained in **Asset** – something important for an organisation is something that has value for it – and the term **device** can be understood as a **physical asset**. The sentence "*they (types of devices – **Key classes of components**) represent assets related to critical assets*ťť seems to indicate that a **Key classes of components** is a subtype of **Asset**. As one can see, the elicitation of this inheritance relationship requires the interpretation of the analyst.

We follow the same line of reasoning for the rest of the relationships.

Relationships of the Assets block	
Inheritance: Asset - Entity	*Entity: An asset* such as an organisation, site, personnel, equipment, network. [MHM07]
Inheritance: Asset - Essential Element	Essential element: ... Examples: - list of names; - certification request; - invoice management; - encryption algorithm; - etc. [MHM07] Asset: ... Examples: - list of names; - certification request; - invoice management; - encryption algorithm; - laptop computer; - Ethernet; - operating system; - etc. [MHM07]
Security Criteria.characteristic_of - Essential Element:	The essential elements are usually functions or information for which the owner's or holder's responsibility would be called into question, or which would result in damage to the organisation or third parties, if their availability, integrity, confidentiality or other security criteria were not guaranteed. (p. 11) [DCS04]

Table 8.1: Elicitation of relationships between concepts of Assets block

Relationships of the Risk block	
Aggregation : Risk - Threat	*Risk: Combination of a threat and* the losses it can cause. [MHM07]
Aggregation : Threat - Threat Agent	Threat: Possible attack of a threat agent on assets. [MHM07]
Threat Agent.carries_out - Attack:	*Attack method*: Standard means (action or event) by *which a threat agent carries out an attack.* [MHM07]
Threat Agent.uses - Attack Method:	The attack methods are used by threat agents which must be characterised for each attack method. (p. 26) [DCS04]
Aggregation : Threat - Attack	Threat: Possible attack of a threat agent on assets. [MHM07]
Aggregation : Attack - Attack Method	Attack: Exploiting one or more vulnerabilities using an attack method with a given opportunity. [MHM07]
Attack Method.exploits - Vulnerability:	Attack: Exploiting one or more *vulnerabilities using an attack method* with a given opportunity. [MHM07]
Aggregation : Attack - Vulnerability	Attack: Exploiting one or more vulnerabilities using an attack method with a given opportunity. [MHM07]
Aggregation : Risk - Impact	*Risk: Combination of* a threat and *the losses it can cause.* [MHM07]
Impact.consequence_of - Threat:	Impact: *Consequences* for an organisation *when a threat is accomplished.* [MHM07]

Table 8.2: Elicitation of relationships between concepts of Risk block

Relationships between the Assets and the Risk blocks	
Risk.significance_assessed_by - Security Criteria:	Security criterion: Characteristic of an essential element allowing *the various sensitivities to be assessed.* [MHM07]
Threat.attacks - Entity:	Threat: Possible attack of a threat agent on assets. [MHM07]
Impact.consequence_on - Essential Element:	Impact: Consequences for an organisation when a threat is accomplished. [MHM07] Threat: Possible attack of a threat agent on assets. [MHM07] Essential element: Information or function with at least one non-nil sensitivity. [MHM07]

Impact.damages - Security Criteria:	The consequences of any damage can be assessed from several points of view. The significant impacts for the organisation must be identified by the manager using the system. Here are some examples of damage relating to the main security criteria (the situation and context should lead to listing specific damage for each criterion selected) (p. 22) [DCS04]:
	• for availability:
	– degrading of performance;
	– short interruption;
	– long interruption;
	– inaccessibility;
	– total loss (destruction).
	• for integrity:
	– accidental modification;
	– deliberate modification;
	– incorrect results;
	– incomplete results.
	• for confidentiality:
	– internal disclosure;
	– external disclosure.

Table 8.3: Elicitation of relationships between concepts of Assets block and Risk block

Relationships between the Countermeasures blocks	
Risk Treatment.needs_definition_of - Security Objective:	Risk treatment: Process for selecting and implementing measures aimed at modifying the risk. [MHM07] Security functional requirements: to contribute to covering one or more security objectives for the target system. [MHM07] Security objectives: Expression of the intention to counter identified threats or risks. [MHM07] The security functional requirements contribute to the treatment of ISS risks, which may consist not only in reducing them, but also in rejecting, transferring or assuming them. (p. 38) [DCS04]
Inheritance: Risk Treatment - Risk Reduction	Risk reduction: Process aiming to minimise the negative consequences and opportunities of a threat. [MHM07] Risk treatment: Process for selecting and implementing measures aimed at modifying the risk... [MHM07]
Inheritance: Risk Treatment - Risk Rejection	Risk rejection: "Rejection of a risk will result in security functional requirements for a structural modification of the target system situation that eliminates its exposure to the risk." [MHM07] Risk treatment: Process for selecting and implementing measures aimed at modifying the risk. [MHM07]
Inheritance: Risk Treatment - Risk Retention	Risk retention: Acceptance of the possible loss associated with a particular risk. [MHM07] Risk treatment: Process for selecting and implementing measures aimed at modifying the risk. [MHM07]
Inheritance: Risk Treatment - Risk Transfer	Risk transfer: Sharing with another party the possible loss associated with a particular risk. [MHM07] newline Risk treatment: Process for selecting and implementing measures aimed at modifying the risk. [MHM07]
Security Functional Requirement.covers - Security Objective:	*Security functional requirements*: to contribute to *covering* one or more *security objectives* for the target system. [MHM07]
Security Measure.implements - Security Functional Requirement:	Security measure: *A measure* designed to improve security, *specified by a security requirement and implemented* to comply with it. [MHM07]

Table 8.4: Elicitation of relationships of concepts of Countermeasures block

Relationships between the Countermeasures and the Risk blocks	
Risk Treatment.modifies - Risk:	Risk treatment: Process for selecting and implementing measures *aimed at modifying the risk*, i.e. risk reduction, risk transfer or risk retention. [MHM07]
Security Objective.counters - Risk:	*Security objectives*: Expression of the intention *to counter* identified threats or *risks* (depending on the context). [MHM07]

Table 8.5: Elicitation of Relationships between Concepts of Countermeasures and Risk Blocks

8.4.2 UML Class Diagram of Concepts Completed

Relationships between concepts can now be added to the class diagram related to ISSRM sources. At the end of this process, outputs are the Class diagram as presented in Fig. 8.8, for each of the elements of the four families studied. Comprehension of these methods, frameworks, security or risk management standards is increasingly improved and leads to the creation of the ISSRM domain model. Note that all relationships elicited previously are not drawn on Fig. 8.8 in order to give a big picture of concepts and relationships from EBIOS v2 that are common with the ISSRM domain model.

8.4.3 Creation of the ISSRM Domain Model

Analysing common concepts between all class diagrams and their relationships leads to the design of the ISSRM domain model. Once more, common sense, human deliberation and the experience were used.

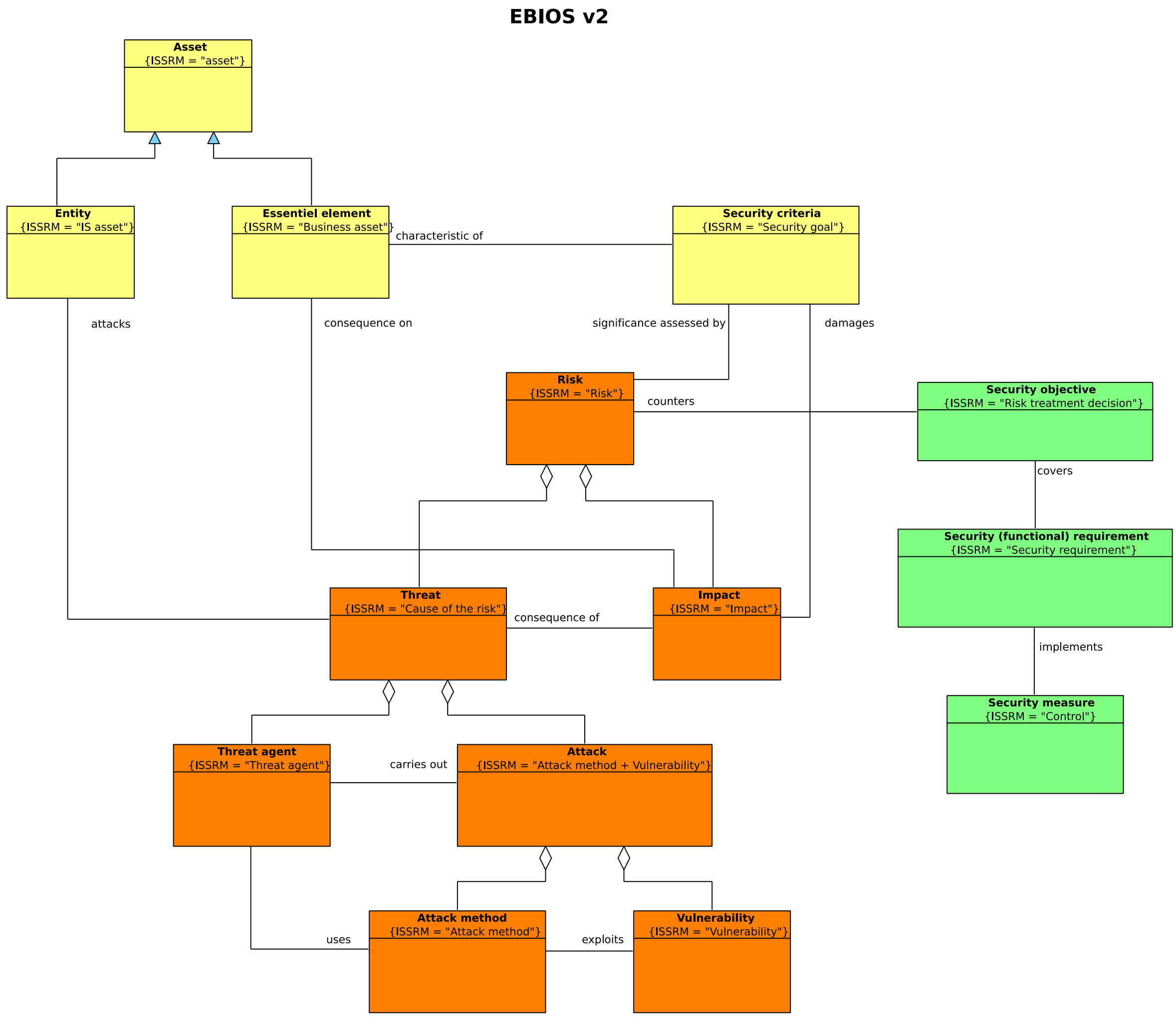


Figure 8.8: Complete UML Class diagram of EBIOS v2

8.4.4 Validation of Relationship Elicitation

We can ask ourselves how it is possible to validate our elicitation of the relationships between concepts described in ISSRM sources. Even if this question is legitimate, we do not really care about this validation. Indeed, relationship elicitation aims to validate the existence of the concepts extracted from ISSRM sources. So from this point of view, we consider that validating the relationships is not really useful. As this elicitation of relationships does not make appear relationships between concepts that are not linked in the same way in the ISSRM domain model, we can consider that the relationship elicitation achieves its main aim.

8.5 Other standards

The security and risk related sources described in Chapter 5 have all been analysed in order to elicitation the relationships between their concepts. We applied the same process that we used for the elicitation of the relationships of EBIOS v2. Outcomes of this elicitation can be found in Appendix 2.

8.6 Summary

In this chapter, we discussed why the extraction of relationships from security and risk related sources was needed in order to enhance the ISSRM domain model. We also explained the way we elicited these relationships and the glossary where they are gathered. We presented the elicitation of EBIOS v2 relationships. Finally we discussed the validity of our extraction of the relationships. This address the first objective of the thesis and we will new focus on the second objective. The elicited relationships were used to improve the ISSRM domain model. The choice of relationships that have been added in the ISSRM domain model was on the behalf of N. Mayer.

Chapter 9

Alignment of KeS and Secure TROPOS with the ISSRM Domain Model

9.1 Step 3 of the Research Method

The ISSRM domain model is to be used as a framework for security modelling languages. We consider security modelling languages that we described in Chapter 7. These five languages – KeS, secure TROPOS, Misuse cases, Abuse cases and Abuse frames – have been analysed. We focus on Kes and secure TROPOS. The analyses of the other security modelling languages can be found in Appendix 3. As underlined in [MHM07]: as outcome of this process, we will obtain the coverage of each existing languages with respect to the ISSRM domain model and a) obviously the gaps to fill to fully support ISSRM but also b) concepts that ISSRM should perhaps take into account. Fig. 9.1 gives a graphical representation of the retained process described just below.

For each of them, the analysis follows this process:

1. comparison between the ISSRM model with the metamodel of related/close languages
   (a) take relevant textual documents as reference documents for the language to analyse;
   (b) search the metamodel of the language;
   → if not available, build it following the method below:
   - put discovered concepts on the metamodel;
   - create a glossary containing each concept and its definition extracted from reference documents (write explicitly the page number and the document where definition comes from);
   - search for relationships (UML association, inheritance, aggregation/composition) between concepts and give a role to the associations (an “active” role [i.e: *conceptA mitigates conceptB* and not *conceptB is mitigated by conceptA*]);
   - create a glossary containing each relationship (and role if available) and sentences, figures, examples from reference documents which explain, prove the existence of this relationship;
   - check if
     – there is no UML semantic inconsistency (cyclic inheritance or cyclic aggregation);
     – there is no security risk domain inconsistency (e.g.: security requirement increases risk);
     – there is no semantically equivalent relationships by transitivity.

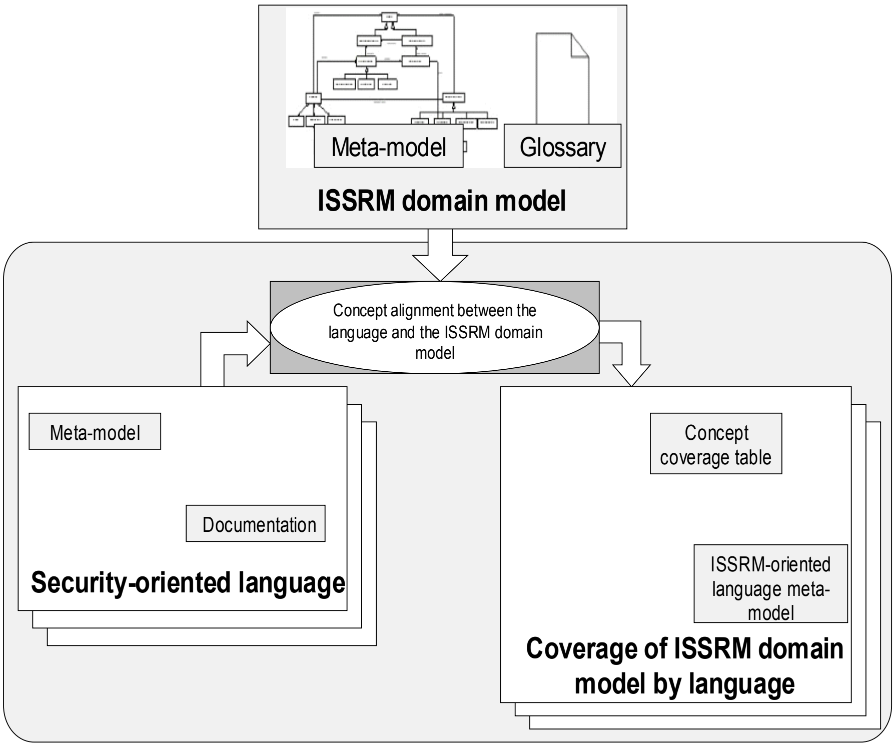

Meta-model
Glossary
ISSRM domain model
Concept alignment between the language and the ISSRM domain model
Meta-model
Documentation
Security-oriented language
Concept coverage table
ISSRM-oriented language meta-model
Coverage of ISSRM domain model by language

Figure 9.1: Comparison between ISSRM domain model and security-oriented languages (from [MHM07])

if none of the 3 cases above : continue.
else iterate the metamodel build process

(c) choose or build an example modelled with the metamodel of the language;

(d) align concepts of the language, based on:

  i. the metamodel of the investigated security risk modelling language;
  ii. the textual descriptions contained in source documents;
  iii. the example obtained in previously.

  with the concepts of the ISSRM domain model. Some interpretations and corrections could be necessary due to the fact that manipulating concept definitions can lead to discover some “hidden sense”. This step is thus done iteratively;

(e) draw the ISSRM-oriented language metamodel of the investigated language. This metamodel contains only the concepts that are aligned with the ISSRM domain model. This metamodel presents the overlap between the security modelling language and the ISSRM domain model;

2. highlight the lack (concepts + relationships) in the supporting modelling language, based on its alignment with the ISSRM domain model. (Note that this stage is performed in Chapter 10).

The process of step 3 is used to drive the alignments of the KeS and secure TROPOS languages (also used for the three other security modelling languages that can be found in Appendix 3). Note that the description of the alignments of KeS and secure TROPOS follows, step by step, the process. So we do not refer directly to it.

9.2 Alignment of KeS with the ISSRM Domain Model

Based on the metamodel of KeS, its description presented in Chapter 7 and the Banking Services example described in 7.1, we proposed in Table 9.2 the alignment of KeS concepts and ISSRM domain model concepts. We split the analysis into three blocks corresponding to the asset-related, risk-related and risk treatment-related concepts.

Asset-related concepts: Kes deals with the security of the system-to-be, but it does not distinguish the IS and business aspects. We align KeS Object as ISSRM concepts of Asset, Business asset and IS asset. KeS states of the system-to-be are described using Object attributes. Introducing security Goals aims to protect the system against security threats. Security Goals should define confidentiality, privacy, integrity and availability of Object attributes. ISSRM security criteria can be aligned with KeS Object attributes.

Risk-related concepts: KeS Anti-goal is also called obstacle or threat. It can be identified at various abstraction levels and, depending on this level, may need to be refined until reaching Anti-requirements or Anti-expectations that are assigned to Anti-agents.

At the higher level of abstraction, we consider the Anti-goal as the ISSRM Cause of the risk. This Cause of the risk is defined in the ISSRM domain model as the combination of a ISSRM Threat and one or more ISSRM Vulnerabilities.

At a lower abstraction level, the Anti-goal (or Anti-requirement or Anti-expectation) denotes an ISSRM Threat which is a potential attack or incident to assets.

ISSRM Vulnerabilities are modelled as KeS Domain properties. A Domain property is an hypothesis about the domain that holds independently of the system-to-be. This corresponds to ISSRM Vulnerabilities that are characteristics of ISSRM Assets.

ISSRM Threat is composed of a Threat agent and an Attack method. A Threat agent can potentially cause harm to the assets. KeS Anti-agent (also called attacker) monitors or controls Objects and their Attributes. An Anti-agent is thus capable of threatening Objects of the system-to-be. We decide to align the ISSRM Threat agent with the KeS Anti-agent.

Concerning the Attack method as defined in the ISSRM domain model, it is clear that it characterises the means by which a Threat agent carries out his attack. KeS Anti-agent performs Operations that satisfy his Anti-goal. Operations change the state of the system-to-be using input/output relationships over the Objects and their Attributes. This means that by performing Operations, the Anti-agent breaks the security criteria related to the Objects attributes. The ISSRM Attack method can be represented in KeS using a set of constructs that are elements of the operationalization of the Anti-goal. These constructs are namely Operationalization, Domain and Required properties and Operations.

The two last concepts of the block of the risk-related concepts are ISSRM Risk and Impact. They have no equivalent in the KeS language. This can be explained by the fact that KeS was not specifically designed to consider the business context of an IS.

Risk Treatment-related concepts: ISSRM Risk treatment decisions correspond to the countermeasures cited in [vL04, vLL]. The countermeasures are elaborated after the identification of Anti-goals. Each Anti-goal can be categorized as described in Table 9.1. The countermeasures are not KeS constructs but are modelling idioms or “patterns” adopted by modellers.

The elaboration of countermeasures usually results in new KeS security Goals that are themselves refined as new realizable security Requirements and security Expectations. This leads us to consider ISSRM security Requirements as the KeS security Goals (security Requirements and security Expectations).

The refinement and the operationalization of the new security Goals, their concerned Objects and Attributes and the Agents who are responsible for them form the new components of the system-to-be realising the necessary security means. These new system components correspond to ISSRM Controls.

Risk treatment decisions	**KeS**	
	Countermeasures	**Description**
Avoiding risk	Goal substitution	Choose an alternative goal producing a different design.
	Goal weakening	Change the specification of a goal so as to make it circumvent the threatening anti-goal.
	Agent substitution	Change the agent assignment (internal) so that the obstacle scenario may no longer occur.
Transferring risk	Agent substitution	Change the agent assignment (outsourcing) so that the obstacle scenario may no longer occur.
Retaining risk	Goal restoration	Add a new goal stating that if the obstacle condition becomes true then the obstructed goal assertion should be satisfied again in some reasonably near future. The behaviour of the obstacle is thus tolerated.
Reducing Risk	Anti-goal mitigation	Tolerate the anti-goal (cause of the risk) but mitigate its effects.
	Anti-goal prevention	Add a new goal requiring the anti-goal to be avoided (mitigated in therms of ISSRM).
	Protect vulnerability	Make the derived vulnerability unmonitorable by the attacker.
	Defuse threat	Make the derived anti-requirement uncontrollable by the attacker.
	Avoid vulnerability	Add a new goal requiring the anti-goal to be avoided (mitigated in terms of ISSRM).

Table 9.1: Correspondence between the ISSRM risk treatment decisions and KeS countermeasures (countermeasures definitions adapted from [vLL, vL04])

Fig. 9.2 and Fig. 9.3 present the KeS metamodel. Each KeS metaclass possesses tagged values indicating the corresponding ISSRM concept and KeS synonyms used in [vL04]. Moreover, metamodel elements are gathered according to corresponding ISSRM domain model elements. The two figures present the alignment of the KeS metamodel with respect to the ISSRM domain model (see Chapter 6). Fig. 9.2 identifies the asset- and risk-related concepts of the ISSRM domain model described in [vL04]. Fig. 9.3 presents the security Goals, which correspond to the ISSRM security Requirements. Other ISSRM security treatment-related concepts are not modelling constructs of the KeS language but modelling idioms or patterns (see Table 9.1). On the other hand, the analysis of security Goals involves the specification of new goal models, which metamodel is depicted in Fig. 7.1.

ISSRM domain model		**KeS**		
		Synonyms	**Language concept**	**Element from our example**
Asset-related concepts	Asset	Asset, object	Object	Database Of Account Numbers
	Business asset			
	IS asset			
	Security criteria	–	Object attribute(s) concerned by Anti-goal	AccountNumbers, PinCodes
Risk-related concepts	Risk	–	–	–
	Impact	–	–	–
	Cause of the risk	Malicious obstacle, Anti-goal, Goal-anchored, Goal negation, Anti-requirement, Anti-expectation	Obstacle (when negating security Goal), Goal, Requirement and Expectation (in anti-model)	Payment Medium Known By Thief, Matching Accounts Found, Account Checked For Pin Match
	Threat			
	Vulnerability	Vulnerability, Domain property	Domain property	CheckRepeatable
	Threat agent	Attacker, Malicious agent, Anti-agent	Agent	Attacker
	Attack method	Potential capabilities of the attacker	Operationalization + Domain and required conditions + Operations	Input Next Account Number, Check If Account Number Match
Risk treatment-related concepts	Risk treatment decision	Countermeasures	–	Vulnerability avoidance
	Security requirements	Security goal, Security requirements, Security expectations	Goal, Requirement, Expectation	Avoid[Repeatable Pin Check From Account Number], Avoid[Repeatable Account Number Check From Pin]
	Control	–	New model implementing security components	–

Table 9.2: Alignment between KeS and the ISSRM domain model

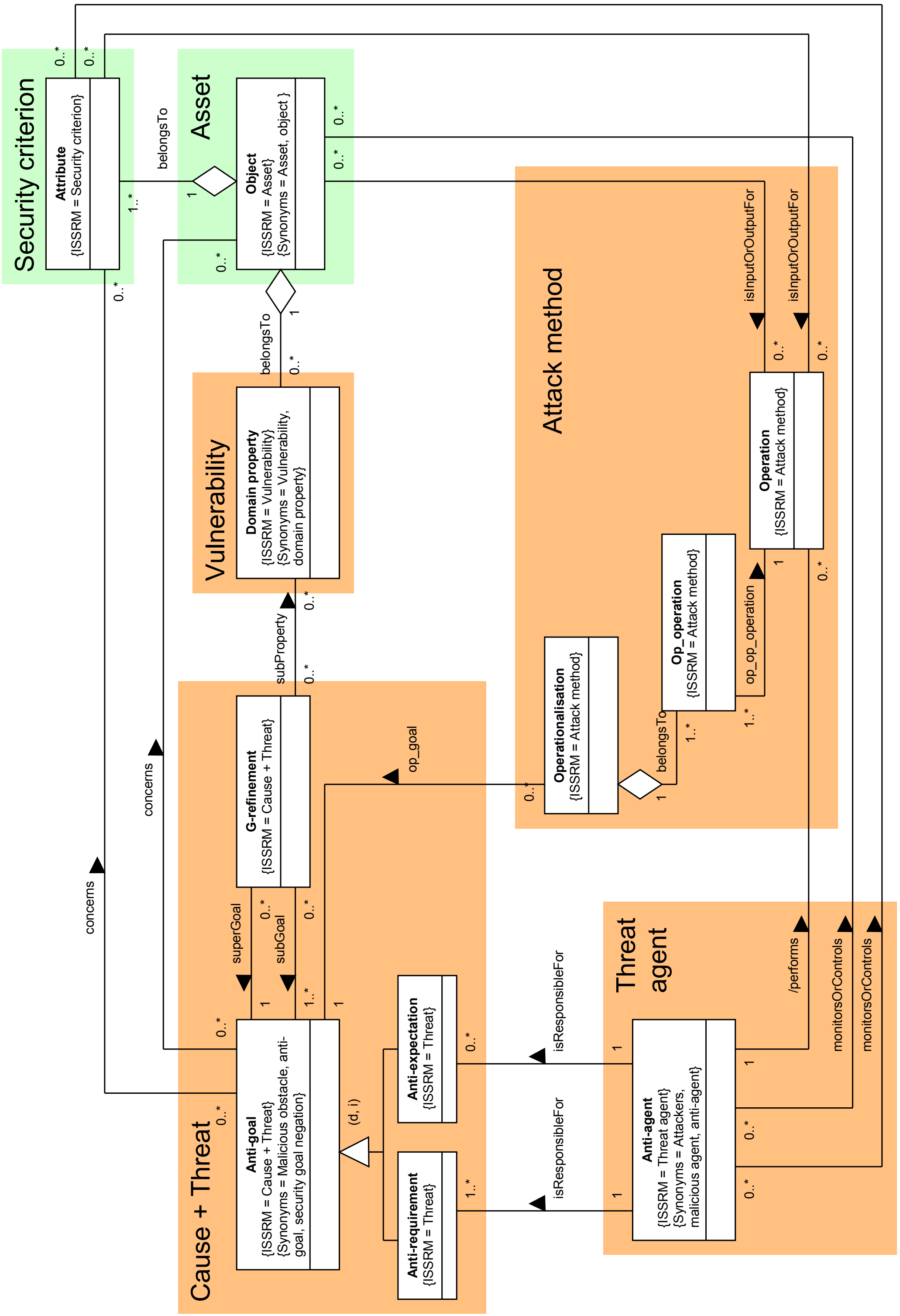


Figure 9.2: Alignment of KeS with ISSRM domain model (focus on risk- and assets-related concepts)

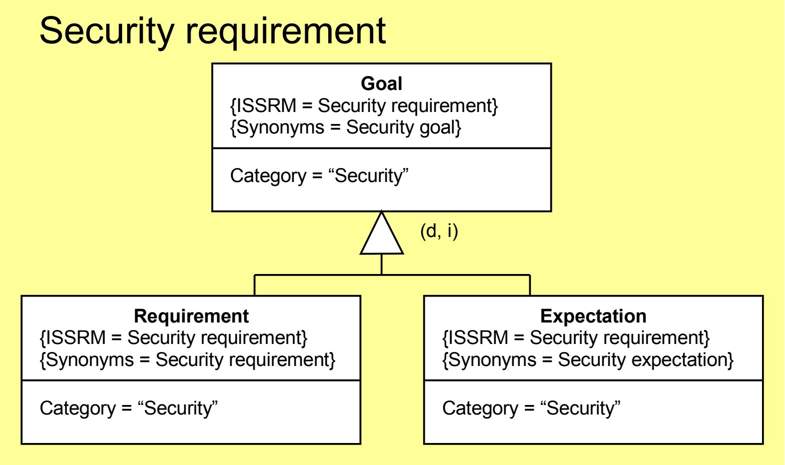

Security requirement
Goal
{ISSRM = Security requirement}
{Synonyms = Security goal}
Category = "Security"
(d, i)
Requirement
{ISSRM = Security requirement}
{Synonyms = Security requirement}
Category = "Security"
Expectation
{ISSRM = Security requirement}
{Synonyms = Security expectation}
Category = "Security"

Figure 9.3: Alignment of KeS with ISSRM domain model (focus on risk treatment-related concepts)

9.3 Secure TROPOS

Before describing the alignment of secure TROPOS with the ISSRM domain model, we need to investigate in further detail the syntax of secure TROPOS and how its concepts can be used when modelling. We first propose a metamodel of secure TROPOS. Then we build an example based on our metamodel. And finally, we present the alignment of secure TROPOS with the ISSRM domain model.

9.3.1 Secure TROPOS Metamodel

Literature does not really define the secure TROPOS metamodel. In order to design one that can help us during the alignment phase between secure TROPOS and the ISSRM domain model, we based on the metamodel presented in [Mou06]. It lacks of precise explanations of relationships and precise multiplicities and labels on the relationships. As secure TROPOS uses TROPOS concepts, we investigated the metamodel of TROPOS presented in [BGG$^+$04]. For sake of legibility, the complete metamodel has been cut into three parts: TROPOS metamodel specifying a) the actor concept Fig. 9.4, b) the goal concept Fig. 9.5 and c) the plan concept Fig. 9.6. Finally, we took into account the metamodel of the **G**oal-oriented **R**equirements **L**anguage (GRL) [HSDP06], a member of the i^* family of languages for which a metamodel had already been investigated by our supervisors.

Fig. 9.4 represents the concept of Actor and the Dependency relationship. The Depender and Dependee roles of the Dependency relationships are played by Actors. A Dependency also has a Dependum that can be either a Plan, a Resource or a Goal. The *why* link of the Dependency is presented as "*an optional reason for the dependency*" [BGG$^+$04]. Without any further information, the usefulness of the *why* link does not seem relevant to us. We just do not retain it. The TROPOS philosophy is to design a model by beginning with the elicitation of the actors' goals. This explains the existence of the *wants* association between actors and goals on the metamodel. The last concept still not discussed in Fig. 9.4 is Belief. A Belief is something that an actor takes for the truth from his point of view.

Fig. 9.5 illustrates the concept of Goal. A Goal can be an Hardgoal (usually called Goal) or a Softgoal. It can be decomposed depending on Decomposition relationships into a set of sub-goals. The notion of Means-ends analysis and Contribution as described in [BGG$^+$04] are quite different

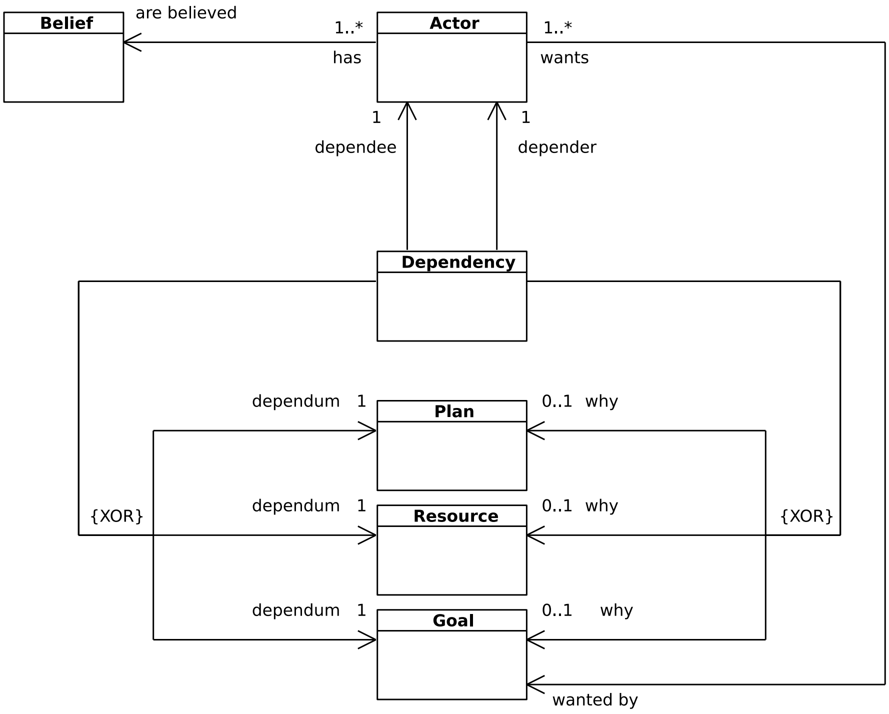


Figure 9.4: The UML class diagram specifying the actor concept in the TROPOS metamodel (adapted from [BGG$^+$04])

from the notion of Means-ends relationship and Contribution relationship as described in i^*, and we will discuss the difference in 9.3.1.

Fig. 9.6 focuses on the concept of Plan. A Plan fulfills a Goal on the behalf of an Actor. A Plan can be deeper analysed by using a AND-OR decomposition.

The secure TROPOS metamodel, presented in [Mou06] and depicted in Fig. 9.7, introduces five new concepts related to the security: a) the secure Goal, b) the secure Resource, c) the secure Plan, d) the security Constraint and e) the secure Dependency. As for Goals, a secure Goal is fulfilled by secure Plans and it contributes to achieve security Constraints. As different types of constraints can be introduced during the design of the system-to-be, for example performance or reliability constraints), a security Constraint is a specialization of the concept of Constraint and it restricts Dependencies. The same argumentation holds for secure Dependency related to Dependency concept. The three subtypes of secure Dependency: a) Double SD (Double **S**ecure **D**ependency), b) Depender SD and c) Dependee SD make possible to described the type of a secure Dependency (see Fig. 7.11).

The last metamodel we used to create our secure TROPOS metamodel is the GRL metamodel. GRL is the result of the "*integration of the i* goal-modelling language [Yu97] and the **N**on-**F**unctional **R**equirements (NFR) framework [MCN97]*". Fig. 9.8 describes the five types of GRL intentional elements – Softgoal, Resource, Task, Goal and Belief – and how they participate to Dependency, Correlation and Contribution relationships. Moreover, it details the five kinds of GRL intentional relationships: Contribution, Means-ends, Decomposition, Dependency and Correlation. Fig. 9.9 depicts how the GRL intentional elements participates to Means-ends and Decomposition relationships and goes in deeper details for Dependency, Correlation and Contribution relationships.

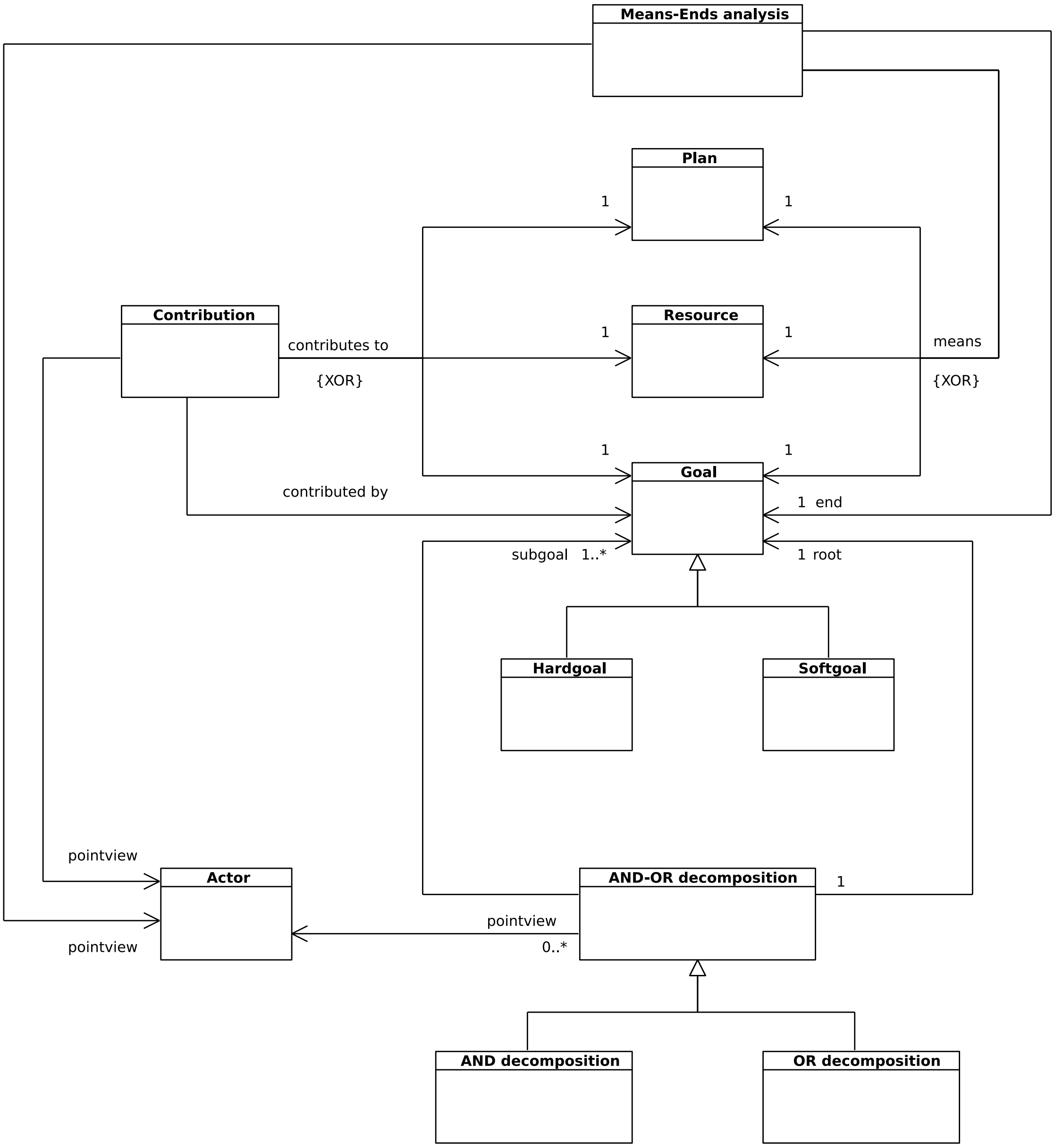


Figure 9.5: The UML class diagram specifying the goal concept in the TROPOS metamodel (adapted from [BGG^{+}04])

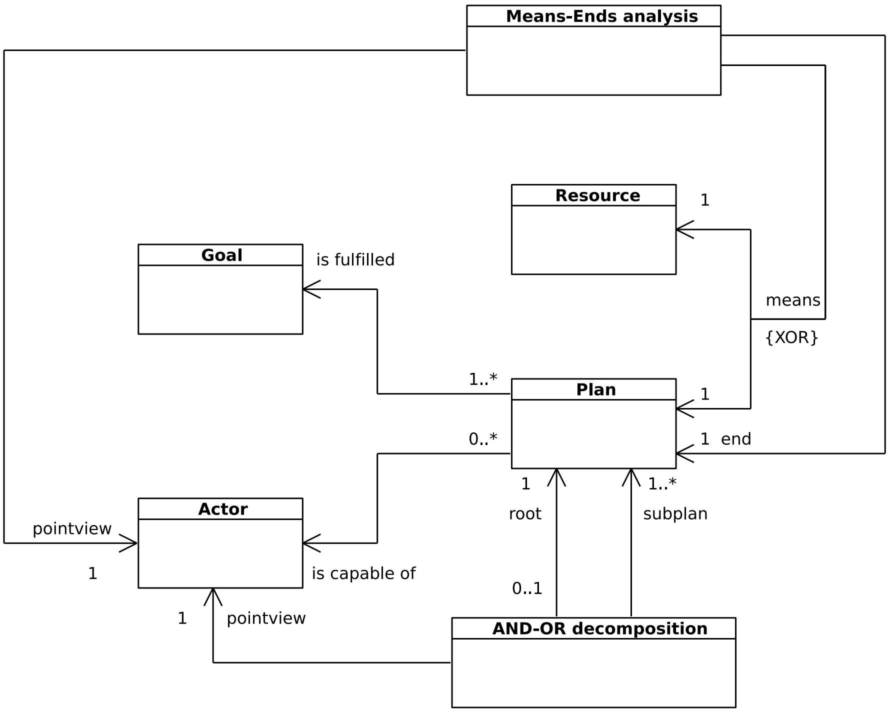


Figure 9.6: The UML class diagram specifying the Plan/Task concept in the TROPOS metamodel (adapted from [BGG$^+$04])

Combining the three previously cited metamodels (of TROPOS, secure TROPOS and GRL), we achieve a new one depicted in Fig. 9.10, Fig. 9.11, Fig. 9.12. We decompose it into three parts for sake of legibility.

The first part – Fig. 9.10 – zooms on the secure TROPOS concepts: a) Softgoal, b) Resource, c) Plan, d) Goal, e) Threat, f) Belief and g) security Constraint. The first four can be gathered as the common abstract concept of Possible Roles of the relationships described in Fig. 9.12. In order to include security concepts and as the security version of TROPOS concepts are used in the same relationships as normal (not secure) version, we add an attribute **isSecure** in the UML classes that represent concepts that can be seen as a secure version. This attribute is a String having cardinality [0..1]. Cardinality 1 makes the concept *secure*. The definitions of Softgoal, Resource, Plan, Goal and Belief come from [BGG$^+$04] and are kept without any kind of changes. The definitions of their secure versions, included the notion of security Constraint, can be found in [MGGP02, Mou06, MG04]. Concerning security Constraint, we choose to express the kind of constrained entity related to the security Constraint by adding two attributes Depender and Dependee. Having the value 1 for one of the two attributes is an exclusive situation. Indeed, a security constraint can be defined without any Depender or Dependee attributes. This is the case when security Constraint does not participate to a secure Dependency relationship. But when it participates to secure Dependency relationship, it can only be linked to the Depender or to the Dependee, not both at the same time. The Threat corresponds to the definition given in [MGGP02]. An Actor holds Possible Roles meaning that an Actor wants to satisficy a Softgoal, to achieve a Goal, to perform a Plan or to make a Resource available [MHO07]. An Actor has also security Constraints that he tries to fulfill and Beliefs that he has on the environment of the system-to-be. Contributee and Contributor are two abstract classes whose purpose is only to allow UML associations between the possible underlying concepts that can be Contributor and/or

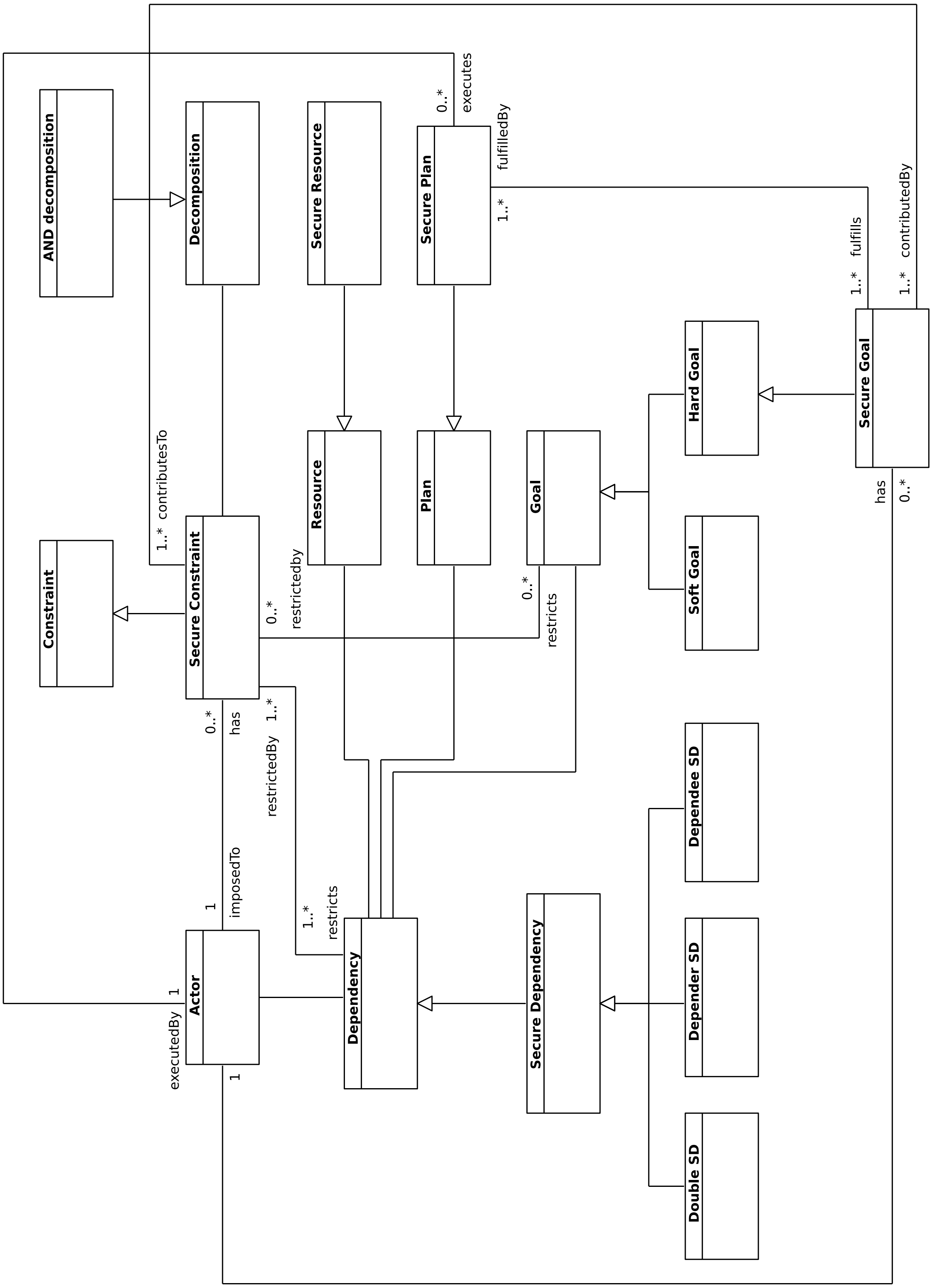


Figure 9.7: The secure TROPOS metamodel (adapted from [Mou06])

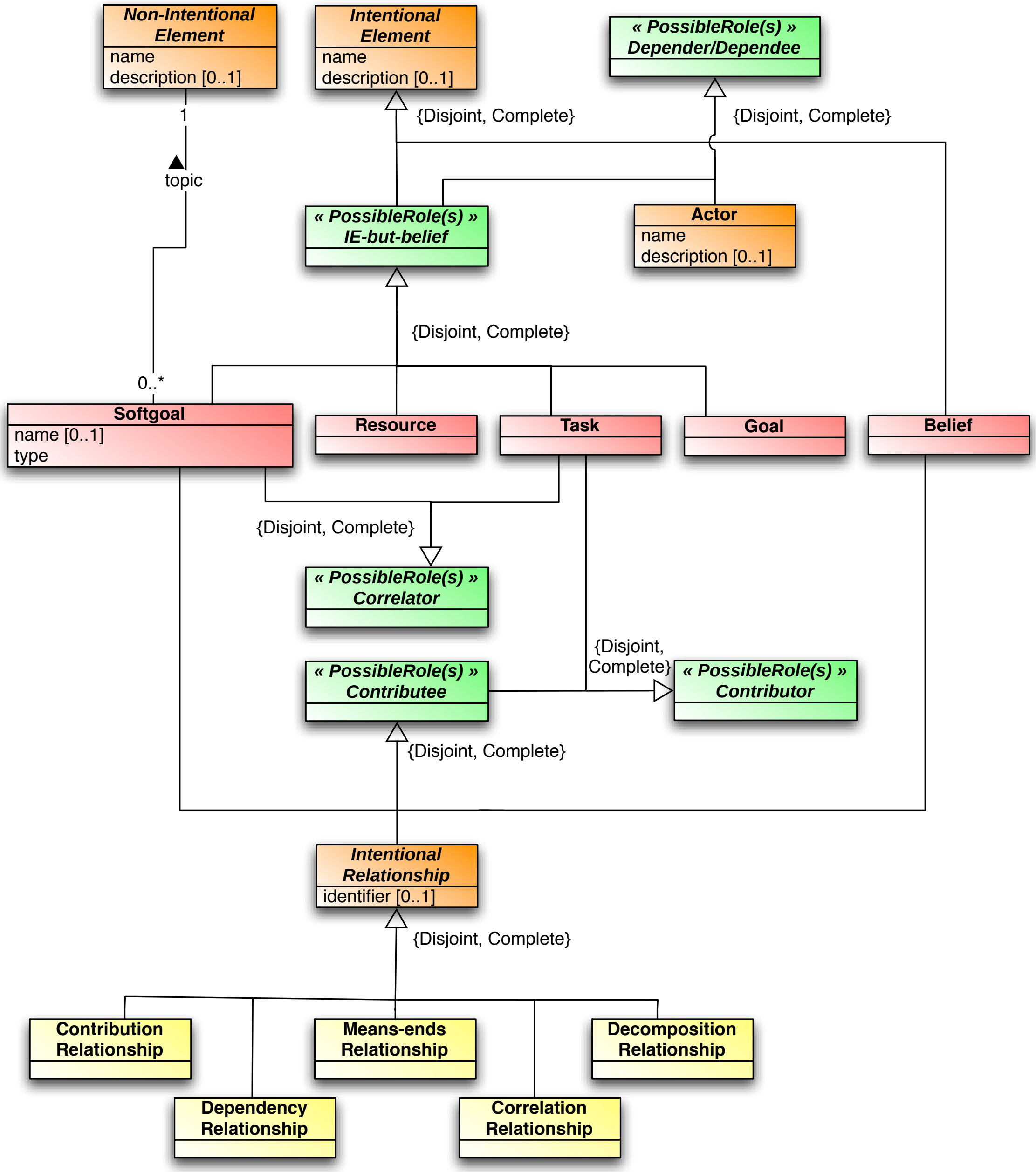


Figure 9.8: GRL metamodel: Zoom on intentional elements (from [HSDP06])

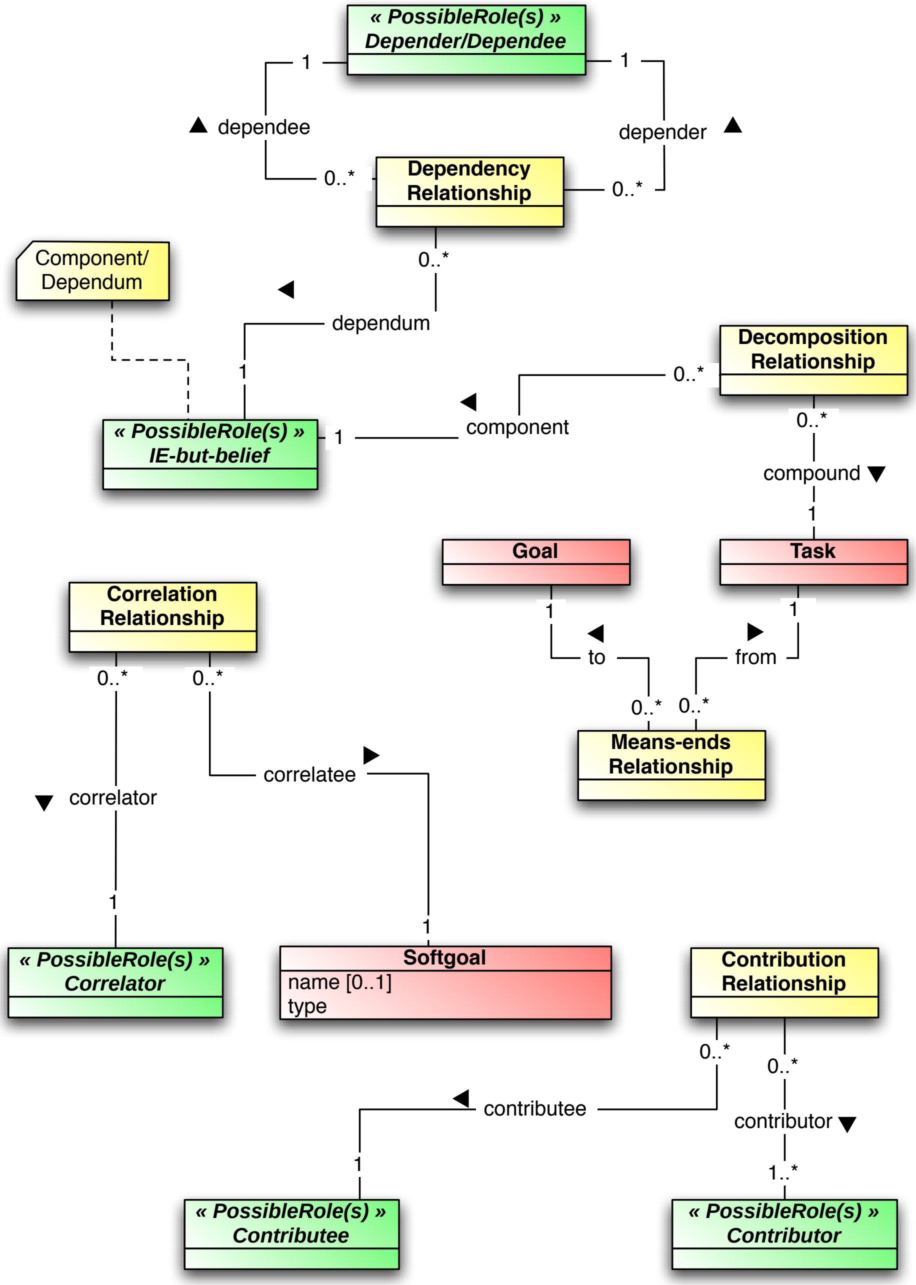


Figure 9.9: GRL metamodel: Zoom on intentional relationships (from [HSDP06])

Contributee. All types of Contributee can also be Contributor including Plan and Threat too. We consider that **only** negative Contribution relationship can link a Threat to Softgoals. The inheritance relationships of Depender/Depender, Contributee and Contributor are discussed when describing Fig. 9.12.

Fig. 9.11 goes into further detail relating to the secure TROPOS relationships. Secure TROPOS handles six types of relationships:

- Contribution: we define a contribution relationship as a relationship showing the impact of a Contributor on a Contributee [MHO07];
- Dependency: a dependency indicates "that one actor depends, for some reason, on the other in order to attain some goal, execute some plan, or deliver a resource." [BGG$^+$04];
- Means-ends: the Means-ends relationship specifies a means (represented by a Goal) to satisfy a Plan [KSR];
- Decomposition: the decomposition relationship defines the sub-components of a Plan [MHO07];
- Restricts: a restricts relationship expresses a restriction of a security Constraint on a Goal. A restriction is a kind of property that the Goal must respect.
- Attacks: an *attacks* relationship shows which is the target of an attacker's Plan. Note that our definition is quite different from the one cited in [MG04].

Fig. 9.12 zooms on the secure TROPOS relationships and particularly the authorized types of the concepts that they connect. We have already analysed the Depender and Dependee of a Dependency relationship. We focus now on its Dependum. The Dependum is the object around which the Dependency centers [BGG$^+$04, BPSMb]. The type of the Dependum can be one of the Possible Roles (see Fig. 9.10). A secure Dependency is a sub-type of Dependency. It is defined as a Dependency restricted by a security Constraint. A detailed definition is available in [MGGP02]. A secure Dependency can be constrained by several security Constraints that restrict either the Depender – that corresponds to a Depender SCs – or the Dependee – that corresponds to a Dependee SCs. We assume that a secure Dependency can have several Depender SC **and/or** several Dependee SC. Referring to [HSDP06], only a Plan can be decomposed into elements whose type belongs to Possible Roles. Note that in order to realise the Plan, all the sub-components needs to be achieved. To satisfy a Goal, several **alternative** Plans can be considered. These alternative means are expressed by a Means-ends relationships.

Characteristics of our Secure TROPOS Metamodel

As we presented previously, our secure TROPOS metamodel introduces some restrictions in comparison with the TROPOS and other secure TROPOS metamodels. The motivation for these restrictions is that we want to design a metamodel which, on the one hand, leaves as little as necessary in the interpretation of the language's semantics but, on another hand, will be powerful enough to model everything proposed in secure TROPOS [MGGP02, GMZ06, MG04, Mou06]. The results are that our metamodel is less inclined to be understood in different manners. Building a restrictive metamodel could be seen as an obstacle to an intuitive modelling. We discuss below how our metamodel can be used to model situations that are supported with other metamodel of (secure) TROPOS.

Contribution Relationship between a Goal and a Softgoal. If we consider the construction depicted in Fig. 9.13 in which a Goal positively contributes to a Softgoal (same reasoning for negative contribution), our metamodel can express this situation by adding a Plan as a means to

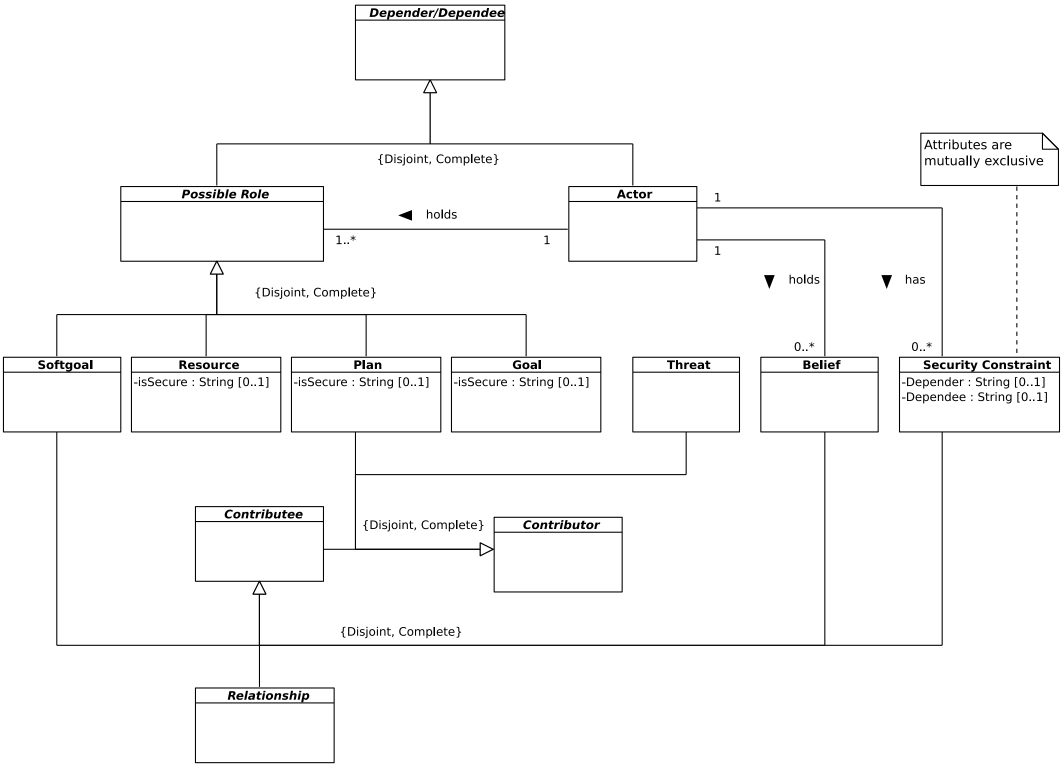


Figure 9.10: Our secure TROPOS metamodel: Zoom on secure TROPOS concepts

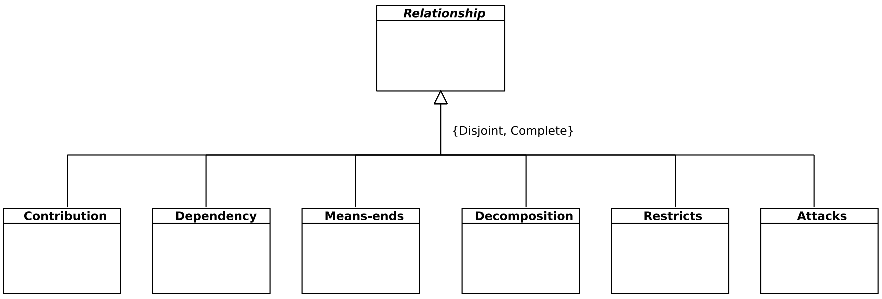


Figure 9.11: Our secure TROPOS metamodel: Zoom on the different types of secure TROPOS relationships

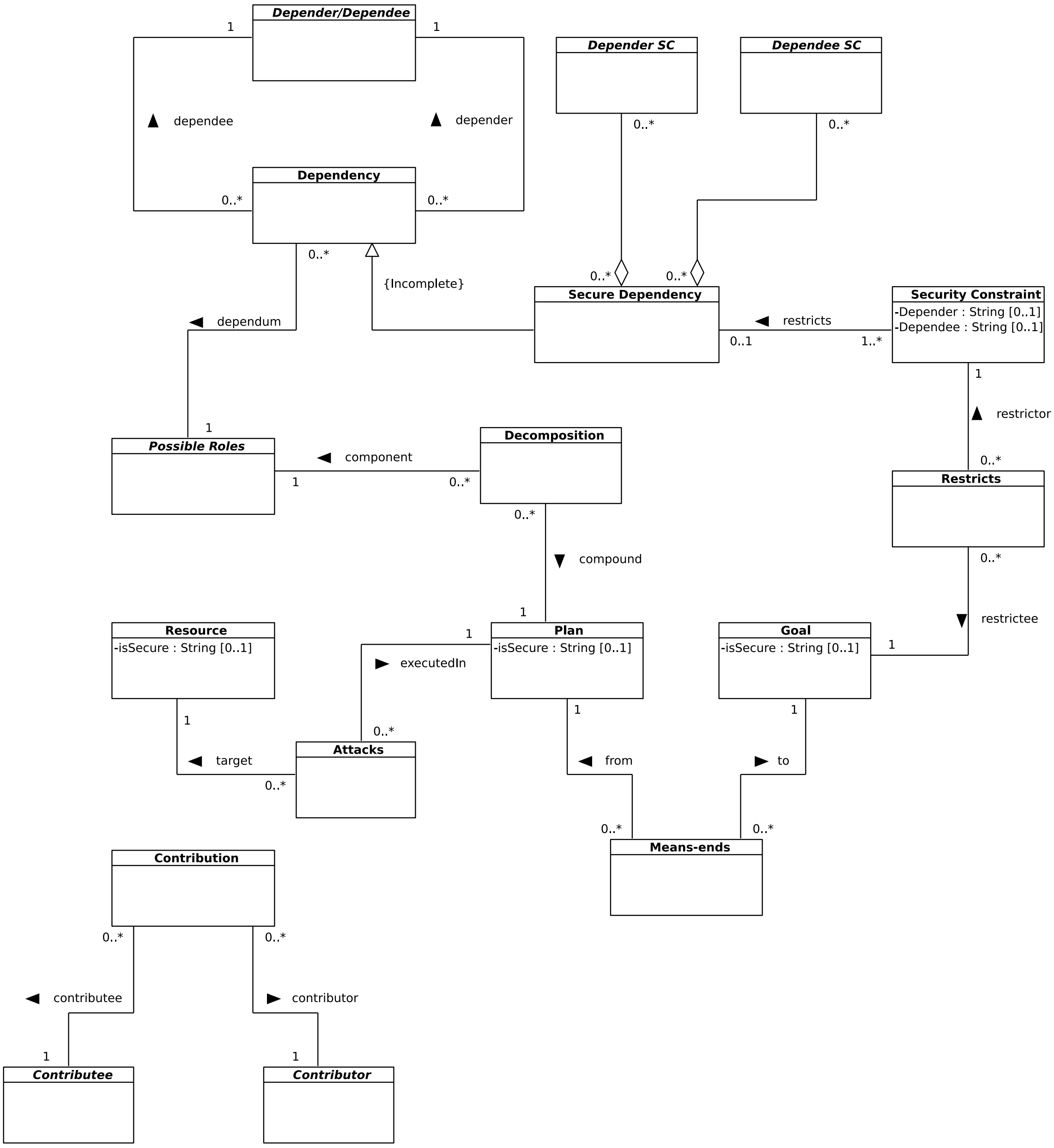


Figure 9.12: Our secure TROPOS metamodel: Zoom on the secure TROPOS relationships components

achieve the Goal and the Plan can contribute positively (or respectively negatively) to the Softgoal (see Fig. 9.14). Indeed, if a Plan **P**, that is a means to achieve a goal **G**, contributes (for example, positively) to a Softgoal **SG**, than we can consider that by achieving the Goal **G**, the Plan **P** contributes in a positive way to Softgoal **SG** and thus that the Goal **G** contributes to attain the Softgoal **SG**.

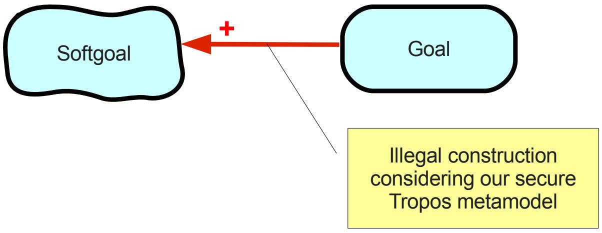

Softgoal
+
Goal
Illegal construction considering our secure Tropos metamodel

Figure 9.13: Illegal contribution of a Goal to a Softgoal

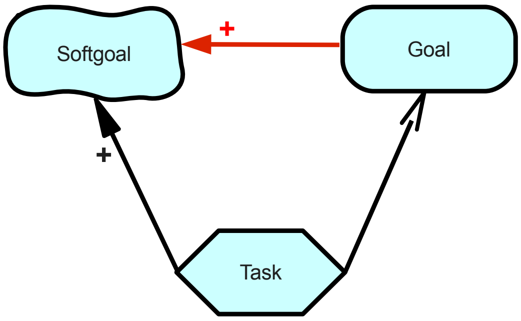


Figure 9.14: Underlying concepts used to express the contribution relationship from a Goal to a Softgoal using our secure TROPOS metamodel

Decomposition and Means-ends Relationships In TROPOS [BGG$^+$04], the following decomposition is legal (see Fig. 9.15). According to our secure TROPOS metamodel, only a Plan can be decomposed into sub components (which are Softgoal, Goal, Resource or Plan). So, as depicted in Fig. 9.16 in order to represent some sub-goals of a Goal, we first have to find a Plan that is a means to achieve the Goal. Then we can decompose the Plan into sub-goals. Even if this intermediate step can seem constraining, it can lead to investigate possible hidden sub-goals. Indeed, by defining the Plan, we can discover that this Plan can be subdivided into unelicited sub-goals. The decomposition between the Plan and the sub-goals is, from the point of view of our secure TROPOS, an AND Decomposition. All sub-goals need to be fulfilled in order to realise the Plan and thus to achieve the main Goal. We choose this interpretation for the decomposition since it permits to strictly refer to the metamodel whatever the type of the sub-components.

An underlying question is: "*how to represent a OR decomposition from a Goal into sub-goals according to our secure TROPOS metamodel*". The solution depends on a characteristic of the Means-ends relationships in the context of our metamodel. Our Means-ends relationship presents **alternatives** to attain a Goal. As for Decomposition, we restrict the meaning of the Means-ends in order to limit the number of basic concepts needed to understand a secure TROPOS model. So in order to express an OR decomposition or an AND means-ends, we have to use a set of Decomposition and Means-ends relationships. A brief example is given in Fig. 9.17.

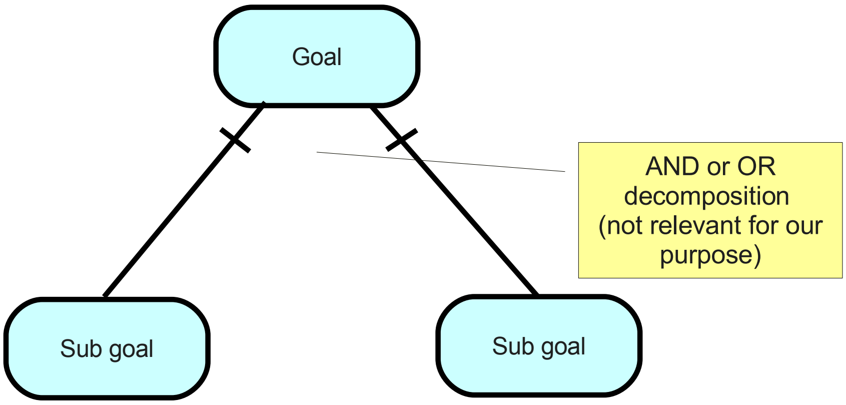


Figure 9.15: Example of the decomposition of a Goal into sub-goals following the TROPOS metamodel (AND or OR decomposition)

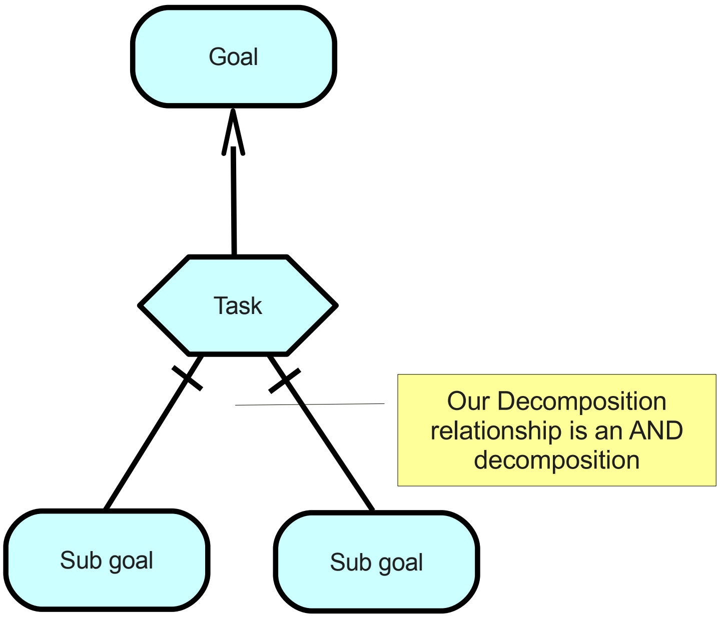


Figure 9.16: Expression of the decomposition of a Goal into sub-goals following our secure TROPOS metamodel

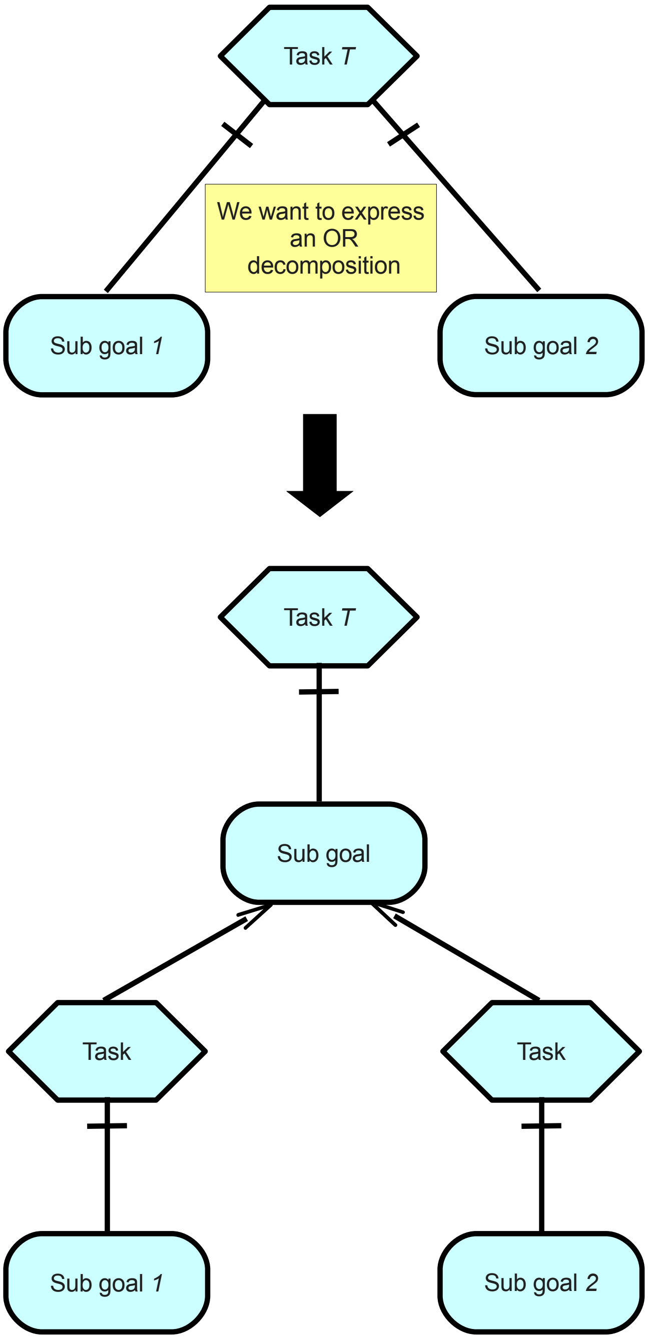


Figure 9.17: Example representing the expression of an OR decomposition following our secure TROPOS metamodel

Restricts Relationship. As depicted in Fig. 9.12, a security Constraint restricts a secure Dependency. In other words, it means that a security Constraint restricts the Dependum of the secure Dependency. On another hand, the Restricts relationship has only Goals as Restrictee. We made the choice to keep what should look as an inconsistency because a) it respects the literature [MG04] on this aspect and b) the Restricts relationship can indirectly restrict any concept of a type supported by the Possible Roles class. Indeed, using Decomposition and/or Means-ends, we can argue that the restriction affects the sub-components of the Decomposition and/or Means-ends relationships.

9.3.2 eSAP Example

We elaborated an example with our secure TROPOS in order to illustrate the alignment between the ISSRM concepts and secure TROPOS concepts. The alignment is described in 9.3.3. We based on the subject – the eSAP system – already discussed in [MGM] but we imagined new features required by the actors of the system-to-be. Moreover, we did not exactly follow the (secure) TROPOS method. We designed step by step the system-to-be, each step making a deeper investigation in the system's analysis. Our example takes place in the Health Care sector. The system-to-be is called the **e**lectronic **S**ingle **A**ssessment **P**rocess (eSAP) and is designed in order to deliver "*an integrated assessment of health and social care needs of*" patients [MGM]. One of the most important aspects to make the `eSAP` running is the `Patient Personal Information`. This information to be provided by the `Patient` is the first Goal elicited from `eSAP`. Following our secure TROPOS metamodel, the situation is designed as two Actors (`Patient` and `eSAP`) linked together by a Dependency relationship, as depicted in Fig. 9.18. The Depender is the `eSAP` system, the Dependee is the `Patient` and the Dependum is the Goal labelled `Information Provided`.

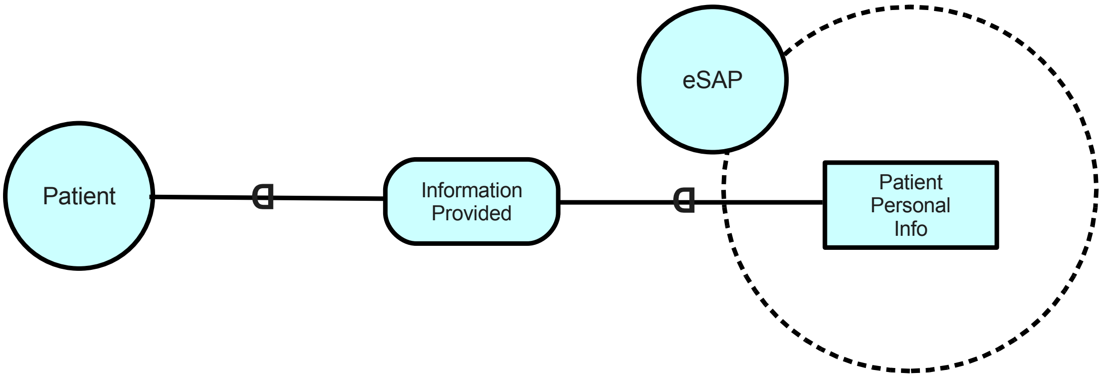


Figure 9.18: Elicitation of the dependencies between the `eSAP` system and a `Patient`

A `Social Worker` is in charge of the management of care plan of patients. In order to do his work, he needs the `Patient Personal Information`. In our modelling, the `Social Worker` has a Goal `Obtained Care Information` that depends on the `eSAP` system. However, as the `Patient Personal Information` is an valuable asset for the system, achieving the `Social Worker`'s Goal is submitted to a security property assuring that the `Patient's Consent` has been obtained before his personal information be sent to a third party. We model this constraint on the Social Worker's goal as a dependency between him and the eSAP system having as dependum the `Obtained Care Information` goal. We added a security Constraint on the behalf of the `eSAP` ensuring that the `Patient Personal Information` is shared only if his consent has been obtained. As this constraint has to be respected by the `eSAP`, the discussed Goal and constraint were placed within the `eSAP` actor. The *restricts* relationship permits to link the constraint to the targeted goal. The relationship between the `Social Worker` and the `eSAP` system is presented in Fig. 9.19.

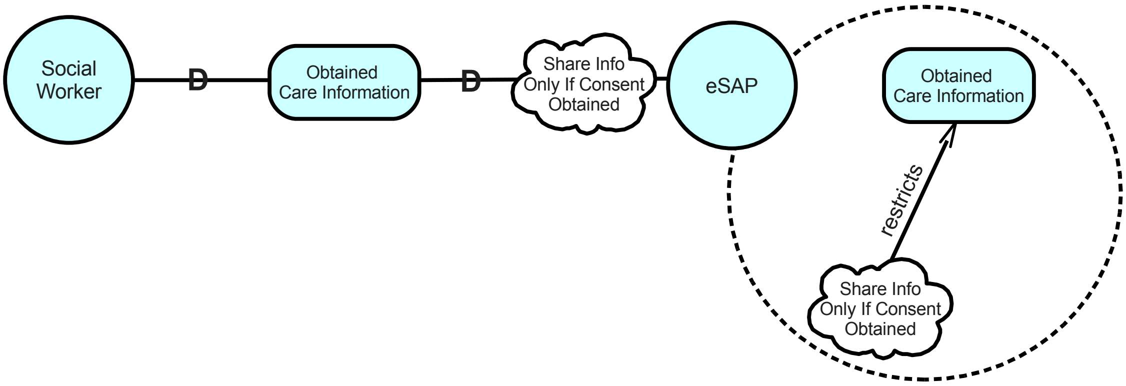

Social Worker
D
Obtained Care Information
D
Share Info Only If Consent Obtained
eSAP
Obtained Care Information
restricts
Share Info Only If Consent Obtained

Figure 9.19: Presentation of the relationship between the `Social Worker` and the `eSAP` system

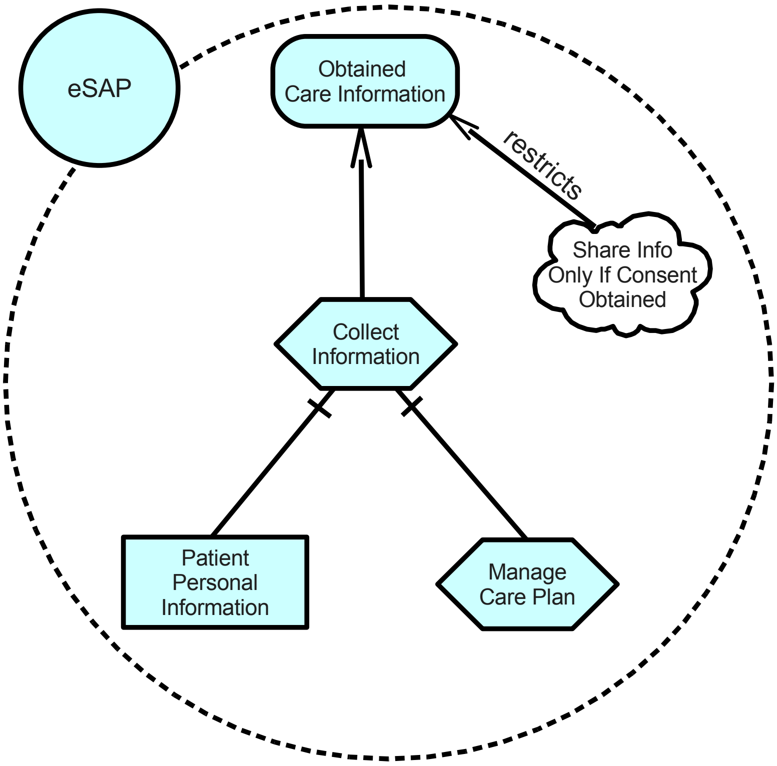

eSAP
Obtained Care Information
restricts
Share Info Only If Consent Obtained
Collect Information
Patient Personal Information
Manage Care Plan

Figure 9.20: `eSAP` system internal structure refinement

Once dependencies between Actors are elicited, we focus on the internal structure of the eSAP system, taking from starting point the Obtained Care Information Goal as it is under the responsibility of the eSAP. We try to refine the goal by expressing means to achieve it (see Fig. 9.20). In our case, we retain the Plan Collect Information. This Plan is the process needed to gather the Patient Personal Information. So in order to realise it, the eSAP system needs the Resource Patient Personal Information but also the Plan Manage Care Plan. Without this Plan, the eSAP system does not have the Patient's Care Plan under its responsibility, so it cannot collect Patient's information. The necessity of the sub-resource and the sub-plan is expressed using a Decomposition relationship. Following our secure TROPOS metamodel, this kind of relationship is a type of AND Decomposition, which corresponds to our depicted situation. As we are building the design of the system-to-be step by step, we keep concepts previously added: the security Constraint Share Info Only If Consent Obtained and the related *restricts* relationship.

Fig. 9.21 focuses on the possible risks to which the system-to-be is be exposed. As exhaustivity is not the objective of this example, we only consider the Threat Authentication Attack. Such a Threat attempts to let a threat agent pretend he is a trusted actor from the point of view of eSAP, in order to use privileges allowed to the fake identity to damage assets of the system. In the Health Care sector, a main risk is a breach in the privacy of the Patient Personal Information. So the risk Authentication Attack has a negative influence on the Privacy Softgoal. To take into account the privacy aspect in the eSAP system, we introduce a new security Constraint Keep System Data Privacy that has a positive influence on the privacy of the system and that can make the possible risk harder to realise. Note that the Authentication Attack is not placed as an internal concept of the eSAP. Indeed, the risk does not depend on the existence of the actor whose assets it threatens.

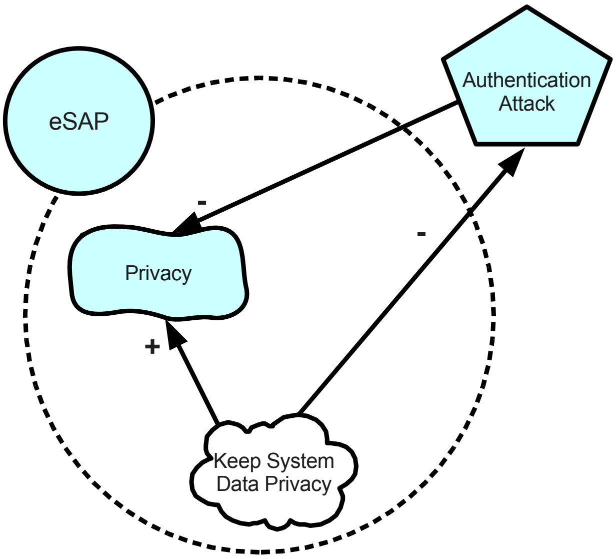


Figure 9.21: Investigation of the security risk against the eSAP system

We merge concepts added in Fig. 9.20 and Fig. 9.21 and elicited Plans aiming to respect the security Constraint Keep System Data Privacy (Fig. 9.22). The secure Plan Perform Authorization Checks is a means to ensure the system privacy (see System Privacy Ensured Goal in Fig. 9.22). To realise the Plan, the consent on the information for which the authorization will be checked has to be obtained. Otherwise, without access to the information, it is not possible to check if authorization rights are respected. We represent it under a Goal related to security. The second element to successfully perform in order to attain the secure Plan `Perform Authorization Checks` is the `Plan Check Authentication` (which is related to the security depicted with a **(S)** tag). The authorization on an element aims to check if an actor (whose identity is checked by the `Check Authentication` Plan) has appropriate rights on a resource (rights are represented in our case by the obtained consent) to perform an action. The security Constraint `Share Info Only If Consent obtained` has a positive contributor that is a Plan labelled `Check Data For Consent`. As this Plan is positively related to a security Constraint, it is also realised with respect to security concerns. Moreover, this Plan is a means to achieve the secure Goal `Consent has been obtained` previously described.

Once `eSAP` internal elements have been elicited, we can investigate the point of view of a possible threat agent (see Fig. 9.23). We choose to describe an attacker whose aim is to steal the `Patient Personal Information`. To achieve his goal, he needs to execute an attack (depicted as a Plan). Two sub-elements are required in order to fulfill the attack: 1) the attacker has to get the consent of the `Patient` on the desired Information (the Goal `Consent on Data obtained`) and 2) he needs to find the authentication code associated to the targeted Information (the Plan `Check eSAP access Repeatedly`). To get the consent, two alternatives have been retained: a) to `Steal the data from a Social Worker` or b) to `Buy the data from an untrusted Social Worker`. These two alternatives are depicted as Plans. The `Check eSAP access repeatedly` clearly exploits a vulnerability of the `eSAP` system that is known by the Attacker. The concept used to model it is a Belief that positively contributes to the Decomposition relationship related to the `Check eSAP access repeatedly`. The Resource targeted by the attack is the `Patient Personal Information` on behalf of the `eSAP` system. We linked the targeted Resource to the attack scenario by an *attacks* relationship as defined in our secure TROPOS metamodel. The entire Fig. 9.23 can be conceptually seen as the refinement of the risk discussed in Fig. 9.20 and Fig. 9.21.

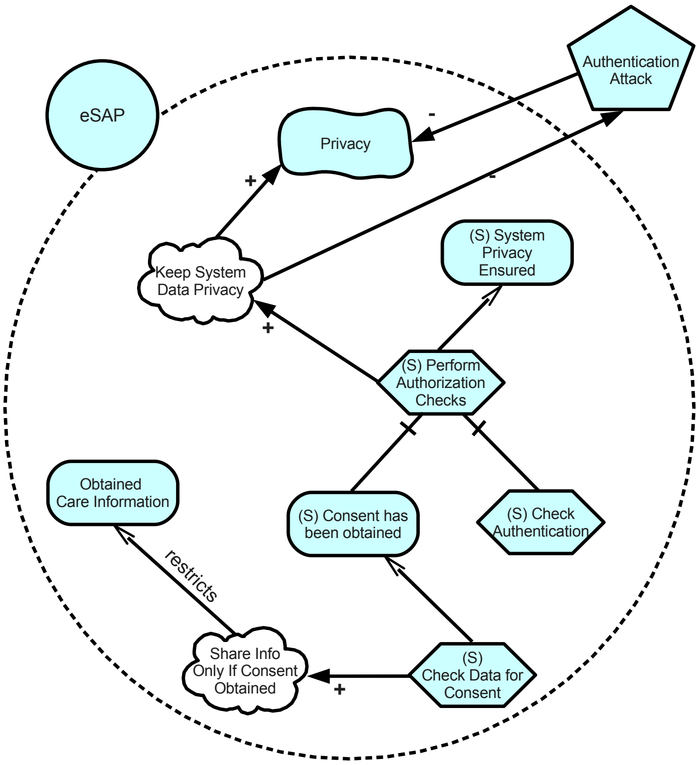

eSAP
Authentication Attack
-
Privacy
+
-
Keep System Data Privacy
(S) System Privacy Ensured
+
(S) Perform Authorization Checks
Obtained Care Information
(S) Consent has been obtained
(S) Check Authentication
restricts
Share Info Only If Consent Obtained
+
(S) Check Data for Consent

Figure 9.22: Merging of all elicited `eSAP` internal elements

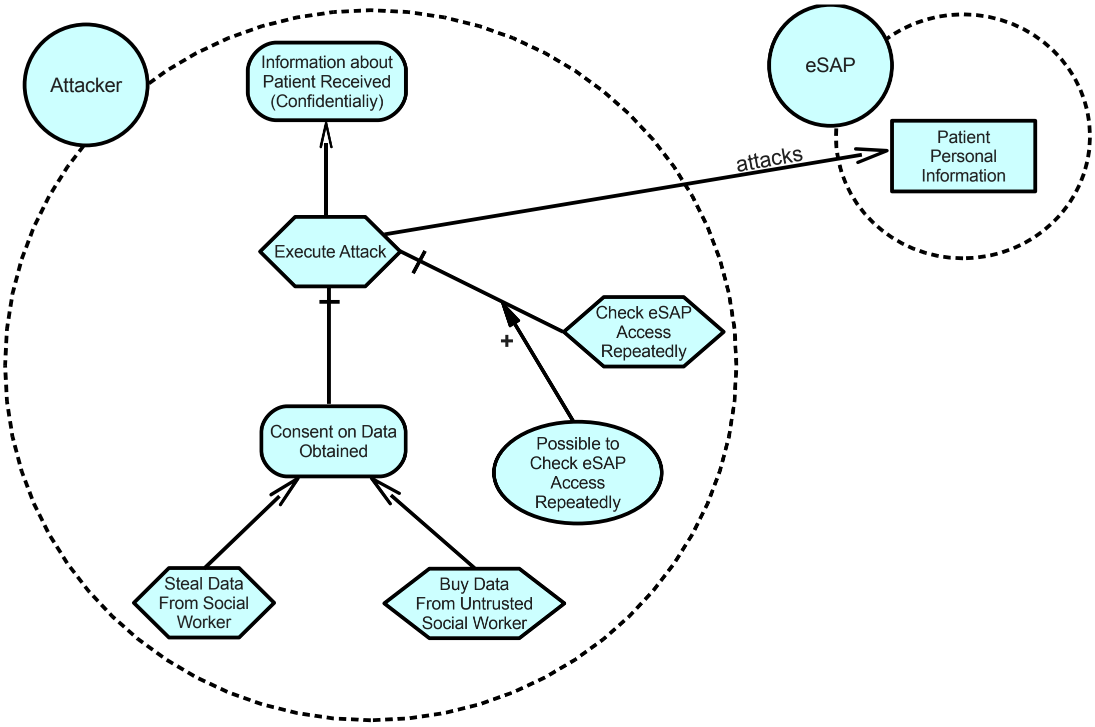

Attacker
Information about Patient Received (Confidentialiy)
eSAP
attacks
Patient Personal Information
Execute Attack
Check eSAP Access Repeatedly
+
Consent on Data Obtained
Possible to Check eSAP Access Repeatedly
Steal Data From Social Worker
Buy Data From Untrusted Social Worker

Figure 9.23: Internal structure of an attack against the Resource on the behalf of the `eSAP` system

The last step of the design of our example is the elicitation of the countermeasures that help to mitigate the risk identified in Fig. 9.22 and its possible exploitation by an attacker (see Fig. 9.23). From the model depicted in Fig. 9.22, we try to find an alternative means to achieve the Goal `System Privacy Ensured`. A solution is to `Perform Cryptographic Procedures`. We add a **(S)** tag in order to express that it is a newly introduced security countermeasure. This way, even if the attacker manages to steal the `Patient Personal Information` Resource, he cannot use it since he does not know the cryptographic key. Thus the privacy of the data of the system should still be ensured. To fulfill this countermeasure Plan, three sub-elements are needed: 1) the consent on the data to protect, 2) to encrypt data when they are sent to a third party and 3) to decrypt data when received from a third party. Combining these three elements, we can perform cryptographic procedures that help to ensure system privacy and thus that help to satisficy the privacy Softgoal.

We stop our example at this point of the design of the system-to-be. A complete process from early requirements to the implementation of the system should normally require to iterate several times on the described step, until the envisioned risks are considered acceptable or disappeared. So, if we have to iterate on our example, the newly introduced countermeasures will become elements of the eSAP system and the possible risks investigation will have to evaluate the vulnerabilities linked to these new elements.

9.3.3 Alignment of Secure TROPOS and the ISSRM Domain Model

In Table 9.3 we present how the secure TROPOS covers the ISSRM domain model.

Asset-related concepts: secure TROPOS attempts to cover the entire development process of an IS, from the early requirements to the design phase. This security modelling language is mainly focused on the elicitation of Goals that the different Actors of the system-to-be want to achieve. It takes into account security concerns by eliciting in terms of Threat (secure TROPOS

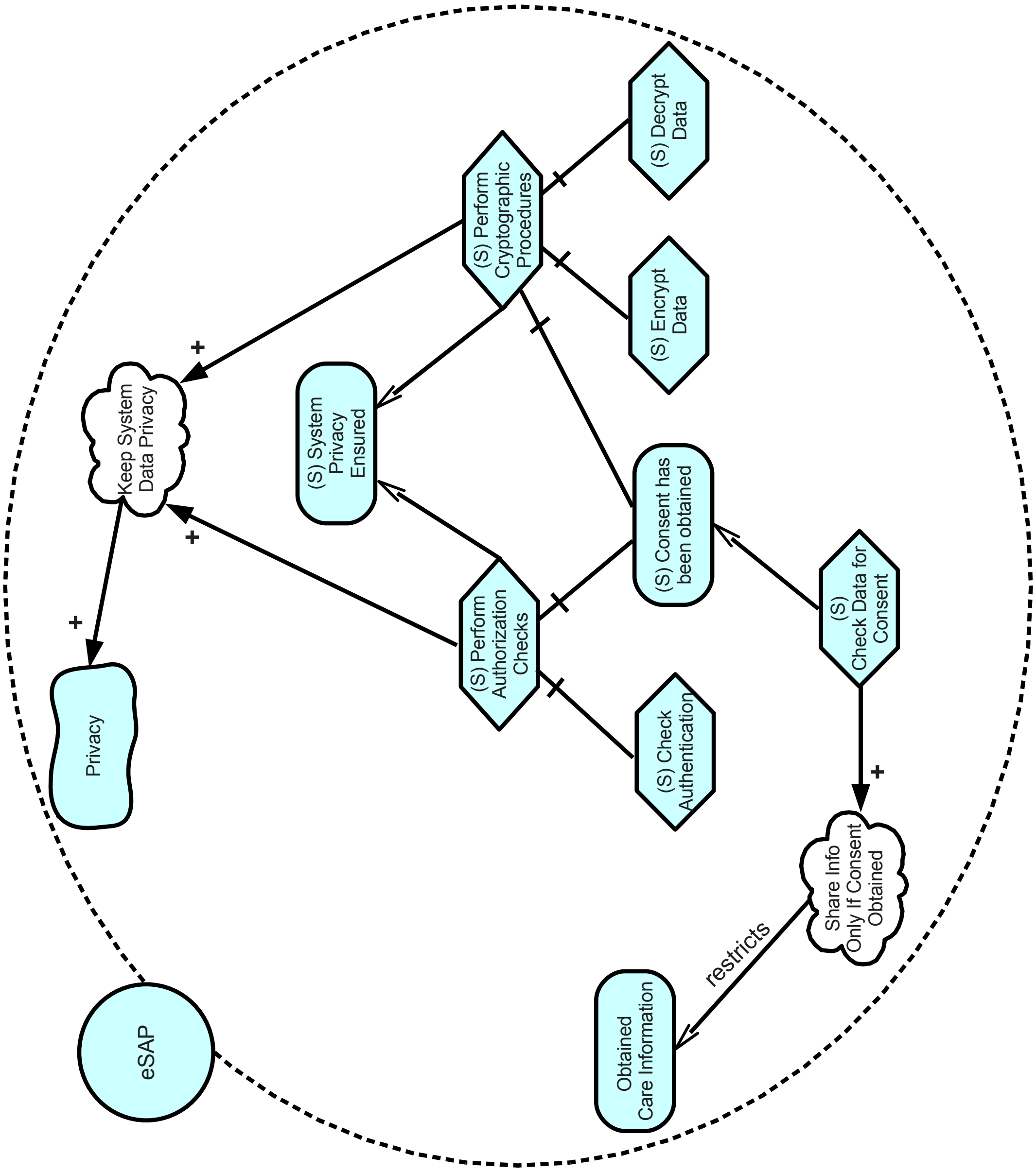

(S) Decrypt Data
(S) Perform Cryptographic Procedures
(S) Encrypt Data
+
Keep System Data Privacy
(S) System Privacy Ensured
(S) Consent has been obtained
+
(S) Check Data for Consent
(S) Perform Authorization Checks
+
Privacy
(S) Check Authentication
+
Share Info Only If Consent Obtained
restricts
eSAP
Obtained Care Information

Figure 9.24: Refinement of countermeasures to mitigate revealed risks

concept) the possible risks against the system and by constructing in terms of Goals the objectives of potential attackers. If we begin by considering the ISSRM concept of *security criteria*, we can find two secure TROPOS concepts corresponding to it: a) the security Feature and b) the security Constraint.

A security Feature (also called Protection Property) is described in [MGGP02] as a "*feature associated to security that the system-to-be must have*". "*Examples of security features are privacy, safety, accountability, availability and integrity*". This definition covers the semantics of the ISSRM concept of *security criteria* given in [MHM07]: "*properties or constraints on business assets characterising their security needs*". Moreover, [MHM07] gives, as examples of security criteria, terms like confidentiality, integrity and availability. The second concept, security Constraint, is defined in [MGGP02]: "constraint that is related to the security of the system". An example of a security Constraint can be found in our example in Fig. 9.19, represented with a white cloud shape and labelled `Share Info Only If Consent Obtained`. It appears that Softgoals that have a label specific to security (for example: privacy) can be considered as security criteria. But we have to be aware that such Softgoals represent a higher level of security criteria than security Constraints. This is mainly reflected in the label of the two kinds of concepts. Moreover, a Softgoal used as security criterion can only be used on a SRM, so it cannot be placed on secure dependency relationships.

Once the alignment of ISSRM security criteria is done, we can investigate the notion of Asset. A relevant clue is given in ISSRM definition of security criteria: "*properties or constraints* ***on*** *business assets*". As shown in the ISSRM domain model [MHM07], there is an association relationship between security criteria and business assets. Secure TROPOS expresses such a constraint using a secure Dependency. A security Constraint can be an element of a secure Dependency that constrains the Depender or Dependee (as described in 7.2.2 on the achievement of the Dependum). The Dependency relationship links two Actors or Possible Roles, meaning that one needs either a Goal, either a Softgoal, either a Plan or either a Resource from another one. Hence, each Dependum linked with the analysed organisation can be considered as an asset. As said in the definition of asset from ISSRM domain model, "*asset is anything that has value to the organisation and necessary for achieving its objectives*" [MHM07]. The definition of the Dependency relationship has the idea of the necessary element to achieve an objective: "*dependency between two actors indicates that one actor (the Dependee)* ***depends*** *for some reason on another one (the Depender)* ***in order to achieve a goal, to execute a task, to deliver a resource***" [MHM07].

We cannot give a clear criterion to distinguish Business assets from IS assets. In fact, the type of an asset depends on the context in which we consider this asset. If an asset supports another asset and is a part of the system, it is possible for this asset to be categorized as an IS asset. However, we can define that an asset used as Dependum in a Dependency relationship is a Business asset. Another tricky problem is the limitation of the secure TROPOS metamodel which does not authorize a Dependency relationship having an Actor as Dependum. This situation can be resolved by representing, in secure TROPOS, the unauthorized relationship using as a combination of different Dependums of Dependency relationships. Hence, we can add the concept of Actor to the alignment with the ISSRM concept of asset. On the other hand, Business asset represents "*information or process inherent to the business of the organisation that has value to the organisation and is necessary for achieving its objectives*" [MHM07]. The secure TROPOS notion of Resource can be aligned with the term **information** in Business asset definition. We align secure TROPOS Plan with *process*. Concerning the notion of Goal and Softgoal that *should be able to be* considered as Business assets, we propose the following explanation to achieve the alignment. Goals and Softgoals are neither information nor processes. However, we can consider that a (soft)Goal can be achieved by a **combination** of Resources and Plans in the internal environment of the Actor responsible of the Goal. An example is given in Fig. 9.20 where the Goal Dependum `Obtained Care Information` can be attained by a Plan `Collect Information` decomposed into a Resource `Patient Personal Information` and another sub-plan `Manage Care Plan`.

As we can see, it is possible to achieve quite a good alignment of secure TROPOS asset-related concepts with those from the ISSRM domain model. However, an issue is still open: following the ISSRM domain model, only Business assets can be constrained by security criteria. But, following our proposed distinction between IS assets and Business assets, it seems that secure TROPOS offers the possibility to constrain IS assets too. A possible explanation is that this possibility expresses the combination of the ISSRM *constraint of* association with the *supports* association, linking thus *IS assets* to *Security Criteria* (see ISSRM domain model [MHM07]).

Risk-related concepts: in [MG04] a new sub-activity, called *testing the developed solution*, is inserted in the secure TROPOS methodology. This activity permits to the developers to test how their solution copes with potential attacks. The analysis of potential attacks is based on **S**ecurity **A**ttack **S**cenario (SAS). A SAS is defined in [DvLF93, MG07] as "*an attack situation describing the agents of a multiagent system and their secure capabilities as well as possible attacks*". It involves potential attacks to a multiagent system, a possible attacker, the resources he is attacking and the agents of the system related to the attack. We find, in this definition, some of the risk-related concepts such as *threat agent*, *attack method*, *assets*.

We did not build a SAS in our example but we used its concepts to design an attacker and his internal structure that lead to a risk against the system-to-be. We did not consider how actors can mitigate risks directly on the attacker model. We include such countermeasures in the internal structure of the `eSAP` system. In our example, the `Attacker` wants to steal `Patient Personal Information` using an attack method based on the knowledge of a vulnerability in the check authentication procedure that allows to test repeatedly several solutions to match the Patient authentication resource. Further details have already been described and can be seen in Fig. 9.23.

An ISSRM threat agent is called an attacker in secure TROPOS terminology. An attacker is described as "*an agent who aims to break the security of the system"* [MG04]. The ISSRM threat agent definition speaks about "*an agent that can potentially cause harm to assets of the IS*". The secure TROPOS notion of Agent is a kind of actor and this concept of Actor can be considered as related to the ISSRM notion of Agent. Moreover, in each definition, the concept of harm against assets is present. However, in order not to too restrictive with the secure TROPOS notion of Actor, we choose to consider, as an ISSRM threat agent, not only a secure TROPOS Agent but its concept of Actor.

The ISSRM concept of attack method is composed of the secure TROPOS concepts of Plan (seen from the point of view of an attacker) and the secure TROPOS concept of attack relationship. This attack relationship is depicted as a link containing an "attacks tag" (see Fig. 9.23) and it starts from one of the attacker's Plans and ends at the attacked Resource. We consider the Plan and the *attacks* link as equivalent to the ISSRM attack method because the Plan expresses the way the attacker tends to harm assets and the *attacks* link designates the targeted Resource. In our example, the attack method is represented by the Plan `Execute attack` which targets the `Patient Personal Information` in the `eSAP`. It seems that there is an inconsistency in [MG04, MG07]. The *attacks* link does not start from a Goal (as defined in [MG04]) but from a Plan and some of the attacked Resources are Agents. If we make further investigation, we can propose an explanation to the inconsistency. For this discussion, we base on our secure TROPOS metamodel. The Decomposition link between the Goal `Modify Content of Message` and the Plan `Cryptographic Attack` is an instance of a Decomposition relationship. In Fig. 9.12, the Decomposition relationship has a relationship with the Possible Roles called *component*. It means that an instance of a Decomposition relationship has a component of one and only one type of Possible Roles. On another hand, we see that a Possible Rols is specialized into either a Softgoal, a Resource, a Plan or a Goal. Thanks to this explanation, we can validate the Goal participating in the Decomposition link. The usage of a Plan in the Decomposition is proved by the existence of the association called *compound* between the Decomposition relationship class and the Plan.

The non-existence of the inconsistency becomes clear when we investigate the Means-ends relationships in our secure TROPOS metamodel, which has two associations: one with the Goal class, the second with the Plan class. We see that an instance of the Plan class expresses the means-ends used to fulfill an instance of the Goal class. Thus, the *attacks* link between a Resource and a Plan should be understood as a link between a Resource and a Goal if the previous Plan is a component of the decomposition of the Goal. A second inconsistency appears between the term Resource in the definition of the *attacks* link and the example ([MG04]) where *attacks* links are attached to Agent (secure TROPOS meaning). As written in [MG04], "*the second sub-goal indicates the attacker trying to change the values in data files of the system. The fulfillment of this Goal can be satisfied by means of changing data of resources in the eSAP system*" . One suggestion is that the Resource (secure TROPOS meaning) is not mentioned for sake of brevity.

The ISSRM concept of Vulnerability has no explicit counterpart in secure TROPOS defined in [GMZ06] [MG04]. However we think that this concept can be expressed by the TROPOS syntax. Indeed, TROPOS contains the notion of Belief that is described as *the actor knowledge of the world*. An attacker needs to be aware of characteristics of assets that can constitute flaws in terms of security before he can try to threaten these assets. Our last sentence fits the definition of ISSRM Vulnerability. The attacker in our example bases his attack on the fact (from his point of view) that the check authentication procedure can be repeated infinitely. However, a vulnerability can be expressed by a secure TROPOS Belief but the two notions are not exactly covering the same space of knowledge. Trying to reformulate the definition of Belief, we want to draw attention on the fact that a Belief is something that an attacker takes for real. For example, in Fig. 9.23, the Attacker knows that it is `PossibleToCheckeSAPAccessRepeatedly` While a vulnerability is a fact, something that can be taken for sure in the environment. So, by modelling a vulnerability as a Belief, we introduce a notion of possible mistake in the knowledge of the attacker.

This means that there might be vulnerabilities that are not known by the actor. Also, not all the actor's knowledge about the world is correct. Thus to make alignment between Belief and Vulnerability we should consider the intersection between these two definitions: i.e, only the facts known by the actor, that are true.

Secure TROPOS has no precise notation for the notion of Threat as defined in the ISSRM. An ISSRM Threat is expressed in secure TROPOS as the hierarchy of attacker's Goals, from the unique highest Goal to its decomposition. By representing an ISSRM Threat in this way, we achieve to align on its definition: "*a potential attack or incident, which targets one or more IS assets that may lead to harm to assets*". Indeed, a Goal is the expression of the wish of an Actor and a wish is, by definition, something to be attained. Hence, it is something still remaining in the domain of the potentiality. An attacker's wish against a system is generally something that aims to damage the assets of the system. The Goal `Obtained Care Information` is the root attacker's Goal in our example. The target of the Threat seems to not be present in secure TROPOS. But we have already explained that a *attacks* link that starts from a Plan can be understood as it starts from the Goal that is fulfilled by previously cited Plan. Referring to our example, the target of the Threat is the `Patient Personal Information` Resource.

The notion of *cause of the risk* is represented by a secure TROPOS Threat. The Threat is defined in [MGGP02] as "*circumstances that have the potential to **cause** loss or problems that can put in danger the security objectives of the system*". The term of cause is a clue in the process that we used to align the secure TROPOS Threat with the ISSRM Cause of the risk. The second reason is depicted in Fig. 9.21. The Threat is linked with assets of the system by a negative Contribution relationship that we can consider as ISSRM Impact (discussed below). Moreover, we can understand the concept of Threat as composed of Threat agent, Attack method, Threat and Vulnerabilities as presented in Fig. 9.25[1].

[1] This example contains some constructs that do not belong to secure TROPOS syntax but they are used for

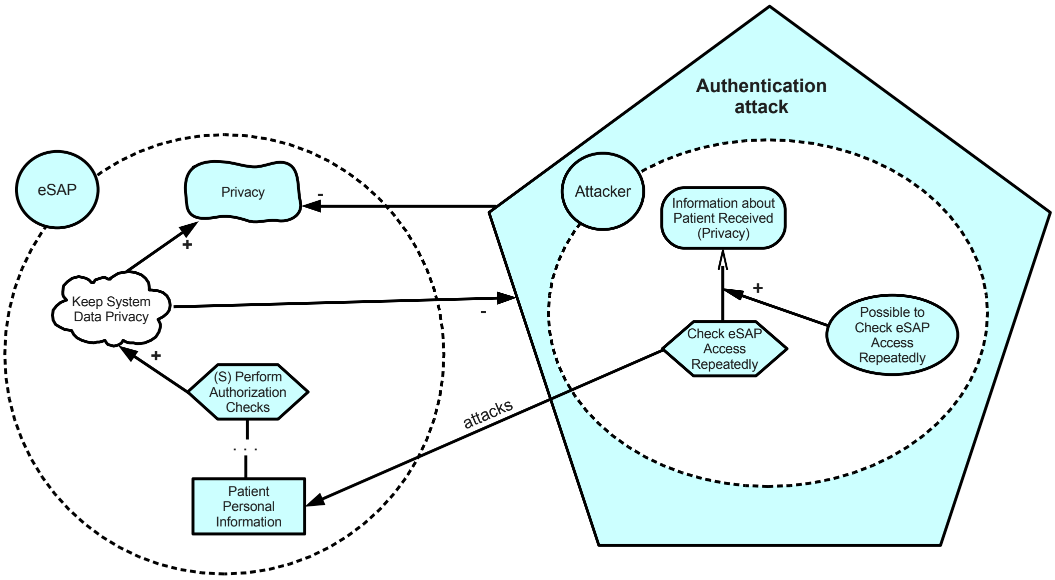

Authentication attack
eSAP
Privacy
-
+
Keep System Data Privacy
-
+
(S) Perform Authorization Checks
. . .
Patient Personal Information
attacks
Attacker
Information about Patient Received (Privacy)
+
Check eSAP Access Repeatedly
Possible to Check eSAP Access Repeatedly

Figure 9.25: Mental representation of a secure TROPOS Threat as including the Attacker actor and his internal structure

The ISSRM concept of *Impact* is represented, as said before, by a negative Contribution relationship from a secure TROPOS Threat to a Softgoal related to security. This corresponds to the definition of the ISSRM Impact: "*the impact can be described ... at the level of business assets, where it* ***negates security criteria***".

The last ISSRM concept related to the risk is the concept of risk itself. There is no secure TROPOS concept/construct to express an ISSRM risk.

Risk Treatment-related concepts: as we did in our example, once the attacker's Goals have been investigated, it is necessary to find and include in the system-to-be countermeasures to the risks in order to mitigate them. In our case, we introduced the Plan `Perform Cryptographic Procedures` to make information unusable even if it was stolen. It fits the ISSRM security requirement defined in these words: "*the refinement of the risk treatment decision to mitigate the risk. Each security requirement contributes to cover one or more risk treatment decision for the target IS*". The ISSRM concept of control can be seen as the system in itself once it will be ready for implementation. The risk treatment decision is the process leading to the introduction of the countermeasures. So even if they do not have specific secure TROPOS concepts, these two concepts are implicitly present in secure TROPOS models. Moreover, as we discussed before, when countermeasures have been elicited, a newly iteration on the model of the system-to-be must be performed to discover if the new introduced countermeasures – that are at this time considered as assets of the system – introduce new security vulnerabilities which are unacceptable for the system. This iterative process ends when all risks are evaluated as acceptable depending some criteria (for example: time necessary to improve the system, cost of the improvement or also cost of the damage if an attack occurs). Note that each time a modification is made to the system, a risk analysis is required.

sake of legibility of the model. Moreover, the secure TROPOS Threat cannot be designed with another concept within its boundary

ISSRM domain model		**Secure TROPOS**		
		Synonyms	**Language concept**	**Element from our example**
Asset-related concepts	Asset	–	Actor, Goal, Softgoal, Plan and Resource[B]	–
	Business asset	–		Obtained Care Information, Collect Information, Manage Care Plan, Privacy, Patient Personal Information
	IS asset	–		Perform Authorization Checks, Consent has been obtained, Check Authentication, System Privacy Ensured
	Security criteria	Security Feature, Protection Property[A]	Security Constraint and Softgoal	Share Info only if Consent Obtained, Keep System Data Privacy
Risk-related concepts	Risk	–	–	–
	Impact	–	Negative contribution relationship	–
	Cause of the risk	–	Threat	Authentication Attack
	Threat	–	Goals + Plans	Attack eSAP System + Corrupt Available Dates + Data change Attack
	Vulnerability	–	Belief[C]	Possible to check authentication repeatedly
	Threat agent	Attacker	Actor	Attacker
	Attack method	–	Plan + *attacks* relationship	Execute attack and Steal Data From Social worker or Buy Data from Untrusted Social Worker
Risk treat-ment -related concepts	Risk treatment decision	–	–	–
	Security requirements	(Secure) Goal, Plan, Resource and Protection Objective[A]	Actor, Goal, Softgoal, Plan, Resource[B]	Perform Cryptographic Procedures and Encrypt Data and Decrypt Data
	Control	[D]	–	–

Table 9.3: Alignment between secure TROPOS and the ISSRM domain model

^{A} Protection Objective and Protection Property are described in [MGGP02].

^{B} Constructs from TROPOS (not from secure TROPOS). Constructs includes: Actor, Goal, Softgoal, Plan, Resource and Relationships. Relationships are Dependency, Means-ends, Decomposition and Contribution.

^{C} A Belief cannot be strictly aligned with the ISSRM vulnerability concept. To explain it, let us consider theses definitions: Belief – the actor knowledge of the world [BGG$^+$04]. ISSRM vulnerability – characteristic of an IS asset or group of IS assets that can constitute a weakness or a flaw in terms of IS security [MHM07]. This means that there might be vulnerabilities that are not known to the actor. Additionally, not all the actor's knowledge about the world is correct. Thus to align between Belief and Vulnerability we should consider the intersection between these two definitions: i.e, only the facts known by the actor, that are true.

^{D} There is no construct in secure TROPOS that refers to the ISSRM control. But a subset of concepts composed of the countermeasures against a risk and related concepts can be considered as control (the goal used as countermeasure and plans that refine it and eventually other related concepts).

We proposed a graphical representation of the alignment of secure TROPOS with the ISSRM domain model in Fig. 9.26, Fig. 9.27, Fig. 9.28. Fig. 9.26 focuses on the risk- and risk treatment-related concepts. As we explained before, the ISSRM Cause of the risk is represented by secure TROPOS Threat. As the ISSRM Impact is modelled using negative Contribution relationship between a Threat and a Softgoal related to security – that corresponds to ISSRM Security criterion –, the Contributor of the relationship is the Softgoal and the Contributee is the Threat. The Restrict relationship has also a Restrictor – the security Constraint – and a Restrictee that we modelled in this diagram as *Tropos constructs used to model IS and Business assets.* We saw previously in the thesis that a Restrict relationship can only link a security Constraint to a Goal. But since *Tropos constructs used to model IS and Business assets* contain the Goal construct, our modelling is correct. We use this kind of abstract class – *Tropos constructs used to model IS and Business assets* – in order to hide parts of the metamodel that we do not want to discuss in the figure. Also hidden are concepts that composed the secure TROPOS Threat – and called *Risk-related concepts describing the Cause.*

Fig. 9.27 depicts concepts that composed a secure TROPOS Threat. Threat agent is mapped on Actor. An ISSRM Threat is expressed using concepts that can compose a Means-ends and/or a Decomposition relationships. A plan can be used as components of Means-ends and/or a Decomposition relationships. It can also be considered as the ISSRM Attack method – in combination with the Attacks relationship. That is why it is shared by the ISSRM Threat concepts and ISSRM Attack method concept. Beliefs represent a part of ISSRM Vulnerabilities. Vulnerabilities are characteristics of IS assets. Thus Belief is linked to ISSRM IS asset.

Fig. 9.28 represents ISSRM Security requirements. As we saw before, every concepts of secure TROPOS can be considered as related to risk treatment. Indeed, changes in system-to-be's design are made so as to mitigate risks. So the new design of the system-to-be can be seen as Security requirements.

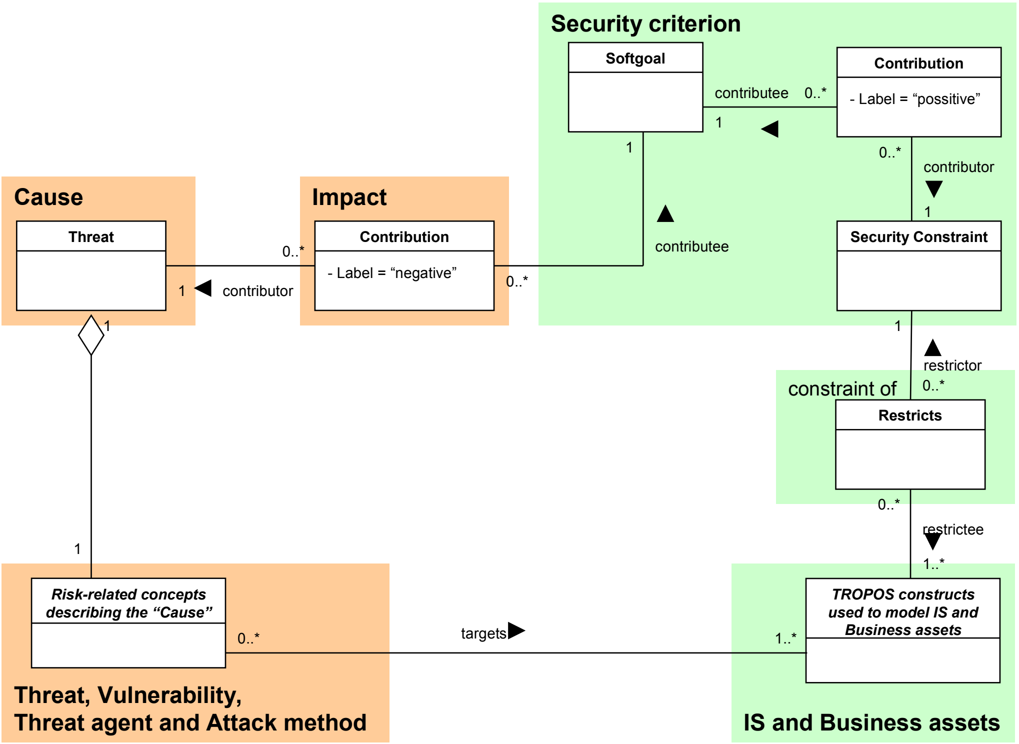


Figure 9.26: Alignment of secure TROPOS concepts with ISSRM risk- and assets-related concepts

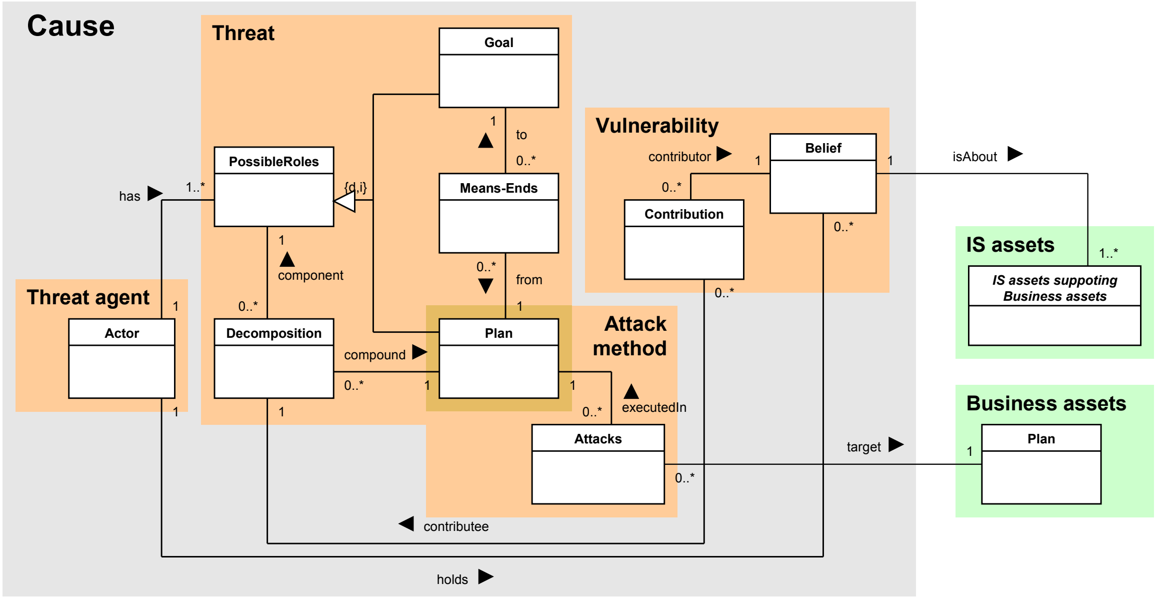


Figure 9.27: Alignment of secure TROPOS concepts with ISSRM focusing on risk-related concepts

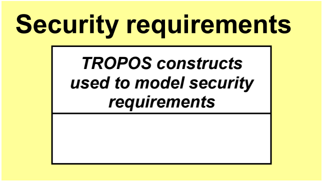


Figure 9.28: Alignment of secure TROPOS concepts with ISSRM risk treatment-related concepts

9.4 Summary

The second objective of the thesis was treated in this chapter. We presented the method we used to align concepts from security modelling languages with those from the ISSRM domain model. We put the theory into practice for the KeS and the secure TROPOS languages. Concerning secure TROPOS we first needed to elaborate its metamodel. Then we built the eSAP example and, based on this example, we aligned it with the ISSRM domain model. For both languages we depicted how they cover the concepts from the ISSRM domain model. This complete the second objective of this thesis. The next stage is the validation of these two alignments in the next chapter.

Part III

Evaluation

Chapter 10

Validation

As the validation of results is a paramount step in a scientific research method, we choose to build a case study in order to assess the validity of our alignment of security modelling languages with ISSRM domain model. Relevant remarks about the qualities as that a good Examplar should have are given in [FFFvL97] as a detailed description of a case study called the Meeting Scheduler Example. We developed our case study as an extension of the Meeting Scheduler. In the sequel of the thesis, we will thus discuss a) the modelling of a case study in KeS and secure TROPOS and b) the validity of our alignment presented in 9.3.3 based on models developed.

10.1 Objectives of the Exploitation of the Case Study

We will exploit models produced in secure TROPOS and KeS in order to validate the alignment between ISSRM domain model and, respectively, secure TROPOS and KeS. If concepts used in secure TROPOS and KeS to model a specific element of the case study have been aligned on the same ISSRM concept, it validates the alignment with this concept of the ISSRM domain model. If no problem arises during the complete analysis of concepts of ISSRM domain model, we can state the validity of our alignments of KeS and secure TROPOS with the ISSRM domain model. Then it will be to draw global conclusions on the coverage of secure TROPOS and KeS vis-à-vis the ISSRM domain model.

10.2 The Meeting Scheduler Case Study

The Meeting Scheduler system is a computer-based scheduler for supporting the setting up of meetings. The main requirement the system has to achieve is to determine a suitable date and time so that most of the intended participants will be able to attend the meeting effectively. The meeting scheduler interacts with the Meeting initiator – the agent responsible for the Meeting – and the Meeting participants – people who are possibly interested in the meeting. The meeting scheduling process is the following: when a meeting is to be scheduled, the initiator invites possibly interested participants. He defines a range of dates in which the meeting has to be scheduled. He also asks them to communicate their preferred and excluded dates depending on their personal agendas. The initiator then waits for the meeting scheduler to receive replies from interested participants and to compute a proposed date for the meeting. The date has to be as convenient as possible for most of the participants. All interested participants receive the proposed date. If it is convenient for them, they communicate their agreement to the meeting scheduler. Agreements from all invited participants are kept in the meeting scheduler for a sufficient period of time. These agreements are used as a kind of contract between the initiator and the participants. If a participant does not attend the meeting without cancelling its participation to the initiator, he can be compelled to some constraints. For the sake of brevity, we do not care about the rest of the meeting scheduling process. We limit the scope of the case study to this part of the scheduling of meeting and let some points rather blurred as this description of the case study reflects the basic knowledge of the domain that an analyst can have when beginning the early requirement elicitation. Other features are possible but not discussed here (see Appendix 4).

Instead of presenting in more details the internal requirements and goals of each actor of the system and, after that, focusing on the modelling in secure TROPOS and KeS, we directly give further explanations by referring to diagrams created with secure TROPOS. The presentation of the context of the meeting scheduler system can be overviewed in Fig. 10.1. Three actors are present: a) the meeting initiator, b) the meeting participant and c) the meeting scheduler. The invitation of initiator to participants is included in the `AttendMeeting` Goal. The initiator depends on the interest of the participant to join a meeting. The initiator relies on the meeting scheduler for the meeting to be scheduled as expressed by the `MeetingBeScheduled` Goal. The Plan `EnterRangeOfDates` expresses that the scheduler needs the initiator to provide a range of suitable dates among which the meeting has to be scheduled. The `ProposedDate` and the `Agreement` are modelled as Resources.

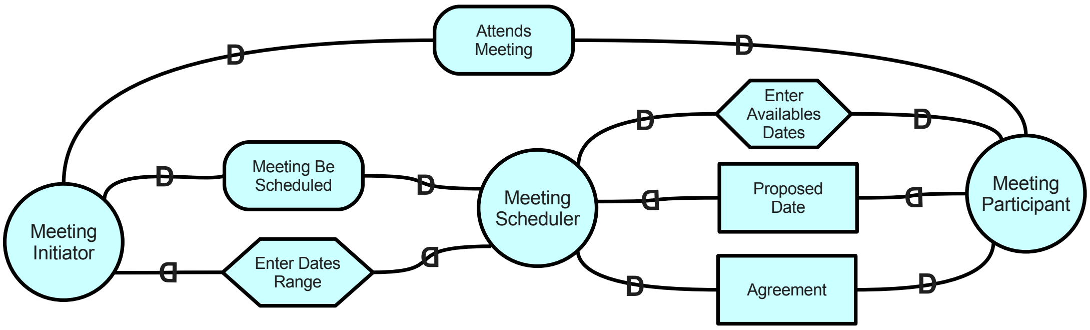


Figure 10.1: Overview of the meeting scheduler system (adapted from [Yu97])

Once we have elicited the relationships between actors, we can describe the internal modelling of each of them using a SRM. We begin by the investigation of the Meeting initiator as depicted in Fig. 10.2.

The Meeting initiator. The main Plan of this Actor is to `OrganizeMeeting`. `BeScheduled` is a way to fulfill this task and the organisation of meeting should be `Quick` and should require `LowEffort` for the initiator. Two alternatives are offered to him to achieve the scheduling of the meeting: a) the initiator can schedule the meeting manually or b) let the scheduler do it. The first Plan a) contributes negatively to the two Softgoals whereas the second Plan b) contributes positively.

The Meeting participant. As his name indicates it, he has to `ParticipateInMeetings`. In order to do this, he needs that a `Convenient` date for a meeting M is found. He also needs `AttendMeeting` M. The last Plan he has to fulfill before attaining his root Plan is to `Arrange-Meeting`. This arrangement should a) require `Low Effort` from the participant. Using a `User-Friendly` system is one of the way to achieve it. b) Finding a convenient date for meeting M must be possible – `Agreeable (meeting, date)`. Making the date convenient for meeting M is supported by the wish that the date is chosen as the most suitable from the point of view of the participant. In other words, the date should not be proposed as soon as a suitable date is found but by taking into account the most convenient date for each participant. Hence, we can speak about the `QualityOfTheProposedDate` and we can consider that the richer is the medium – the

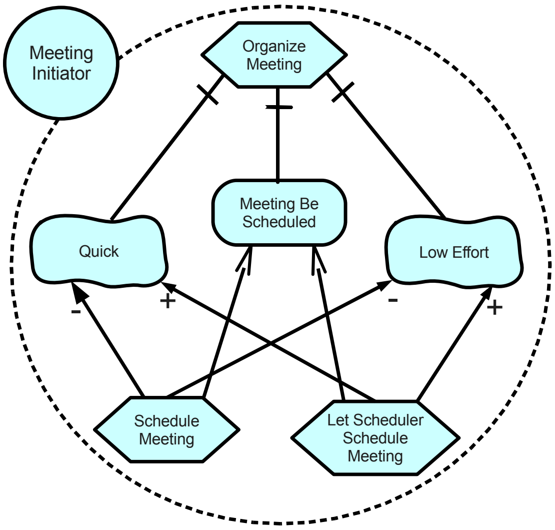


Figure 10.2: Secure TROPOS modelling of meeting initiator (adapted from [Yu97])

manner participants communicate to agree on proposed date – better the quality of the proposed date is. Two alternatives are considered to find agreeable dates: 1) to `FindAgreeableDates-UsingScheduler` or 2) to `FindAgreeableDatesTalkingToInitiator`. The first is a refinement of how a participant can make his agreement known by the meeting scheduler – `AgreeToThe-Date`. Nonetheless, it does not contribute to make the medium richer and to make the system user friendly. On the other hand, the second alternative has positive contributions on these two sub-goals. However, both are means to find an agreeable date for meetings.

The Meeting scheduler. This is the description of the computerized part of the system-to-be (see Fig. 10.4). The Meeting scheduler' s first Goal is that a `MeetingBeScheduled`. To achieve its Goal, the system performs a Plan called `ScheduleMeeting` that can be decomposed into three sub-components: a) the Plan `ObtainAvailableDates`, b) the Goal `FindAgreeableSlot` and c) the Plan `ObtainAgreement` (on the agreeable slot). The Plan `ObtainAvailableDates` is the process used to get the preferred and excluded dates set from all participants to the meeting M. `FindAgreeableSlot` consists of computing a suitable date for the meeting M that depends on the participants' available dates. This Plan is refined by the Plan `MergeAvailableDates` of all participants. `ObtainAgreement` deals with waiting and ensuring that a sufficiently large number of agreements from interested participants are received.

Once the three actors have been detailed, we can depict in Fig. 10.5 the global view of the system. As we focus on the Meeting scheduler in the sequel of our modelling, we have looked at the dependencies in which the scheduler is the Depender or the Dependee of the relationships. The description given above is based on [Yu97]. We now extend the case study with security concerns using the secure TROPOS language. The first step consists of adding security Constraints on the Dependency relationships. We have identified three security Constraints depending on the Meeting scheduler as presented in Fig. 10.6. `AgreementsAvailabilityMustBeEnsured` constraints the Dependum `MeetingBeScheduled` on the availability criterion and it is on the behalf of the Meeting scheduler. This constraint aims to ensure the availability of the agreements once the meeting is successfully scheduled. The two other security Constraints concern the Resource `Agreement (for meeting M)`. `OnlyUsedByParticipantsToM` is a privacy criterion that restricts the use of the agreement to only participants to meeting M. `AgreedDateCannotBeChanged` is related to the integrity of data and especially to the integrity of the agreement (remember that agreements are used as contract between initiator and participants). Once security Constraints of secure Depen-

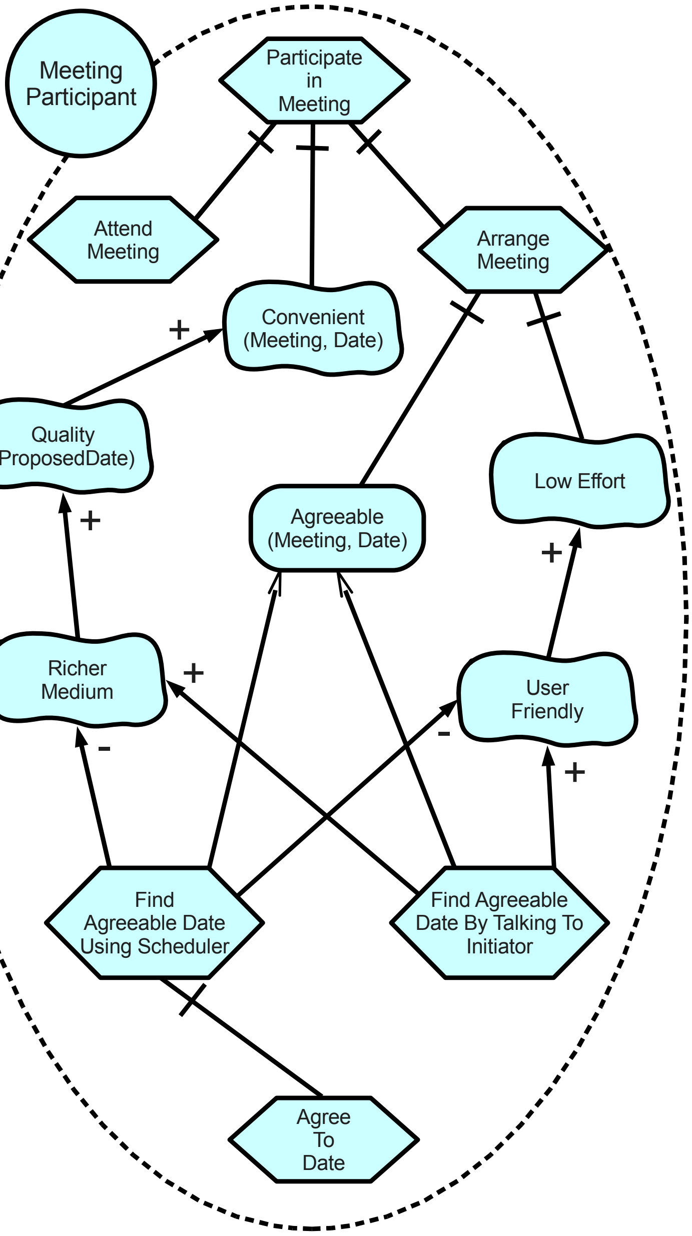


Figure 10.3: Secure TROPOS modelling of meeting participant (adapted from [Yu97])

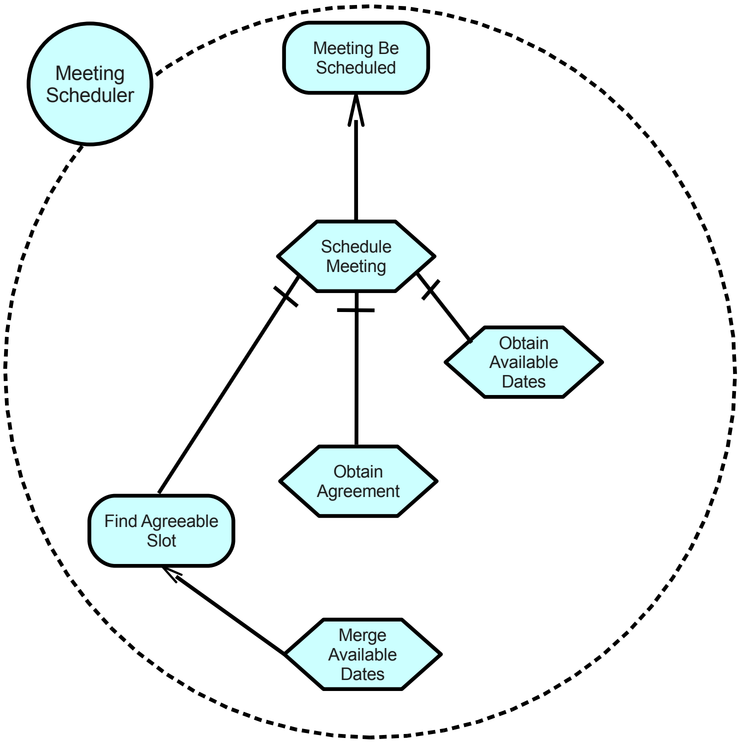


Figure 10.4: Secure TROPOS modelling of meeting scheduler (adapted from [Yu97])

dencies are elicited, the next step focuses on internal elements of the Meeting scheduler that are affected by the security Constraints or that are put in place in order to respect these constraints.

We now extend the case study with security concerns using the secure TROPOS language. The first step consists in adding security Constraints on the Dependency relationships. We have identified three security Constraints depending on the Meeting scheduler as presented in Fig. 10.6. `AgreementsAvailabilityMustBeEnsured` requires that the Dependum `MeetingBeScheduled` respects the availability criterion. This security Constraint has to be ensured by the Meeting scheduler. This constraint aims to ensure the availability of the agreements once the meeting is successfully scheduled. The two other security Constraints concern the Resource `Agreement (for meeting M)`. `OnlyUsedByParticipantsToM` is a privacy criterion that restricts the use of the agreement to only participants to meeting M. `AgreedDateCannotBeChanged` is related to the integrity of data and especially to the integrity of the agreement (remember that agreements are used as contract between initiator and participants). Once security Constraints of secure Dependencies are elicited, the next step focuses on internal elements of the Meeting scheduler that are affected by the security Constraints or that are put in place in order to respect these constraints.

As one can see in Fig. 10.7, the three previously elicited security Constraints are introduced within the boundary of the Meeting scheduler. First, we define the Goals of the meeting scheduler that are *restricted* by the security Constraints. The Goal `MeetingBeScheduled` is restricted by the security Constraint `AgreementAvailabilityMustBeEnsured`, `AgreementBeConfirmed` by `OnlyUsedByParticipantsToMeetingM` and `OldDataClearedOutFromScheduler` by `AgreedDate-CannotBeChangeOnceAgreementOnMReceived`. As we have taken into account security concerns of the system-to-be, it is normal that we refine the internal structure of the Meeting scheduler. That is why some new elements are added. The Goal `AgreementBeConfirmed` means that the Meeting scheduler wants that proposed date to be agreed. It is achieved by the Plan `ObtainAgreement`. A

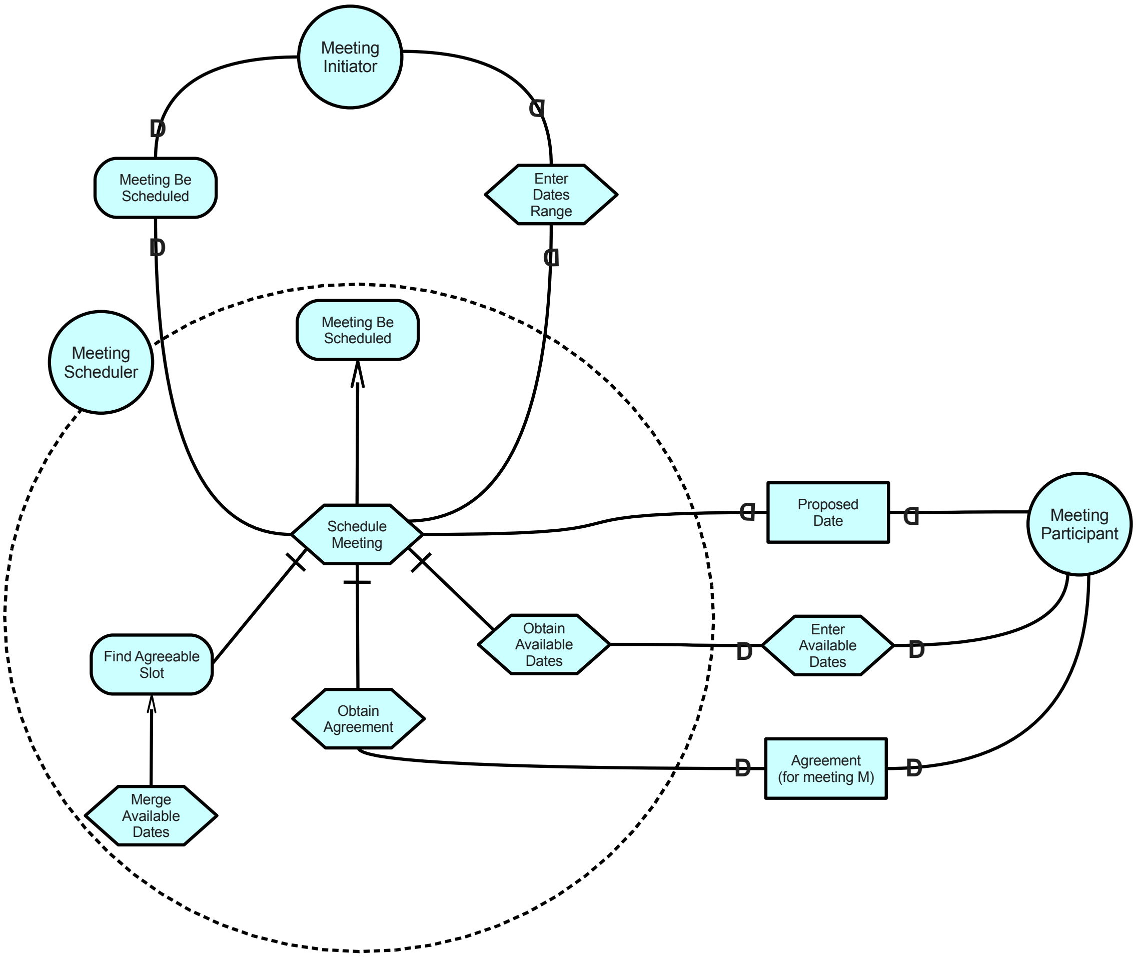


Figure 10.5: Secure TROPOS modelling of meeting scheduler system (adapted from [Yu97])

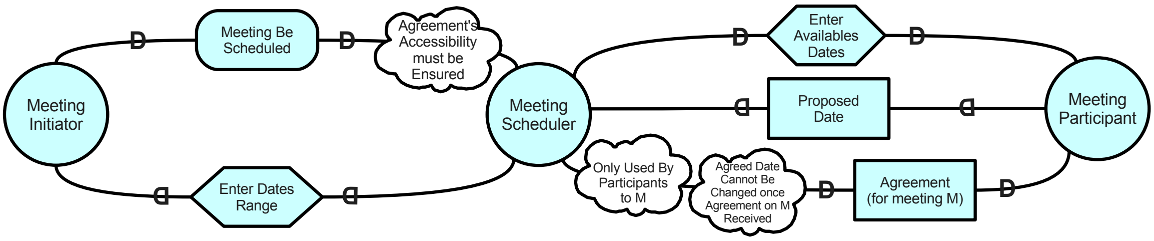

Meeting Initiator
D
Meeting Be Scheduled
D
Agreement's Accessibility must be Ensured
D
Enter Dates Range
D
Meeting Scheduler
D
Enter Availables Dates
D
D
Proposed Date
D
Meeting Participant
Only Used By Participants to M
Agreed Date Cannot Be Changed once Agreement on M Received
D
Agreement (for meeting M)
D

Figure 10.6: Adding security Constraints on the meeting scheduler system

new requirement is introduced in the system: there must be processes that clear out old data from the system. This is expressed by the Goal `OldDataClearOutFromScheduler` and it is fulfilled by the Plan `ClearOutOldData` that manipulates the `Agreements` Resource. As the availability of the agreements requires that they keep their integrity, the Goal `AgreedDateCannotBeChangeOnceAgreementOnMReceived` restricts the new Goal `OldDataClearedOutFromScheduler`.

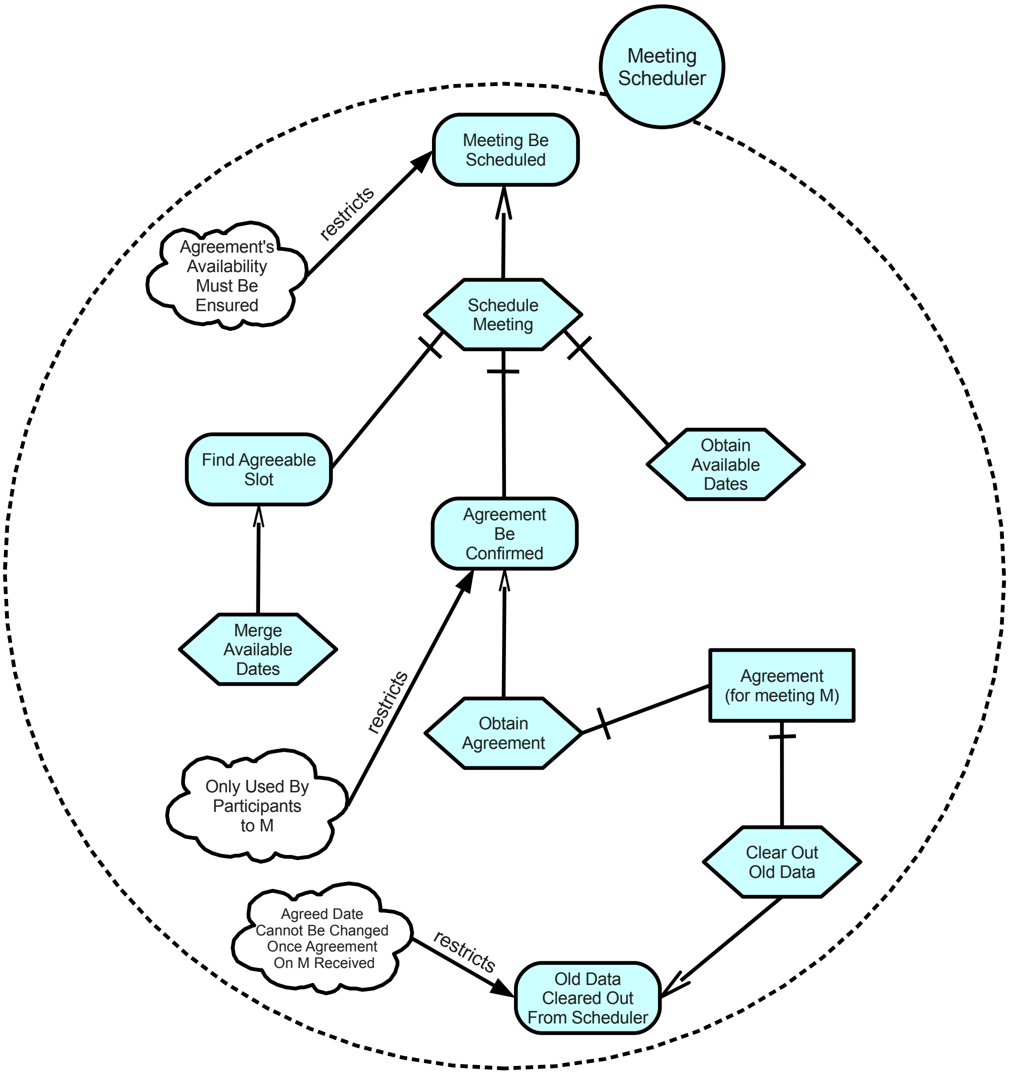


Figure 10.7: Security constraints and related elements in the meeting scheduler

Secondly and before investigating ways to have the system respecting the imposed security Constraints, we have to brainstorm on the possible risks related to the security Constraints. In Fig. 10.8 we present three risks, each one threatening a security Constraint. The way to build this diagram is the following: a) to place the previously elicited security Constraints on the diagram; b) to represent as Softgoals the security criteria that are threatened – in our case study: privacy, integrity and availability; c) we imagine how which kind of risk can break the security criteria. The label of the cause of the risk – depicted using the secure TROPOS construct of threat – should be chosen not too general (for example, attack against privacy) but not too operational either (for example, send confidential data to tier person). We suggest picking a label that does not begin with a verb (infinitive mode) but by a name of the category of the known attack methods against the category of the security criterion at hand. For privacy, we identify the Threat DisclosureAttack. We add contribution relationships in order to ensure that the elicited Softgoal is positively contributed by the security Constraint and that the identified risk has also a negative impact on this Softgoal. As mentioned before, the security Constraint AgreementAvailabilityMustBeEnsured is related to the Softgoal Availability which is threatened by the Threat DestructionOfTheAgreement. AgreedDateCannotBeChangedOnceAgreementOnMReceived concerns the Integrity of the system thwarted by the TamperingOfAgreements.

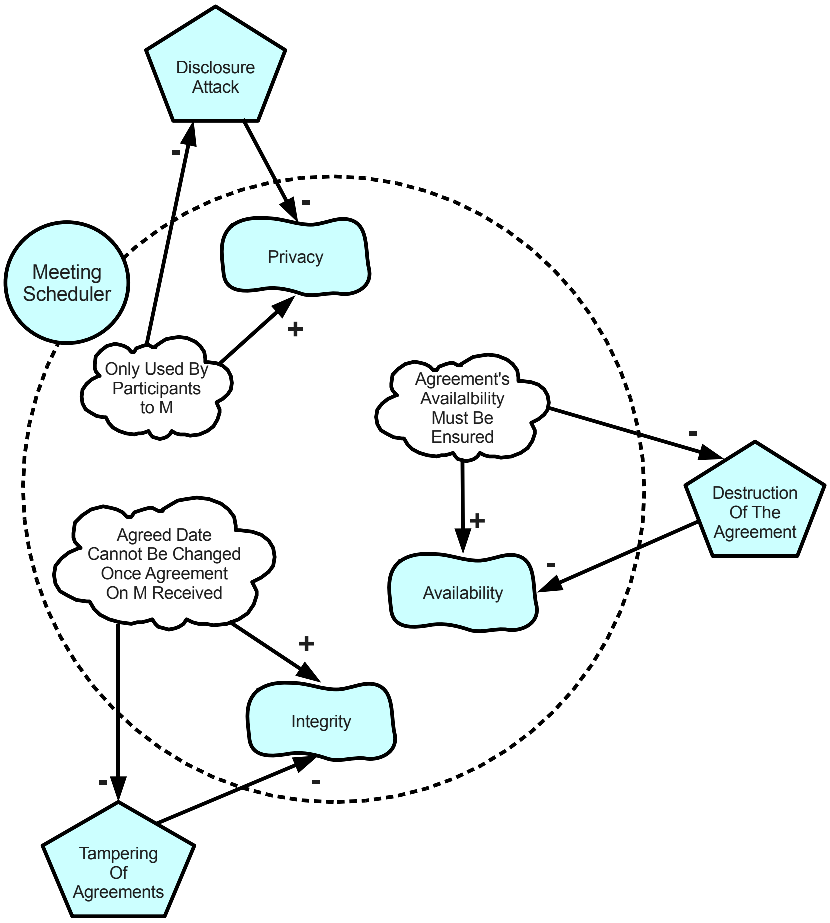

Disclosure Attack
-
-
Meeting Scheduler
Privacy
+
Only Used By Participants to M
Agreement's Availalbility Must Be Ensured
-
Destruction Of The Agreement
+
Agreed Date Cannot Be Changed Once Agreement On M Received
-
Availability
+
Integrity
-
-
Tampering Of Agreements

Figure 10.8: Elicitation of risks against security constraints and security criteria of the meeting scheduler

Thirdly, when we know which are the security criteria concerned by the security Constraints, we define new elements in the Meeting scheduler leading to the satisfaction of these constraints. For the sake of legibility we create one new model per security Constraint as depicted in Fig. 10.10, Fig. 10.9 and Fig. 10.11. We limit deliberately the description of this case study to the refinement of the privacy constraint. We introduce the security criterion Privacy as a Softgoal in the internal structure of the Meeting scheduler. We define the secure Plan (S) AssurePrivacy as a way to contribute positively to the Softgoal. The (S) tag indicates that the element is related to the security of the system. This secure Plan is decomposed into the secure sub-goal (S) PrivacyOfTheSystemEnsured. The process to achieve this sub-goal consists in checking the identity of the participants – (S) CheckParticipantIdentity – that access the meeting scheduler system. The secure Plan (S) CheckParticipantIdentity contributes positively to the security Constraint OnlyUsedByParticipantsToMeetingM. At this point of the analysis of the system-to-be, we have expressed the happy scenario with security concerns of the system. But, in order to reinforce the security, we make an analysis of the system-to-be from the point of view of an attacker. The risk against the privacy criterion is used as a root Goal `DisclosureOfAgreements` in the internal structure of the attacker. One of the means to fulfill the attacker's root Goal is to `DiscloseAgreements`, itself decomposed into two sub-plans `ReadAgreements` and `SendToUnauthorizedPeople`. Remember that the Decomposition relationship is equivalent to an AND decomposition when modelling with secure TROPOS. The attack of the malicious agent can only be successful if he knows some vulnerabilities of the system. The vulnerability that the attack method `DiscloseAgreements` exploits is the knowledge of the attacker who has the conviction that `AgreementsAreNotEncrypted`. Once that he has his attack method and that he knows which are vulnerabilities to exploit, the attacker can attack the Meeting scheduler by targeting the `Agreement (for meeting M)` Resource.

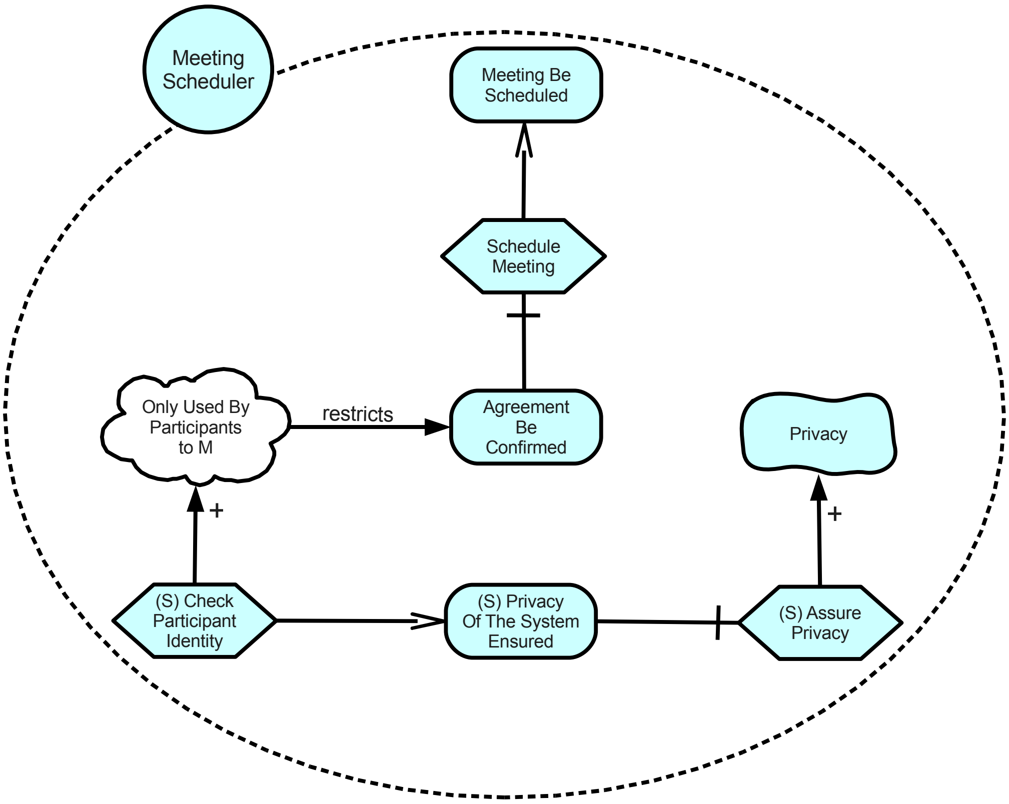


Figure 10.9: Meeting scheduler extended with privacy related elements

When an attacker scenario is elaborated, we need to improve the internal structure of the Meeting scheduler in a way that mitigates the identified attack method (see Fig. 10.13). Such a mitigation is not always possible as we discussed previously in Chapter 3 and as depicted in Fig. 3.2. To counter the `DiscloseAgreements` attack method, we introduce a new Plan `(S) PerformCryptographicProcedures` that is another means to fulfill the secure Goal `(S) PrivacyOfTheSystemEnsured` and that contributes positively to the security Constraint `OnlyUsedByParticipantsToM`. We tag the new plan with **(S)** meaning that this element is a countermeasure related to security. The introduction of security countermeasures is potentially dangerous as they can contain vulnerabilities. So for each countermeasure added to the system, we need to investigate if new risks emerged.

We now model the same case study using the KeS language.

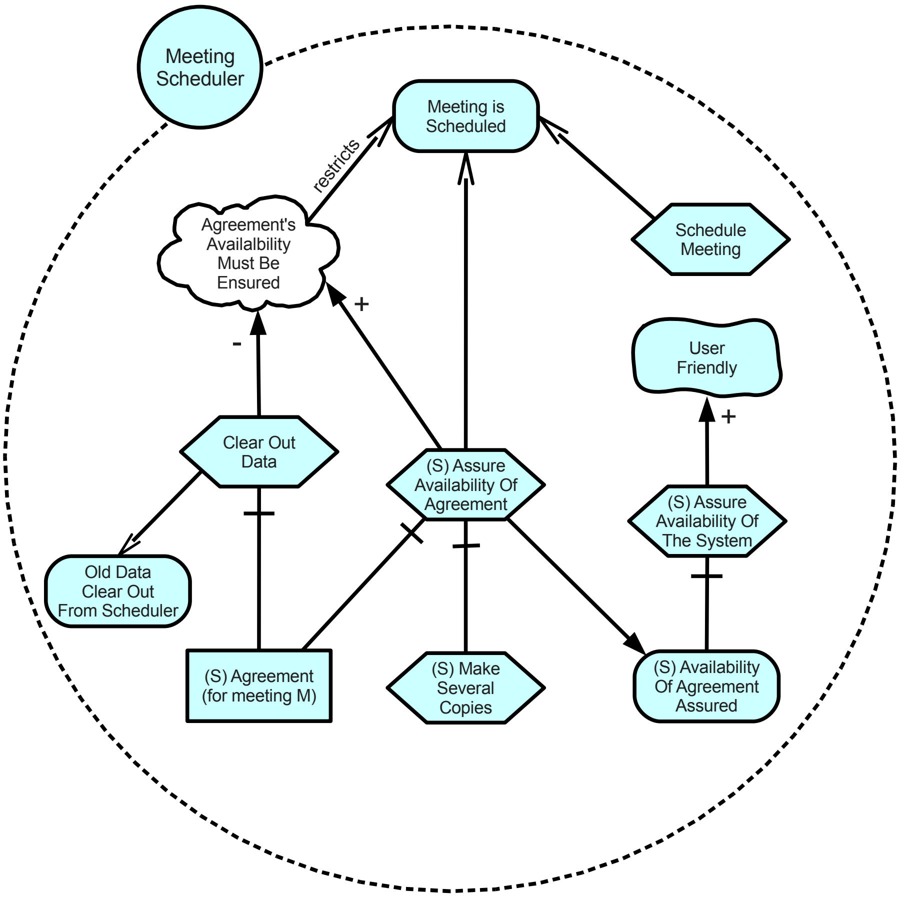

Meeting Scheduler
Meeting is Scheduled
restricts
Agreement's Availalbility Must Be Ensured
Schedule Meeting
+
-
User Friendly
+
Clear Out Data
(S) Assure Availability Of Agreement
(S) Assure Availability Of The System
Old Data Clear Out From Scheduler
(S) Agreement (for meeting M)
(S) Make Several Copies
(S) Availability Of Agreement Assured

Figure 10.10: Meeting scheduler extended with availability related elements

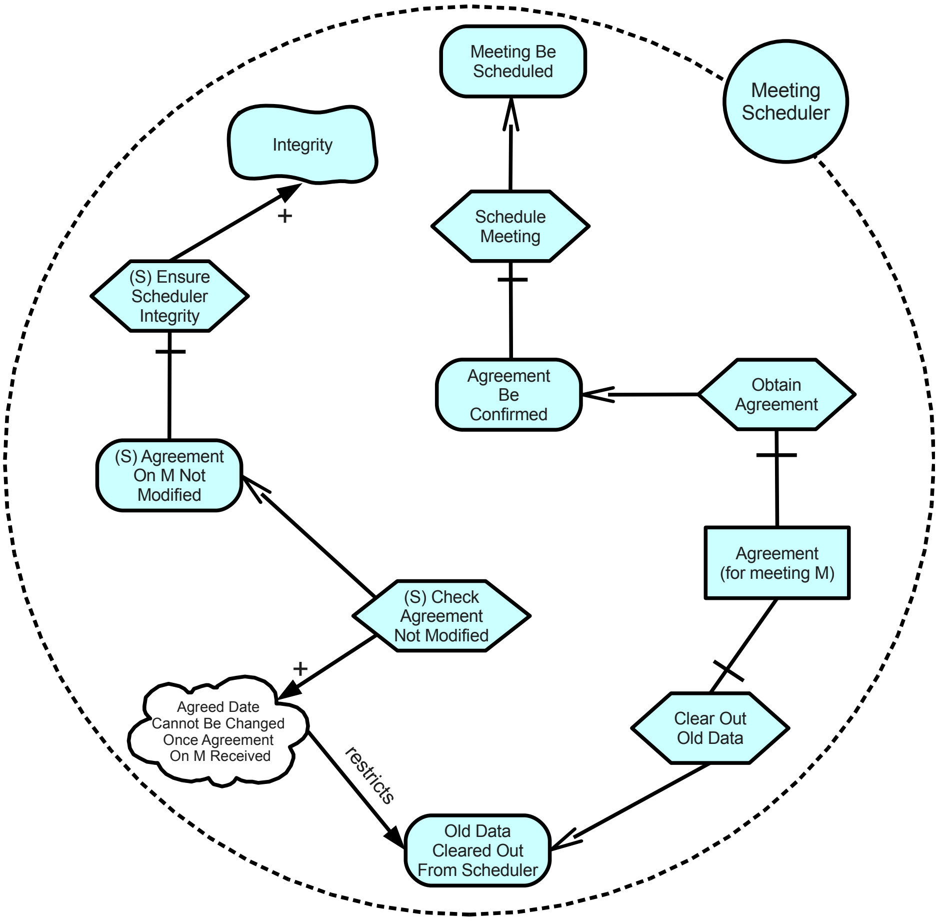

Meeting Scheduler
Meeting Be Scheduled
Integrity
+
Schedule Meeting
(S) Ensure Scheduler Integrity
Agreement Be Confirmed
Obtain Agreement
(S) Agreement On M Not Modified
Agreement (for meeting M)
(S) Check Agreement Not Modified
+
Agreed Date Cannot Be Changed Once Agreement On M Received
restricts
Clear Out Old Data
Old Data Cleared Out From Scheduler

Figure 10.11: Meeting scheduler extended with integrity related elements

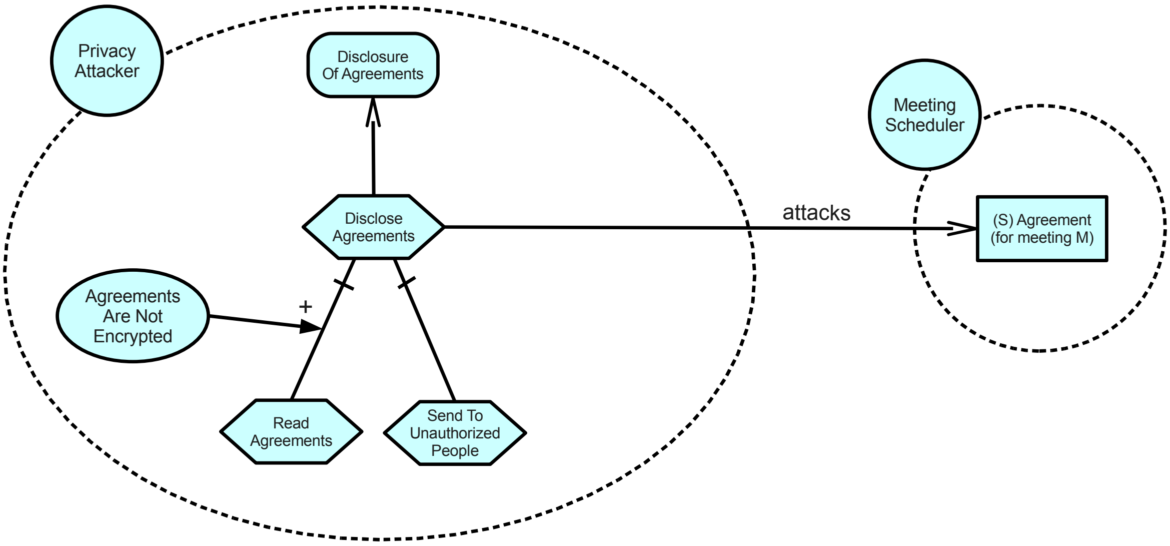


Figure 10.12: Internal structure of the attacker against meeting scheduler privacy

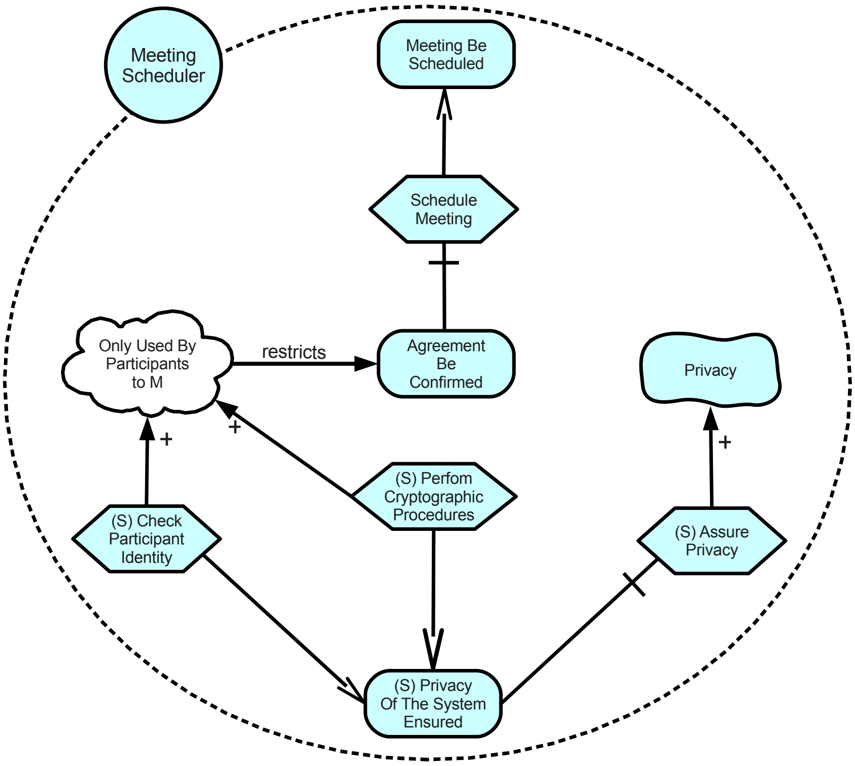

Meeting Scheduler
Meeting Be Scheduled
Schedule Meeting
Only Used By Participants to M
restricts
Agreement Be Confirmed
Privacy
+
+
+
(S) Perfom Cryptographic Procedures
(S) Check Participant Identity
(S) Assure Privacy
(S) Privacy Of The System Ensured

Figure 10.13: Countermeasure against disclosure of agreements

Modelling in KeS

We model the same case with the KeS language. The approach is rather different from the one used for secure TROPOS. Details and comparisons about the two approaches are given in the sequel of the thesis. In this section, we explain how the KeS model is constructed step by step considering first the point of view of the designer of the system and after that the point of view of a potential attacker. In each case, we build in the same time the Goal (respectively Anti-goal) model and the Object (respectively Anti-object) model.

At the beginning of the early requirements phase, the main Goal of the system-to-be is `MeetingBeScheduled`. The content of the Goal has to be achieved so we can categorize it as an *Achieve* Goal (the label of the Goal is preceded by *Achieve* and it is place into square brackets). We distinguish two alternatives sub-goals to achieve the root Goal: a) `MeetingBeManuallyScheduled` and b) `MeetingBeAutomaticallyScheduled`. We do not investigate the manual scheduler because it is not the envisaged scenario of our case study. In the same time, the `MeetingBeAutomaticallyScheduled` fulfills two Softgoals respectively labelled `QuickScheduling` and `LowEffortScheduling`. We refine the sub-goal b) until we elicit Requirements or Expectations whose responsibility or assignment is attributed to agents. To make model legible, we split the entire model (depicted in Fig. 10.14) into three parts Fig. 10.15, Fig. 10.16 and Fig. 10.17 corresponding to the three agents found in the system.

The sub-goal `MeetingBeAutomaticallyScheduled` is decomposed into three Goals b1) `InterestedParticipantsForMeetingMFound`, b2) `PreferredAndExcludedDatesForMeetingMKnownByMeetingScheduler` and b3) `SuitableAgreementForMObtained`. All three Goals are from the category of *Achieve* goals. b1) means that, in order to schedule automatically the meeting, the system needs to find possibly interested participants for the meeting M. b2) permits to participants to send

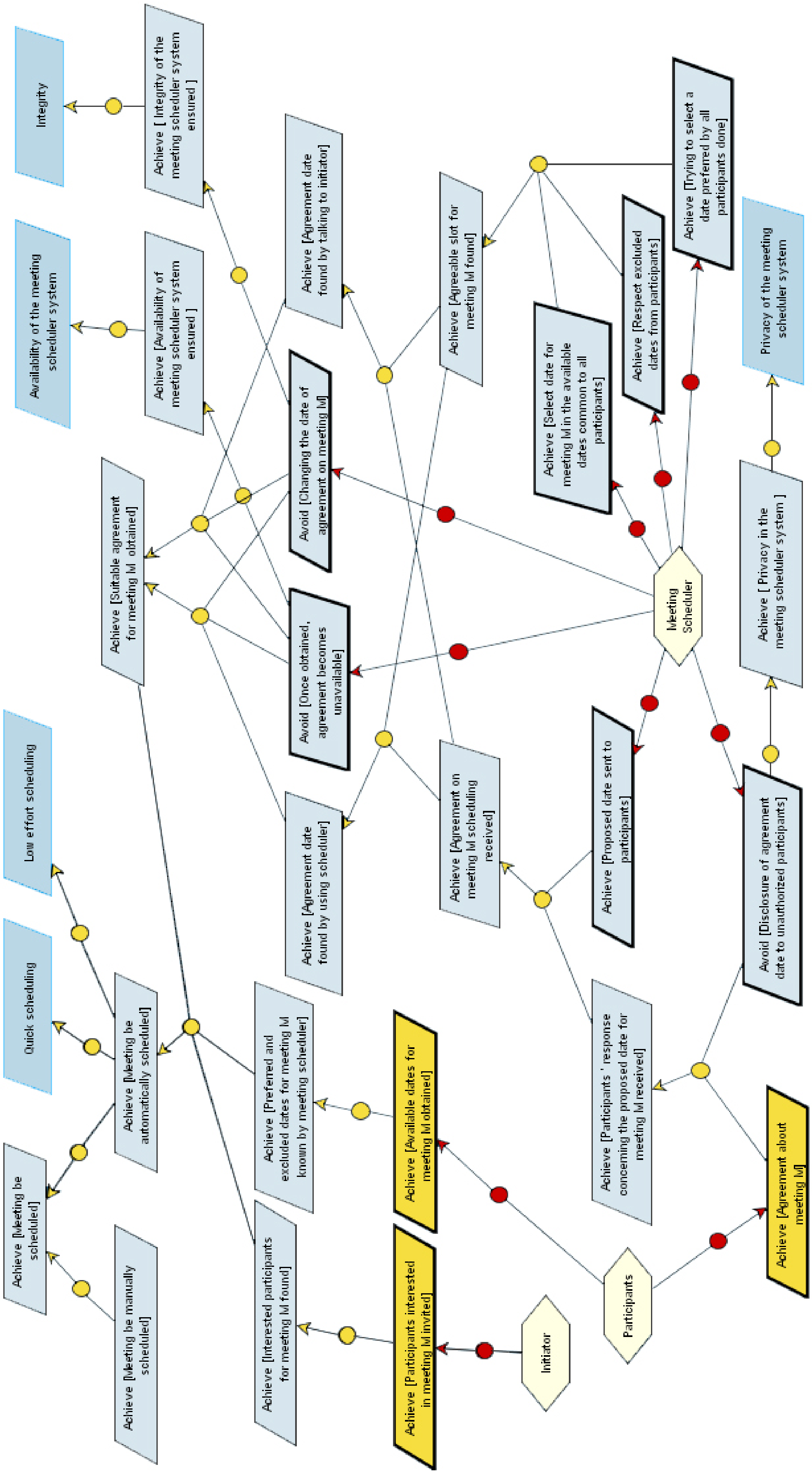

Integrity
Achieve [Integrity of the meeting scheduler system ensured]
Availability of the meeting scheduler system
Achieve [Availability of meeting scheduler system ensured]
Achieve [Agreement date found by talking to initiator]
Achieve [Agreeable slot for meeting M found]
Achieve [Trying to select a date preferred by all participants done]
Achieve [Respect excluded dates from participants]
Achieve [Select date for meeting M in the available dates common to all participants]
Privacy of the meeting scheduler system
Achieve [Suitable agreement for meeting M obtained]
Avoid [Changing the date of agreement on meeting M]
Avoid [Once obtained, agreement becomes unavailable]
Meeting Scheduler
Achieve [Privacy in the meeting scheduler system]
Low effort scheduling
Quick scheduling
Achieve [Meeting be scheduled]
Achieve [Meeting be automatically scheduled]
Achieve [Meeting be manually scheduled]
Achieve [Agreement date found by using scheduler]
Achieve [Agreement on meeting M scheduling received]
Achieve [Proposed date sent to participants]
Avoid [Disclosure of agreement date to unauthorized participants]
Achieve [Preferred and excluded dates for meeting M known by meeting scheduler]
Achieve [Interested participants for meeting M found]
Achieve [Available dates for meeting M obtained]
Achieve [Participants interested in meeting M invited]
Achieve [Participants ' response concerning the proposed date for meeting M received]
Achieve [Agreement about meeting M]
Initiator
Participants

Figure 10.14: Complete goal model of the Meeting scheduler system (adapted from [Yu97])

their preferred and excluded dates to the meeting scheduler system. b3) assures that the Meeting scheduler succeeds in a agreement that is as convenient as possible for all interested participants. In Fig. 10.15 we refine b1) into an Expectation `ParticipantsInterestedInMeetingMInvited`. The expectation is assigned to the Meeting initiator.

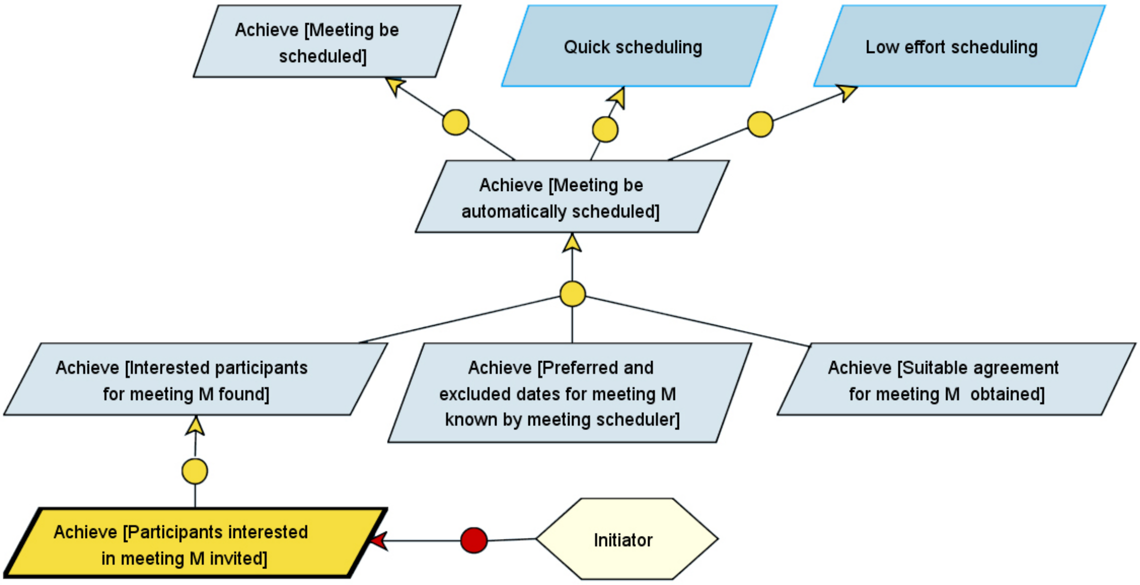

Achieve [Meeting be scheduled]
Quick scheduling
Low effort scheduling
Achieve [Meeting be automatically scheduled]
Achieve [Interested participants for meeting M found]
Achieve [Preferred and excluded dates for meeting M known by meeting scheduler]
Achieve [Suitable agreement for meeting M obtained]
Achieve [Participants interested in meeting M invited]
Initiator

Figure 10.15: Meeting scheduler system refined until Requirements and Expectations: focus on the initiator Agent

Fig. 10.16 presents two Expectations `AvailableDatesForMeetingMObtained` and `AgreementAboutMeetingM`. The first is obtained from the Goal `PreferredAndExcludedDatesForMeetingMKnownByMeetingParticipants` and means that participants to meeting M have to transmit their available dates to the scheduler system. The second is a refinement of the Goal `ParticipantsResponseConcerningTheProposedDateForMeetingMReceived` – itself obtained by successive refinements of the Goal b3) and explained in the sequel. Agreement about meeting M is assigned to participants because they have to give agreements to meetings.

As presented in Fig. 10.17, the Meeting scheduler is responsible for the seven requirements, all obtained by refinement of the goal b3) `SuitableAgreementForMeetingMObtained`. Two alternatives to achieve the obtainment of a suitable agreement are envisaged. The first manner (Fig. 10.18) is composed of the Goal `AgreementDateFoundByUsingScheduler` and two Requirements related to security 1) `OnceObtained,AgreementBecomesUnavailable` and 2) `ChangingTheDateOfAgreemenOnMeetingM`. The two security Requirements are expressed using *Avoid* category. 1) avoids that an agreement received by the Meeting scheduler becomes unavailable – this is a serious problem since agreements are a kind of contracts between initiator and participants – and 2) avoids that the information and particularly the date of the agreement is modified.

The second alternative is `AgreementDateFoundByTalkingToInitiator` and it has to be considered in conjunction with the two previously described security Requirements. Either by the first or the second alternative, both are achieved performing two sub-goals: `ScheduleOfAgreementOnMeetingMReceived` and `AgreeableSlotForMeetingMFound`. To receive the schedule of all interested participants, the Requirement `ProposedDateSentToParticipants` and its results – the Goal `ParticipantsResponseConcerningTheProposedDateForMeetingMReceived` –
have to be fulfilled. This Goal is refined into an Expectation (previously described) and a security

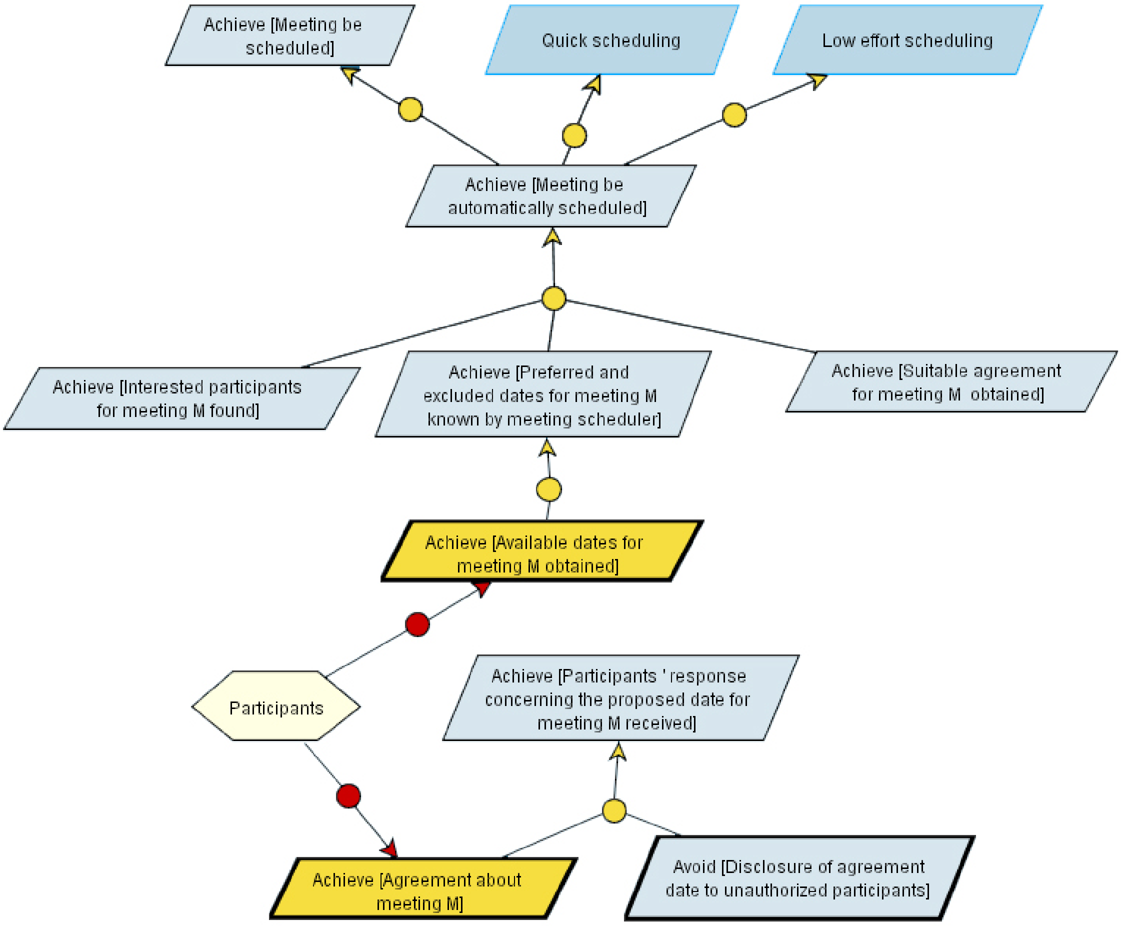

Achieve [Meeting be scheduled]
Quick scheduling
Low effort scheduling
Achieve [Meeting be automatically scheduled]
Achieve [Interested participants for meeting M found]
Achieve [Preferred and excluded dates for meeting M known by meeting scheduler]
Achieve [Suitable agreement for meeting M obtained]
Achieve [Available dates for meeting M obtained]
Participants
Achieve [Participants ' response concerning the proposed date for meeting M received]
Achieve [Agreement about meeting M]
Avoid [Disclosure of agreement date to unauthorized participants]

Figure 10.16: Meeting scheduler system refined until Requirements and Expectations: focus on the participant Agent

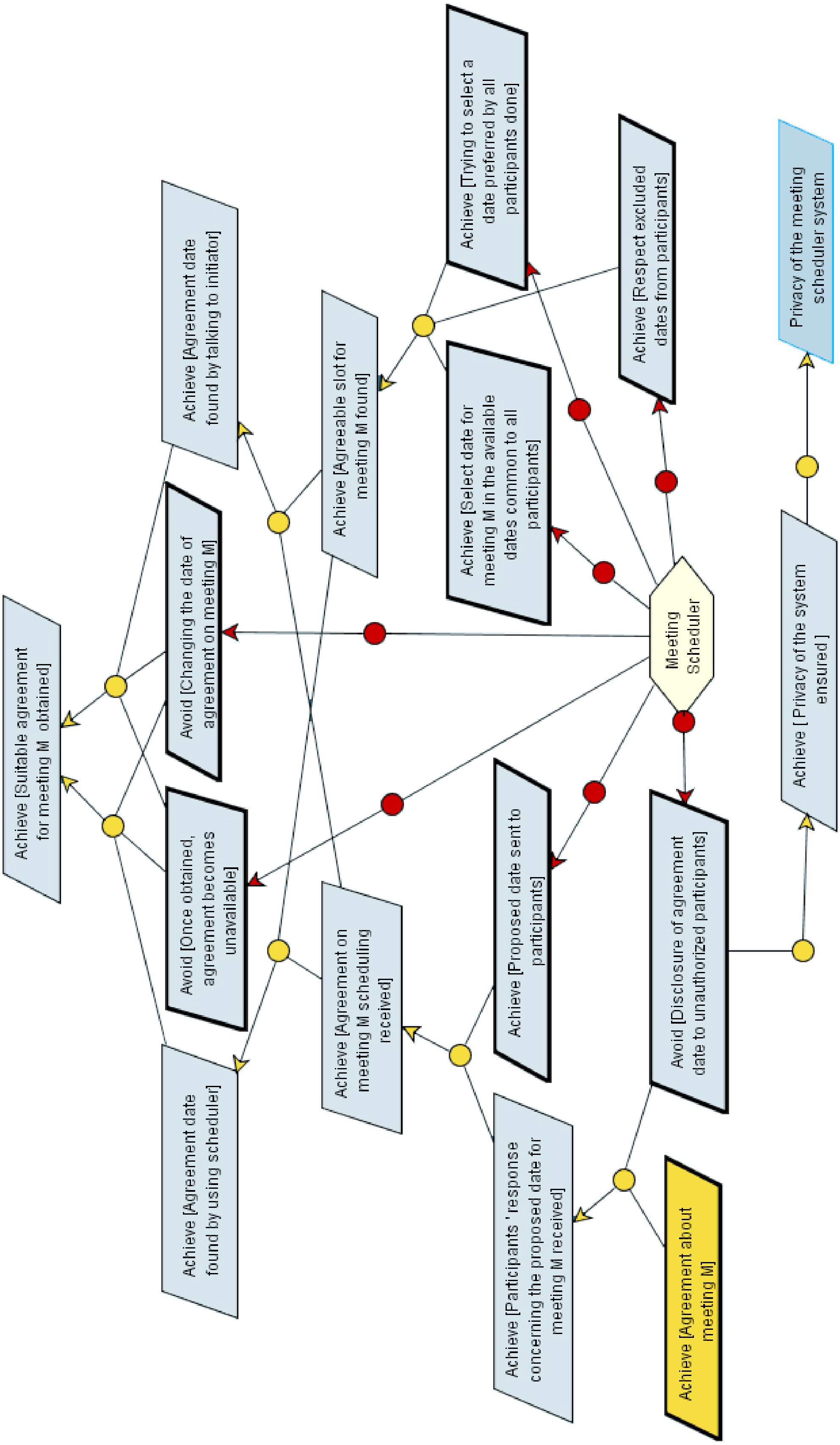

Achieve [Trying to select a date preferred by all participants done]
Achieve [Respect excluded dates from participants]
Privacy of the meeting scheduler system
Achieve [Agreement date found by talking to initiator]
Achieve [Agreeable slot for meeting M found]
Achieve [Select date for meeting M in the available dates common to all participants]
Achieve [Suitable agreement for meeting M obtained]
Avoid [Changing the date of agreement on meeting M]
Meeting Scheduler
Achieve [Privacy of the system ensured]
Avoid [Once obtained, agreement becomes unavailable]
Achieve [Proposed date sent to participants]
Avoid [Disclosure of agreement date to unauthorized participants]
Achieve [Agreement on meeting M scheduling received]
Achieve [Agreement date found by using scheduler]
Achieve [Participants 'response concerning the proposed date for meeting M received]
Achieve [Agreement about meeting M]

Figure 10.17: Meeting scheduler system refined until Requirements and Expectations: focus on the scheduler Agent

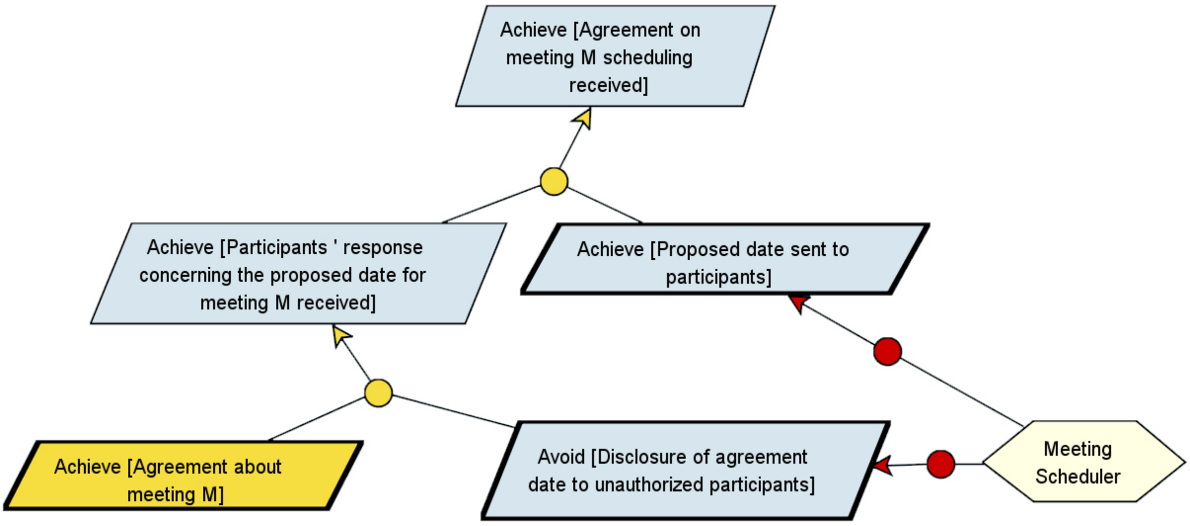

Achieve [Agreement on meeting M scheduling received]
Achieve [Participants ' response concerning the proposed date for meeting M received]
Achieve [Proposed date sent to participants]
Achieve [Agreement about meeting M]
Avoid [Disclosure of agreement date to unauthorized participants]
Meeting Scheduler

Figure 10.18: Refinement of the goal `Agreement on meeting M schedule received`

Requirement `DisclosureOfAgreementDateForUnauthorizedParticipants` that avoids that the date of agreement for a meeting M to be disclosed to unauthorized participants – participants attending other meetings. We can operationalize this requirement as presented in Fig. 10.19. To avoid date to be disclosed, the Meeting scheduler checks the identity of participants who want to manipulate an agreement. Input of the `CheckParticipantIdentity` operation is the Meeting participant Object and the output is an access to the `Database` containing agreements and a specific `Agreement`.

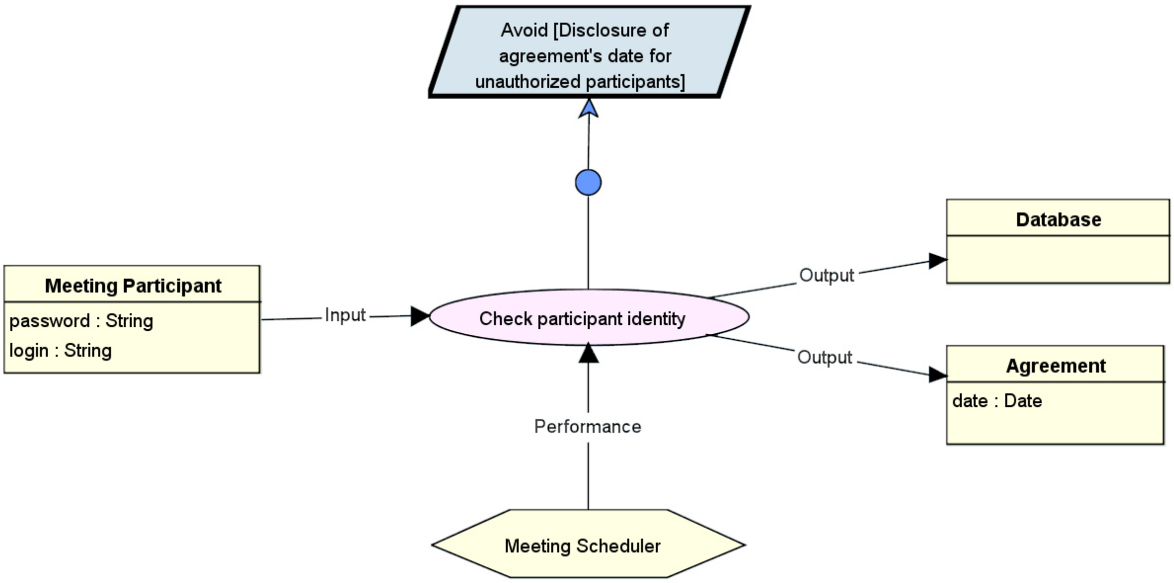


Figure 10.19: Operationalization of Requirement `DisclosureOfAgreementDateToUnauthorized-Participants`

Let us now consider the refinement of the Goal `AgreeableSlotForMeetingM` in Fig. 10.20. It is decomposed into three Requirements a) `SelectDateForMeetingMInTheAvailableDatesCommon-`

ToAllParticipants, b) TryToSelectADatePreferredByAllParticipants and c) RespectExcludedDatesFromParticipants. The meaning of these three Requirements is obvious from their labels.

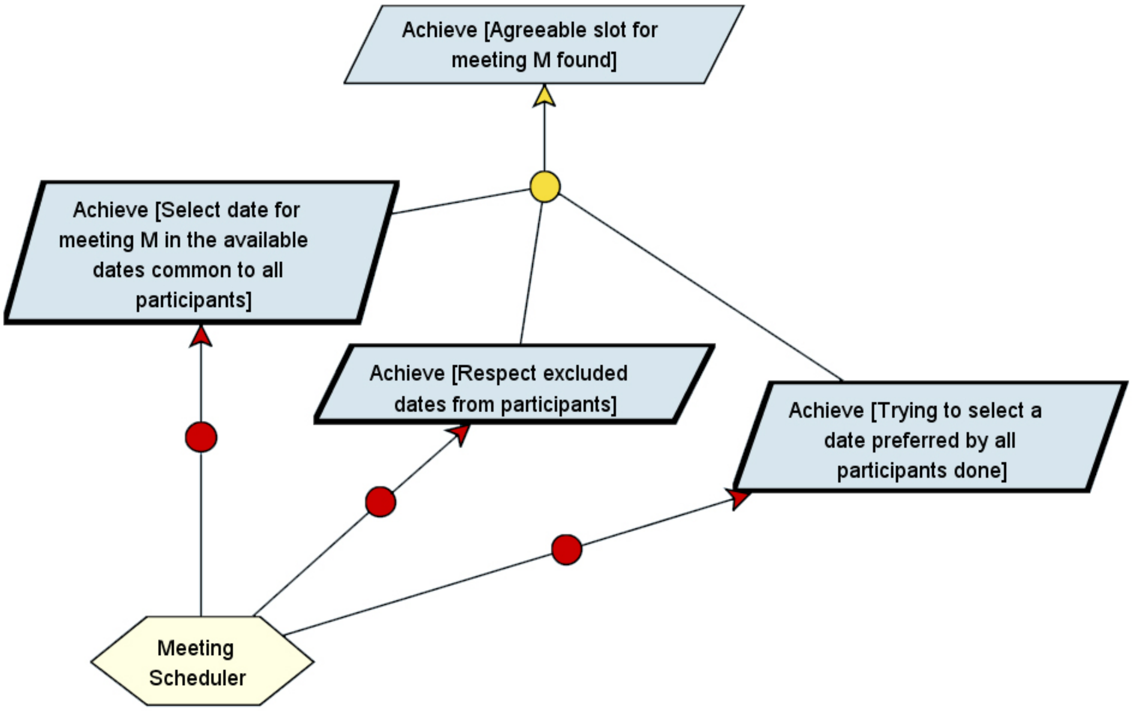

Achieve [Agreeable slot for meeting M found]
Achieve [Select date for meeting M in the available dates common to all participants]
Achieve [Respect excluded dates from participants]
Achieve [Trying to select a date preferred by all participants done]
Meeting Scheduler

Figure 10.20: Refinement of the goal Agreeable slot for meeting M found

The object model is presented in Fig. 10.21. The model contains three Objects representing agents: MeetingInitiator, MeetingParticipant and MeetingScheduler. The MeetingScheduler helps the MeetingInitiator to schedule meetings and it interacts with MeetingParticipants. The notion of Date is crucial for our case study as this object represents the threatened concept of the system. Date can be refined into two sub-classes: ExcludedDates and AvailableDates. ExcludedDates of participant P are dates that cannot be chosen to schedule a meeting if P has to attend the meeting. AvailableDates gather PreferredDates of participants and the date proposed by the Meeting scheduler. The Agreement object corresponds to the Agreement as defined previously in the description of the case study. It has a date as attribute and it is linked to ProposedDate, meaning that an agreement is obtained on a specific ProposedDate. MeetingParticipants send their Preferred and ExcludedDates to the Meeting scheduler in the form of a SetOfDates. Agreements are stored in a Database managed by the MeetingScheduler. Other Objects of the object model will be discussed after the analysis of security.

We analyse the Meeting scheduler system from the point of view of a potential attacker by designing the Anti-goal model. We described the potential attack performed by an attacker who wants to breach the security criterion of privacy of the system (see Fig. 10.22). We proposed to change the category of the privacy Requirement Avoid [DisclosureOfTheAgreementDateToUnauthorizedParticipants] into Achieve [DisclosureOfTheAgreementDateToUnauthorizedParticipants]. To achieve this root Goal, the attacker first needs to InfiltrateTheMeetingSchedulerSystem and after to GetRelevantDateOfAgreement. In order to get the relevant date, the attacker has to AccessTheAgreement, OnceInTheSystemStealDateOfAgreement and finally to RevealStolenDate. By refining the theft of date, we distinguish three requirements and a domain

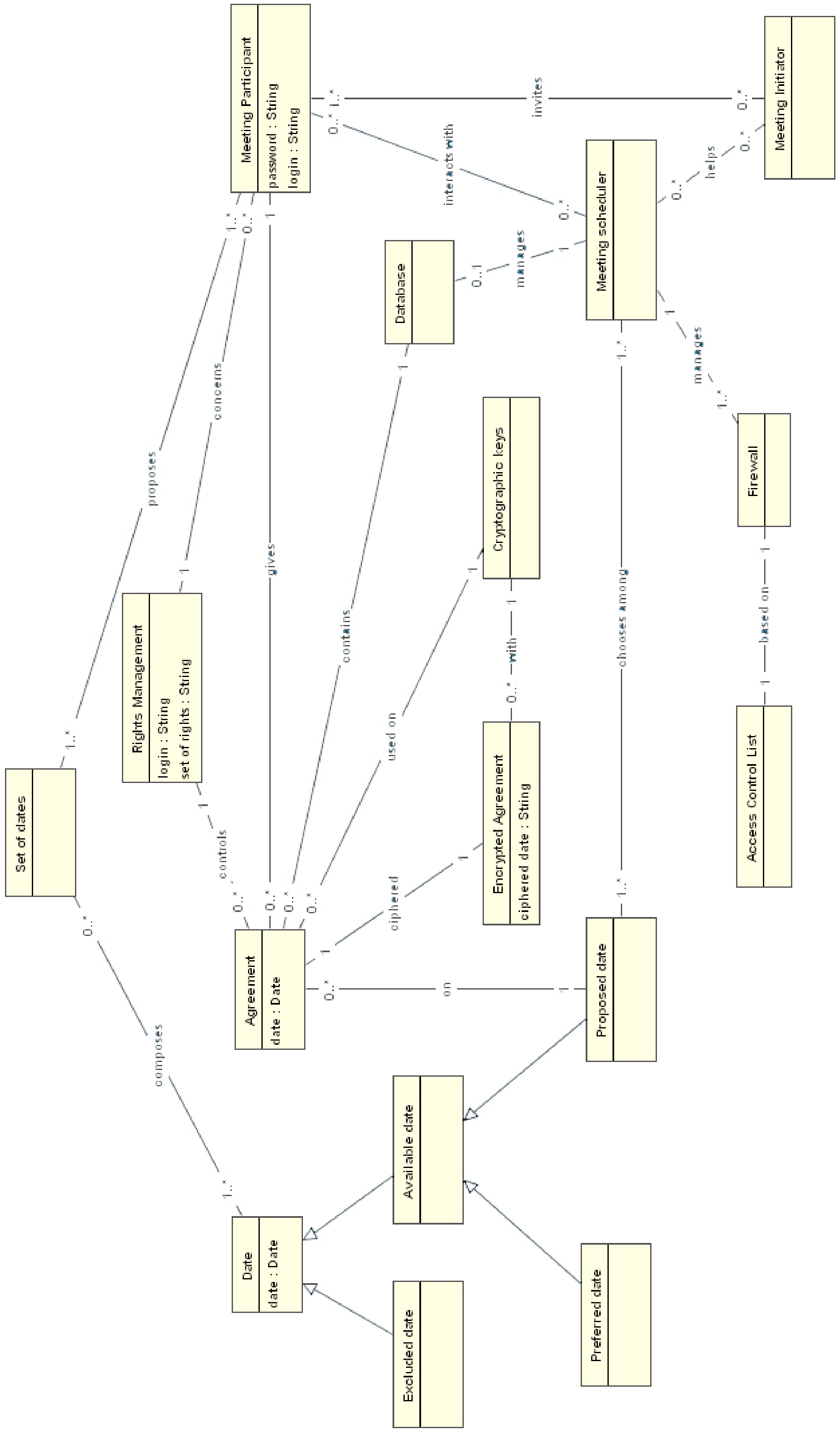


Figure 10.21: Object model of the meeting scheduler system (adapted from [Yu97])

property. Achieve [ReadDate], Achieve [CopyDate] and Achieve [StoreStolenDate] are the attacker's versions of the requirement of type *Avoid* previously elicited. The attack method of the attacker is defined as the operationalization of the three requirements. The exploited vulnerability is described in the Domain property AgreementsAreNotEncrypted.

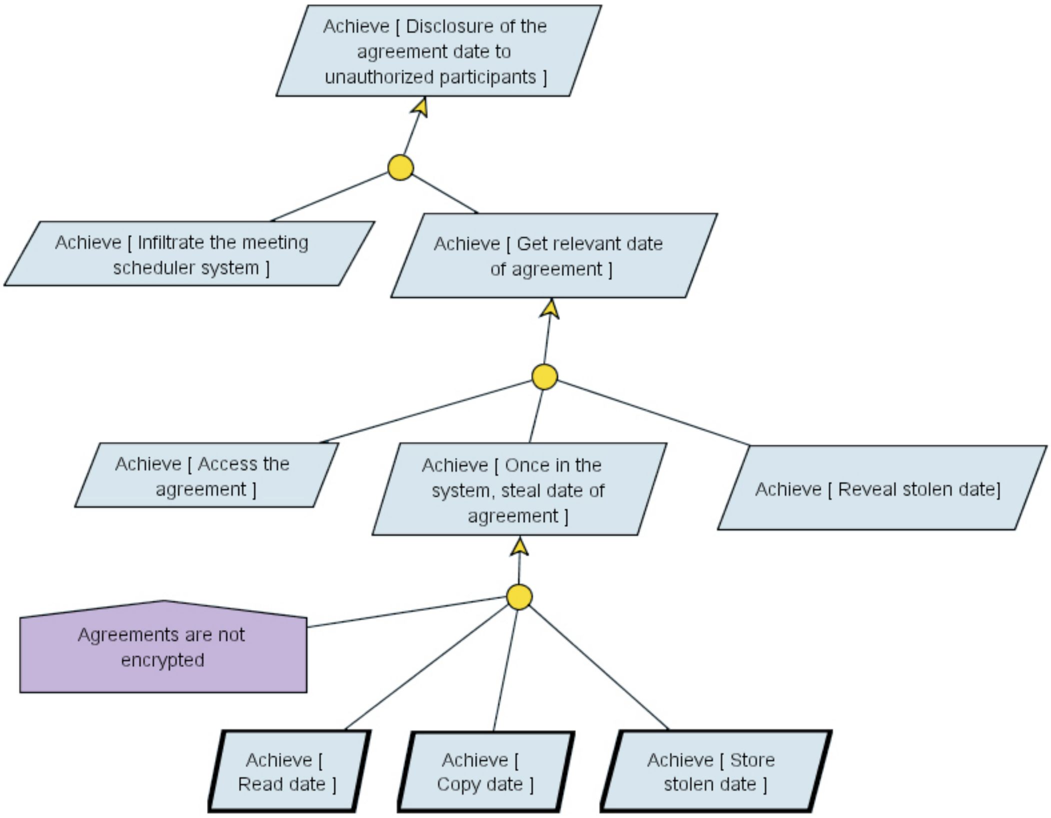

Achieve [Disclosure of the agreement date to unauthorized participants]
Achieve [Infiltrate the meeting scheduler system]
Achieve [Get relevant date of agreement]
Achieve [Access the agreement]
Achieve [Once in the system, steal date of agreement]
Achieve [Reveal stolen date]
Agreements are not encrypted
Achieve [Read date]
Achieve [Copy date]
Achieve [Store stolen date]

Figure 10.22: Goal model of the privacy attacker

We operationalize the three Requirements to elicit the attack method. Achieve [ReadDateOfTheAgreement] is operationalized into two Operations: a) NavigateInAgreementsDatabase and b) ReadDateOfAgreementsOfAParticipantToMeetingM. To find the agreement containing the relevant date to reveal, the attacker has to perform the operation a) using as input the Database object. After that, in b), he uses the agreement to read the date it contains.

Then in Fig. 10.24 we consider the second requirement Achieve [CopyDate] that is implemented by two operations: c) SelectTheRelevantDateOfAgreement and d) MakeACopyOfTheDateOfAgreement. The relevant date is known as a result of the operation read relevant date presented in Fig. 10.23. As we model each requirement in separate diagram, we described the selection of the relevant date as a more general operation using Database and Agreement objects. Once the date retrieved from the agreement, the next operation is d) which results in a new object StolenDateFromAgreement. The new Object is added to the Anti-object model (described in the sequel of the thesis).

The last step of the attack method is to Achieve [StoreStolenDate]. The related operation is StoreStolenDateInDatabase, taking as input the StolenDateFromAnAgreement and resulting in storing the stolen date into the AttackerDatabase. The AttackerDatabase Object is a new Object of the Anti-object model. The operationalization of Achieve [StoreStolenDate] is presented in Fig. 10.25.

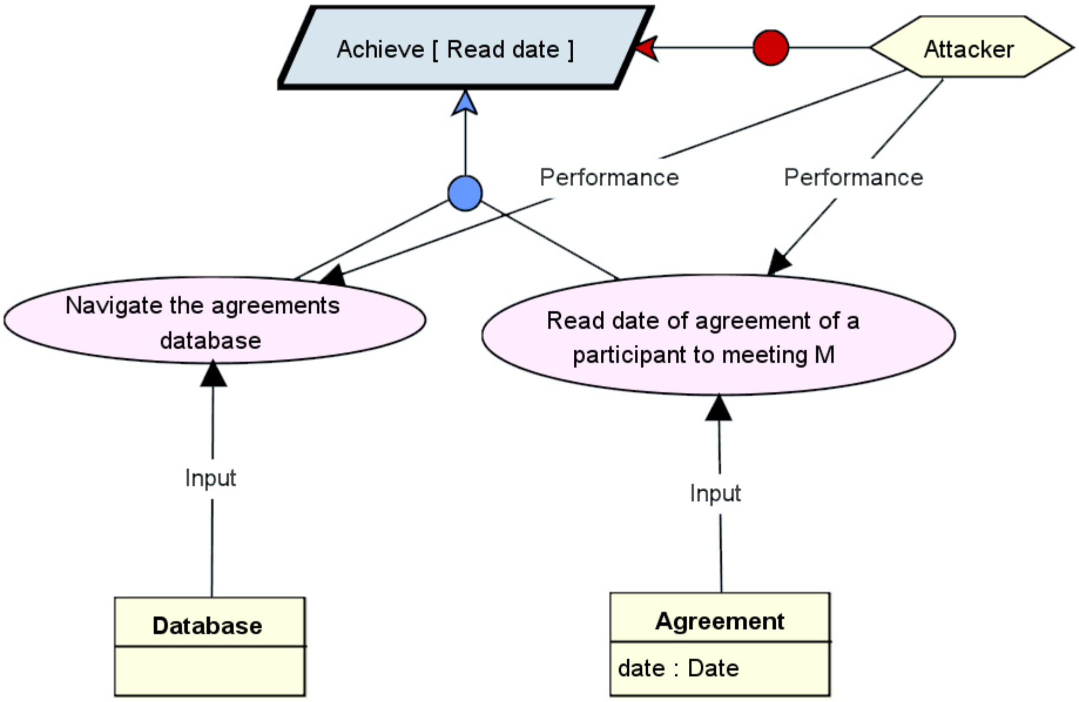


Figure 10.23: Operationalization of the attacker's requirement `Achieve [ReaDate]`

The Anti-object model is depicted in Fig. 10.26. Besides `StolenDateFromAgreement` and `AttackerDatabase`, it contains two other Objects: the `MeetingAttacker` – the actor who performs the attack against the system – and the `UnauthorizedParticipant` – actor to whom dates are revealed.

We are now entering the last step of the modelling of the case study using KeS. Since we have investigated a potential attack against the system-to-be, we need to consider how to deal with the security breach. We choose to mitigate the risk consisting in the `DisclosureOfDateOfAgreements` by using cryptographic procedures on the agreement's data. We build a new goal model Fig. 10.27 as a copy of Fig. 10.18 containing a new requirement `Achieve [AgreementsAreEncrypted]`. We operationalize this new requirement as the `UseCryptographicProcedures` operation as depicted in Fig. 10.28. It uses, as inputs, `Agreement` object and `CryptographicKeys` – new Object added to object model – and it produces an `EncryptedAgreement` – new Object too.

The object model contains two non described Objects: the `Firewall` Object and the `Access Control List` Object. They are used to treat other security breaches against availability and integrity criteria. We do not discuss these breaches.

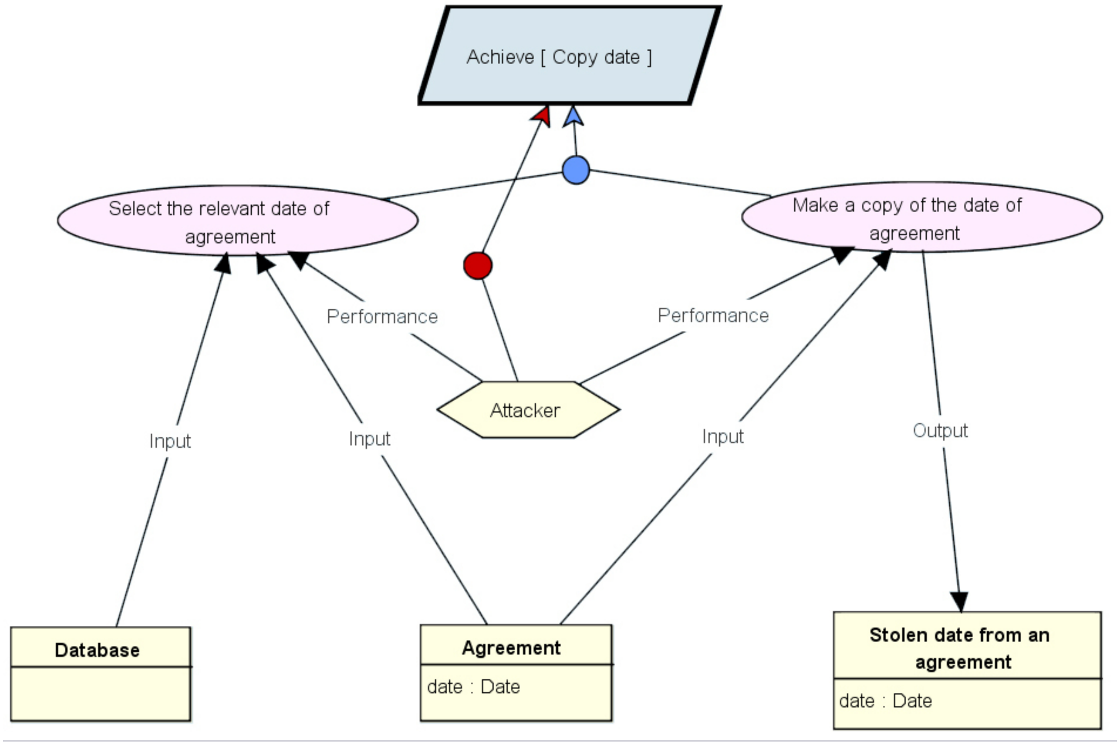


Figure 10.24: Operationalization of the attacker's requirement `Achieve [CopyDate]`

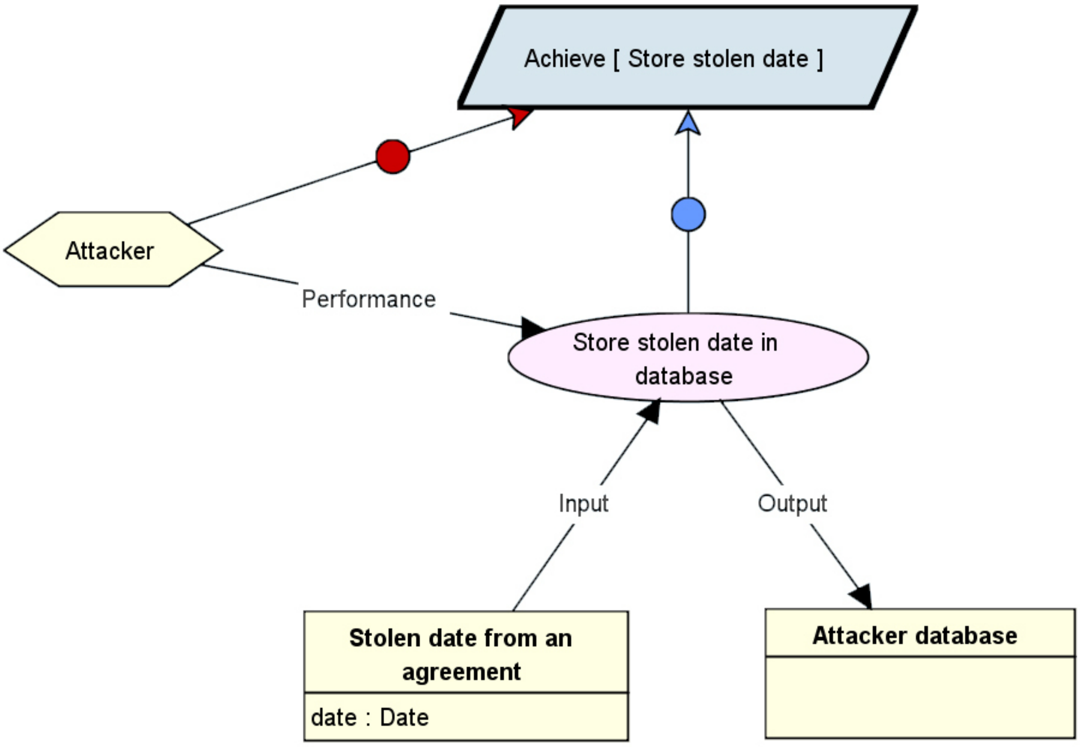

Achieve [Store stolen date]
Attacker
Performance
Store stolen date in database
Input
Output
Stolen date from an agreement
date : Date
Attacker database

Figure 10.25: Operationalization of the attacker's requirement `Achieve [StoreStolenDate]`

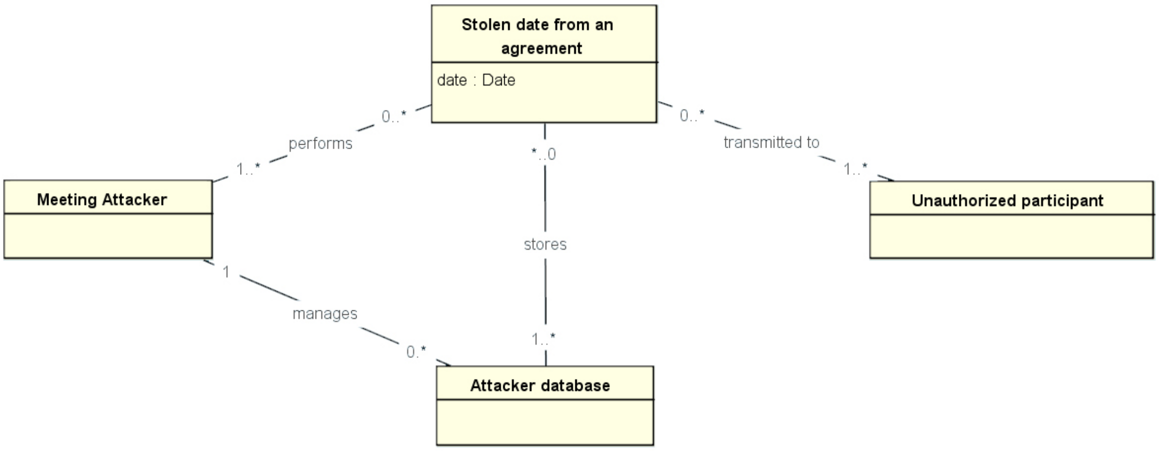


Figure 10.26: Anti-object model for the privacy attacker

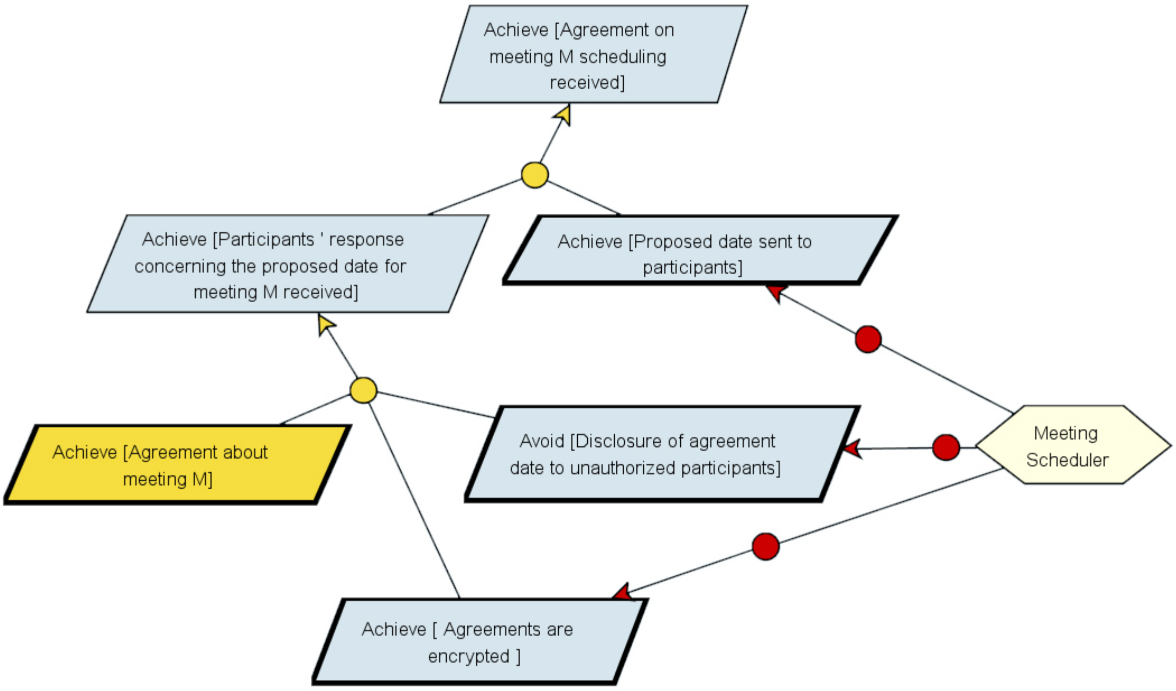

Achieve [Agreement on meeting M scheduling received]
Achieve [Participants ' response concerning the proposed date for meeting M received]
Achieve [Proposed date sent to participants]
Achieve [Agreement about meeting M]
Avoid [Disclosure of agreement date to unauthorized participants]
Meeting Scheduler
Achieve [Agreements are encrypted]

Figure 10.27: Goal model extended with countermeasures against privacy attack

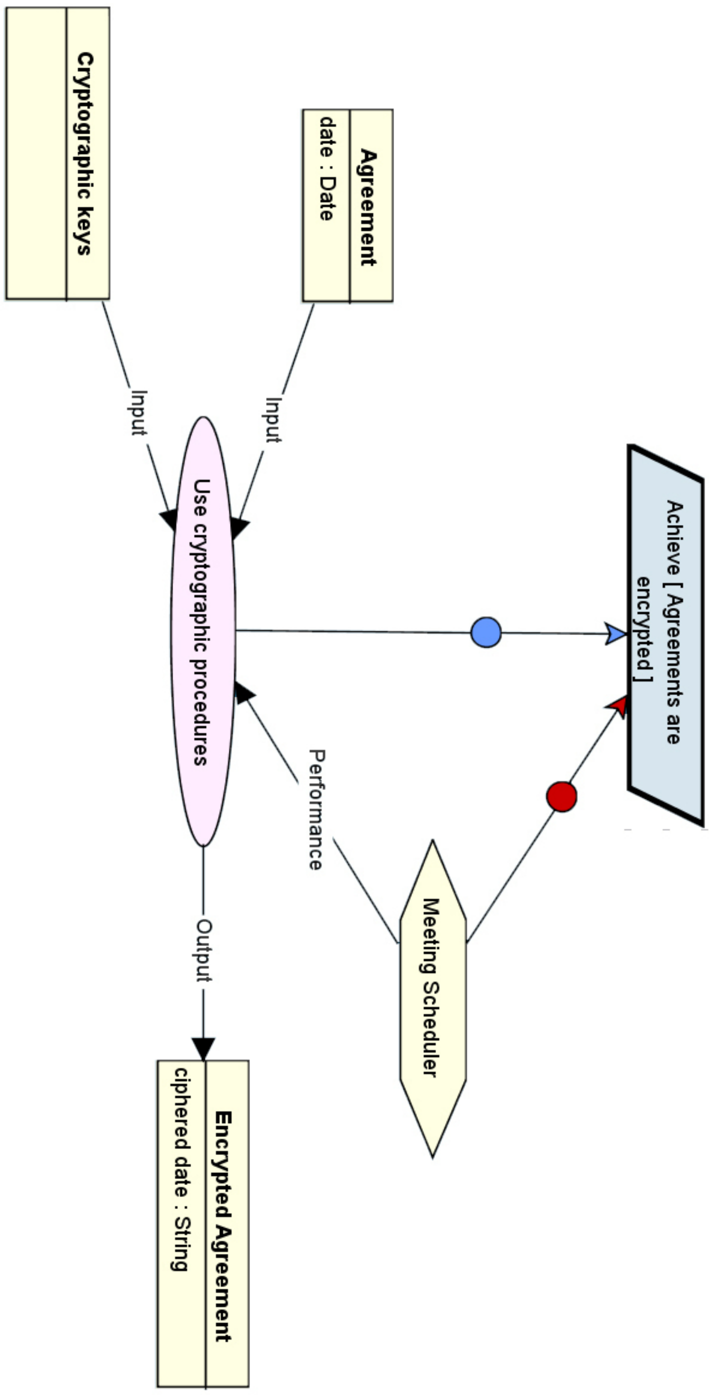


Figure 10.28: Operationalization of the requirement `AgreementsAreEncrypted`

10.3 Exploitation of the Case Study

After the modelling of the Meeting scheduler system using the secure TROPOS and KeS languages, we can analyse models that have been produced. First we detail the objectives of the exploitation of the case study and then we discuss the models obtained.

10.3.1 Analysis of the Case Study

We now discuss the models produced in secure TROPOS and in KeS. We have selected a set of relevant elements of the Meeting scheduler case study in Table 10.1 and, for each line, we indicate how the concept was modelled in secure TROPOS and KeS respectively. The discussion follows Table 10.1 line by line. We first describe how an element is modelled in the secure TROPOS language and after in the KeS language.

(e1)(e2)(e3) The three first relevant elements of the case study are the three actors of the system: the Meeting initiator, the Meeting participant and the Meeting scheduler. Each of them is represented in secure TROPOS using the concept of actor (Fig. 10.1) and in KeS using the concept of Agent Fig. 10.14. The two first Agents are equivalent to ISSRM Business assets while the third corresponds to an IS asset.

(e5) (e6) (e7) Participants can also give their agreement to a proposed date. As for preferred and excluded dates, `Agreement` (Fig. 10.5) is a Resource in secure TROPOS and an Object in KeS (Fig. 10.19). It corresponds to a Business asset. The same explanation can be given for the proposed date which is modelled using the same kind of concept (Fig. 10.5, Fig. 10.21) but in it this case, it is equivalent to an IS asset.

(e4) The Meeting initiator wants the meeting to be scheduled. This is modelled in secure TROPOS using a Goal `MeetingBeScheduled` (Fig. 10.1) and in KeS using also a Goal `MeetingBeScheduled` (Fig. 10.15). This element of the case study is an ISSRM IS asset.

(e8) One of the security concerns of the Meeting scheduler is the privacy criterion. A Softgoal `Privacy` Fig. 10.13 is used in secure TROPOS; in KeS it is labelled `PrivacyOfTheMeetingSchedulerSystem`. It corresponds to a security criterion.

(e9) Another element of the Meeting scheduler that is equivalent to the ISSRM security criterion is the constraint imposing that agreements can only be used by participants to meeting M. It is modelled as a security Constraint and/or a Softgoal in secure TROPOS (Fig. 10.6). There is no equivalent of the security Constraint in KeS but we have a KeS Softgoal `PrivacyOfTheMeetingSchedulerSystem`.

(e10) The attacker against the privacy of the system is represented as an Actor called `PrivacyAttacker` in secure TROPOS Fig. 10.12 and as an Agent `Attacker` in KeS Fig. 10.23. This attacker is aligned on the ISSRM threat agent concept.

(e11) The vulnerability (also ISSRM vulnerability) that is exploited by the described attack against privacy is the fact that agreements are not encrypted in the Meeting scheduler system. In Fig. 10.12, we see that this vulnerability is designed as a Belief `AgreementsAreNotEncrypted` in secure TROPOS, and in Fig. 10.22 as a Domain property `AgreementsAreNotEncrypted`.

(e12) The attack method (equivalent to ISSRM attack method) consists in disclosing agreements and, more precisely, the date of a specific agreement. This is modelled in secure TROPOS by two Plans, `ReadAgreements` and `SendToUnauthorizedPeople`, and the *attacks* relationship (see Fig. 10.12). In KeS, we have to look at a set of Operations `NavigateTheDatabaseAgreements` and `ReadDateOfAgreementOfAParticipantToMeetingM` Fig. 10.23, `MakeACopyOf-`

`TheDateOfAgreement` (Fig. 10.24) and `StoreStolenDateInDatabase` (Fig. 10.25).

(e13) An attacker against privacy wants to disclose agreements to unauthorized participants. This is equivalent to the ISSRM concepts of Threat. In secure TROPOS, it is described as a unique Goal `DisclosureOfAgreements` (Fig. 10.12) just as in KeS where we have the Requirement Achieve [DisclosureOfAgreementDateToUnauthorizedParticipants] (Fig. 10.22).

(e14) The system is exposed to the cause of the risk of a disclosure attack. As we can see this can be aligned with the ISSRM notion of Cause of the risk. The concept of a secure TROPOS Threat is used to model this cause of the risk (Fig. 10.8). On the other hand, this concept is mixed with the KeS Threat.

(e15) The elaboration of countermeasure begins by taking a decision about the response to an identified risk. In our case, we choose to mitigate the risk by adding a new security requirement. The decision has no construct in secure TROPOS, nor in KeS. However, the KeS documentation explicitly speaks about the process of taking decision when facing of a risk. In the ISSRM domain model, it is called the Risk treatment decision.

(e16) The envisaged (ISSRM) security requirement in the case of the risk against privacy is the use of cryptographic procedures. We can represent it in secure TROPOS with a Plan `PerformCryptographicProcedures` (Fig. 10.13) and in KeS as an Operation `UseCryptographic-Procedures` (Fig. 10.28).

The ISSRM notion of control has no specific concept neither in secure TROPOS, nor in KeS. But in the two languages, it is the iterative aspect of their methods that make the countermeasures of an iteration, the new controls and assets of the next iteration.

None of the elements of the meeting scheduler case study is modelled in secure TROPOS (respectively in KeS) using a concept X that is aligned on the ISSRM concept Y and in KeS using a concept Z that is not aligned with Y. Hence, our alignment of concepts of metamodels of these two languages is not invalidated by our case study.

	Elements of case study	**Secure TROPOS modelling**	**KeS modelling**	**ISSRM domain model**
e1	Meeting initiator	Actor [Meeting Initiator]	Agent [Meeting Initiator]	Business asset
e2	Meeting participant	Actor [Meeting Participant]	Agent [Meeting Participant]	Business asset
e3	Meeting scheduler	Actor [Meeting Scheduler]	Agent [Meeting Scheduler]	IS asset
e4	Initiator wants meetings be scheduled	Goal [Meeting Be Scheduled]	Goal [Meeting Be Scheduled]	IS asset
e5	Participants preferred and excluded dates	Resource [Preferred And Excluded Dates]	Object [Excluded Dates] + Object [Preferred Dates]	Business asset
e6	Agreement for meeting M	Resource [Agreement (for meeting M)]	Object [Agreement (for meeting M)]	Business asset
e7	Proposed date for meeting M	Resource [Proposed Date]	Object [Proposed Date]	IS asset
e8	System must ensured privacy of its data	Softgoal [Privacy]	Softgoal [Privacy Of The Meeting Scheduler System]	Security criterion
e9	Agreements can only be used by participants to M	Security constraint [Only Used By Participants To M]	–	Security criterion
e10	Attacker of privacy	Actor [Privacy Attacker]	Agent [Attacker]	Threat agent
e11	Agreements are not encrypted	Belief [Agreements Are Not Encrypted]	Domain property [Agreements Are Not Encrypted]	Vulnerability
e12	Attack system by disclosing agreements	Plan [Read Agreements] + Plan[Send To Unauthorized Participants] + *attacks* link	Operation [Navigate The Database Agreements] + Operation [Read Date Of Agreement Of A Participant To Meeting M] + Operation [Make A Copy Of The Date Of Agreement] + Operation [Store Stolen Date In Database]	Attack method

e13	Attacker of privacy wants to disclose agreements to unauthorized participants	Goal [Disclosure Of Agreements]	Requirement [Achieve[Disclosure Of Agreement Date To Unauthorized Participants]]	Cause of the Risk and Threat
e14	The system is exposed to the risk of a disclosure attack	Threat [Disclosure Attack]	–	Risk
e15	The risk of disclosure agreements is mitigated by encrypting agreements in the scheduler system	–	Anti-goal mitigation	Risk treatment decision
e16	To perform cryptographic procedures to mitigates risk of disclosure agreements	Plan [(S) Perform Cryptographic Procedures]	Operation [Use Cryptographic Procedures]	Security requirement

Table 10.1: Modelling of main elements of meeting scheduler case study using the secure TROPOS and KeS languages

10.3.2 Observation of the Case Study

By analysing our Meeting Scheduler case study, we can formulate several findings:

- all elements of the Meeting scheduler, that we modelled either in secure TROPOS either in KeS, used concepts of the metamodel of the corresponding language;
- all instances of the concepts of the metamodels are aligned with the same ISSRM concept as the concept it is an instance of. So a concept and its instance (in secure TROPOS or in KeS) are aligned with the same ISSRM concept. This validates our alignment as presented in Tables 9.2 and 9.3;
- secure TROPOS and KeS use notion of Goal but with an underlying semantic that differs for each of them. In secure TROPOS, the Actor who **owns** the Goal, which is what he wants. In KeS, the Agent is **responsible for** the achievement of the Goal. Thus, the goal is not what he wants, but what he is responsible for;
- the modelling process is very different in secure TROPOS and in KeS. The manner the analyst has to tackle the problem is focused on the notion of Actors and their inter-dependencies in secure TROPOS, while in KeS, he has to find the main Goal of the system-to-be and refine it until he reaches Requirements. Agents responsible for achieving these Requirements are then elicited;
- secure TROPOS and KeS do not focus on the same level of modelling. Secure TROPOS limits its scope to Plans of a quite high level of abstraction. On the other hand, KeS uses Operation model where Requirements are decomposed into steps that form the process to

fulfill the Requirement. This is a great distinction when we analyse the attack method of possible attackers as KeS attack methods are more refined than in secure TROPOS;

- the label of a Goal, a Requirement or an Object in KeS plays a crucial role to determine if these concepts are used with concerns for security. We find that having such an important distinction (security concerns) only expressed by the words used to describe a concept is insufficient and error-prone. Indeed, labels are written in natural language which is one of the most ambiguous way to communicate. The use of the **(S)** tag in secure TROPOS improves this situation but remains related to only the sole label of the construct;
- the notion of ISSRM Vulnerability has an equivalence in KeS (Domain property) but not really in secure TROPOS. As we discussed previously, Beliefs used to represent Vulnerabilities do not have the same semantic as for Vulnerabilities;
- KeS does not allow to consider a Goal as an ISSRM asset. But based on the experience of the Meeting scheduler case study, it seems that this concept should be added to the set of KeS concepts that can be considered as ISSRM assets.

10.3.3 Threats to Validity

The ideal case study is quite impossible to reach. A case study always suffers of limitations – also called threats to its validity. General limitations of case studies are, for example:

- reduction of the domain of the IS in order to fit the scope of the problem to treat;
- increased focus on positive results and vague discussion of negative outcomes.

In the case of the Meeting Scheduler System, we point out another threat which is that **we** modelled the case study using the secure TROPOS language and the KeS language. Ideally this work should be done by two distinct teams (or analysts) without being aware of how the Meeting Scheduler is designed in another language. Indeed, in this case, we played the triple role of designer, analyst and judge of the modelling of the Meeting Scheduler and the analysis of alignment between secure TROPOS and KeS with the ISSRM domain model. The objectivity is thus not always simple to reach. This situation is reflected by a low level of refinement of some parts of the Meeting scheduler using a language that offers the possibility to be more precise. So it is not always easy to know if the level of details of a model is due to the language itself or to keep comparable concepts.

10.4 Summary

In order to validate our alignments of the concepts of the secure TROPOS and the KeS languages with those of the ISSRM domain model, we modelled the Meeting Scheduler case study – based on [Yu97, FFFvL97, Let01] – using the secure TROPOS and the KeS languages. We summarise how relevant elements of the Meeting Scheduler are modelled using these two security modelling languages. By analysing the summary, we validate our previous alignments. Finally, we discussed threats to the validity of these alignments due to the inherent limitations of case studies and to the way we elaborated our Meeting Scheduler System.

Part IV

Conclusions

Chapter 11

Conclusions and Future Work

This chapter aims to present our conclusions on the results and findings of this thesis, and discuss future work that can be envisaged according to outcomes of the thesis.

11.1 Conclusions

At the beginning of this thesis, we addressed three problems that were stated as objectives of this work:

1. to investigate relationships between ISSRM domain model [MMHD07] concepts that were elicited in previous works in order to complete and improve the ISSRM domain model;

2. to analyse how existing security modelling languages – such as KeS [vL04], secure TROPOS [MJF06] – consider security concerns. We analyse what are the concepts of the ISSRM domain model that are treated in security modelling languages;

3. to elicit potential improvements of the ISSRM domain model – depending on the results of the analysis of security modelling languages – and also potential improvements to security modelling languages in order to make them supporting ISSRM concepts.

We thus choose to elicit ISSRM relationships from security and risk related sources in order to complete the ISSRM domain model. To fulfill this objective, we investigate source documents of security and risk related sources – for example, EBIOS v2 [DCS04], ISO/IEC 27001 [ISO05] and Firesmith [Fir03] – to elicit all relationships between concepts of each source. By analysing and summarising discovered relationships, we introduced associations between concepts of the ISSRM domain model.

Once relationships added on the ISSRM domain model, we analysed security modelling languages KeS and secure TROPOS in order to align them with the ISSRM domain model and explicit the covering of concepts of these languages and concepts of ISSRM domain model covers each. By this way, we addressed our second objective.

To fulfill our first objectives, we described the fundamental concepts in the understanding of the security and risk management domain. Then we survey the most relevant security and risk related sources. The way we achieve our second objective is presented in the research method (see Chapter 4) where we contributes to steps 2 and 3. Outcome of step 2 is a glossary gathering all relevant parts of the source documents that confirm the existence of relationships. We use it to enhance the ISSRM domain model. We then performed a part of the step 3 that corresponds to the elicitation of the mutual covering of ISSRM domain model and security modelling languages.

We validate our alignments by elaborating a case study based on the Meeting Scheduler System. This leads to discussion of the limits of the validity. Finally we exploited results of alignments to fulfill our third objective, that is, making suggestions to the enhancements of ISSRM domain model and improvements of security modelling languages.

Conclusions of this thesis are grouped according to stage of the thesis that they concern. First, we present conclusions on the method we used to fulfill our objectives and then on the results of the thesis.

11.1.1 Conclusions on the Method used to fulfill the Objectives of the Thesis

Conclusions on the method we choose to achieve the objectives of the thesis are:

- when aligning concepts, this process needs to rely on a **very detailed and precised** understanding of the investigated languages. The differences between the alignment of the KeS language as we performed first (see Appendix 3) and the alignment described in Chapter 9 illustrate this point. It is absolutely necessary to analyse the metamodel of investigated languages;

- as we discussed, results have to be validated. The usage of case studies are a mean to achieve this validation but the assessment by the experts of the domain reinforce the validity of the results.

11.1.2 Conclusions on the results of the Thesis

We now considerate conclusions on the results of the thesis:

- security in IS is not optional anymore. It has to be address during the whole development of a IS;

- at this time, there is no method that addresses all security aspects in a consistent and structured way. Thus reinforce the need of a reference document as the ISSRM domain model;

- moreover, imposing a new security modelling languages seems to be hopeless. Practitioners are rather inclined to keep their existing methods and languages;

- we contributed to the enhancement of the ISSRM domain model by eliciting relationships between its concepts. With these improvements, the ISSRM domain model can now be considered as reference document for practitioners who elaborate secure IS. It can be used as a set of guidelines to ensure that all security concepts of the ISSRM domain model are taken into account when modellers design system. Such a document can improve the situations presented in the previous conclusions;

- decomposing the ISSRM domain model into three blocks – asset, risk and countermeasures – allows practitioners to focus only on one block at once. Moreover, when they investigate a specific block, they know which are concepts that need to be taken into account;

- even if the ISSRM domain model seems to be sufficiently mature, some of its concepts need to be clarified. For example, the concept of Cause of the risk that is defined as "the combination of a threat and one or more vulnerabilities". Only defining this concept as the combination of two other does not really express its semantic.

11.2 Future Work

We split future work into two separate concerns. The first is about the development of the ISSRM domain model while the second concerns the enhancement of the secure TROPOS and the KeS languages.

11.2.1 Development of the ISSRM domain model

One of the most paramount questions to answer in the future is: *Will the ISSRM domain model be supported by*:

- a new language developed especially for it;
- a set of existing security modelling languages in which only some concepts will be retained and the corresponding analysis process will be kept. Gathering all the retained concepts will lead to the full syntax of the ISSRM domain model and using all parts of analysis processes will form the process to analyse the ISSRM domain model;
- a unique existing modelling language extended by new concepts that are not present at this time in the languages. The extension will produce a full covering of the ISSRM domain model concepts;

So there are three alternatives that can be considered. However, according to our conclusions (see previously) the ISSRM domain model should be presented as a reference document gathering all relevant security concepts that have to be taken into account when building secure systems. Thus future work on the ISSRM domain model should aim to define guidelines and/or processes that can be applied by a modeller in order to ensure specific security aspects in a system-to-be.

11.2.2 Future Investigations in the Secure TROPOS and the KeS Languages

As previously said, the results of our alignment can be used to enhance the secure TROPOS and the KeS languages. We gave remarks, at the end of Chapter 10, about lacks in the process or in the concepts of these two languages. Some suggestions from improvement are:

- to label components of models respecting a specific pattern. For example, the first term of the label of Goals in secure TROPOS and KeS should be a name. On the other hand, a verb should be the first term of labels of secure TROPOS Plan and KeS Operation. This difference makes clearer the meaning of objective of a Goal and of the meaning of process of a Plan and an Operation;
- we need to differentiate elements of the system-to-be that are related to the security. Secure TROPOS introduced the (S) tag to indicate that a element is considered with respect to security. The same kind of distinction should be introduced in the KeS language;
- secure TROPOS needs a some extra concepts to fully support the concepts of the ISSRM notion. If we consider the ISSRM Vulnerability, this concept cannot be fully supported by the secure TROPOS belief (reasons are described in Chapter 10). We suggest to create a new construct to modelled the ISSRM Vulnerability;
- to build some patterns in each languages in order to pre-design concepts while modelling. In this way, the analyst will be proposed to complete the labels of the predefined constructs that represents the ISSRM concept that he wants to express.

As we can see, the domain of the ISSRM modelling is not yet worn out and a lot of work is still to be done.

Bibliography

[ADA01] Christopher J. Alberts, Audrey J. Dorofee, and Julia H. Allen. OCTAVE Catalog of Practices, Version 2.0. *TECHNICAL REPORT CMU/SEI-2001-TR-020 ESC-TR-2001-020*, 2001.

[AG07] Opdahl A.L. and Berio G. *A Roadmap for UEML.* Springer, 2007.

[A.L07] Opdahl A.L. *The UEML Approach to Modelling Construct Description.* Springer, 2007.

[AS/04] AS/NZS 4360. Risk management. *Australian/New Zealand Standard 4360*, 2004.

[BGG$^+$04] P. Bresciani, P. Giorgini, F. Giunchiglia, J. Mylopoulos, and A. Perini. Tropos: An Agent-Oriented Software Development Methodology. *Journal of Autonomous Agents and Multi-Agent Systems*, 2004.

[BMMZ06] Volha Bryl, Fabio Massacci, John Mylopoulos, and Nicola Zannone. Designing Security Requirements Models through Planning. *CAISE*, 2006.

[BP89] F Bodart and Y Pigneur. Conception Assistées des Systèmes d'Information. *Masson*, 1989.

[BPSMa] Davide Bertolini, Anna Perini, Angelo Susi, and Haralambos Mouratidis. Recommandation Z.151: URM – Goal-oriented Requirement Language (GRL). *Contribution for the AOSE TFG meeting, February 28th, Ljubljana, Slovenia, collocated with the Second AgentLink III Technical Forum (AL3-TF2).*

[BPSMb] Davide Bertolini, Anna Perini, Angelo Susi, and Haralambos Mouratidis. The Tropos visual modeling language. A MOF 1.4 compliant meta-model. *Contribution for the AOSE TFG meeting, February 28 th, Ljubljana, Slovenia, collocated with the Second AgentLink III Technical Forum (AL3-TF2).*

[CKM02] J. Castro, M. Kolp, and J. Mylopoulos. Towards Requirements-Driven Information Systems Engineering: The Tropos Project. *Information Systems*, 2002.

[CLU98] CLUSIF. Marion (méthodologie d'analyse des risques informatique et d'optimation par niveau). *CLUSIF*, 1998.

[CLU04] CLUSIF. MEHARI V3 Concepts and Mechanisms. 2004.

[Com05] Common Criteria. Common criteria for information technology security evaluation version 3.1. *ISO/IEC 15408*, August 2005.

[Con03] Insight Consulting. Référentiel CRAMM. September 2003.

[DCS04] DCSSI. Expression des Besoins et Identification des Objectifs de Sécurité: sections 1 - 5. 2004.

[Dir89] Direction des Constructions Navales. Melisa (methode d'evaluation de la vulnerabilite residuelle des systèmes d'information). *Direction des Constructions Navales*, 1989.

[Dub96] J.-C Dubois. L'analyse du risque : une approche conceptuelle et systémique. *Chenelière-McGrawHill*, 1996.

[DvLF93] A. Dardenne, A. van Lamsweerde, and S. Fickas. Goal-directed Requirements Acquisition. *Science of Computer Programming*, 1993.

[EN03] Steve Easterbrook and Bashar Nuseibeh. Fundamentals of Requirements Engineering. 2003.

[FFFvL97] Martin S. Feather, Stephen Fickas, Anthony Finkelstein, and Axel van Lamsweerde. Requirements and Specification Exemplars. *Automated Software Engineering*, 1997.

[Fir03] Donald G. Firesmith. Common Concepts Underlying Safety, Security and Survivability Engineering. *CMU/SEI-2003-TN-033*, 2003.

[GMNS02] Paolo Giorgini, Haralambos Mouratidis, E. Nicchiarelli, and R. Sebastiani. Reasoning with Goal Models. *the Proceedings of the 21st International Conference on Conceptual Modeling (ER2002), Tampere, Finland*, 2002.

[GMZ06] Paolo Giorgini, Haralambos Mouratidis, and Nicola Zannone. Modelling Security and Trust with Secure Tropos. In *Integrating Security and Software Engineering: Advances and Future Vision*, pages 160–189. IDEA, 2006.

[Hey] Patrick Heymans. Cours d'analyse et modélisation des systèmes d'information, fundp, namur.

[Hey05] Patrick Heymans. Analyse et Modélisation de Systèmes d'Information. 2004-2005.

[HMLN06] Charles B. Haley, Jonathan D. Moffett, Robin Laney, and Bashar Nuseibeh. A Framework for Security Requirements Engineering. In *Proceedings of the 2006 international workshop on Software engineering for secure systems*, pages 35–42, New York, NY, USA, 2006. ACM Press.

[HR04] D. Harel and B. Rumpe. Meaningful modeling: What's the semantics of "semantics"? *Computer 37 (10) pp. 64-72*, 2004.

[HSDP06] Patrick Heymans, Germain Saval, Gauthier Dallons, and Isabelle Pollet. A Template-Based Analysis of GRL. In *Advanced Topics in Database Research*, pages 124–146. IDEA, 2006.

[ISO02] ISO/IEC Guide 73. Risk management – vocabulary – guidelines for use in standards. *ISO/IEC Guide 73*, 2002.

[ISO04] ISO/IEC 13335-1. Information technology – security techniques – management of information and communications technology security – part 1: Concepts and models for information and communications technology security management. *ISO/IEC 13335-1*, 2004.

[ISO05] ISO/IEC 27001. Information technology – security techniques – information security management systems – requirements. *ISO/IEC 27001*, 2005.

[Jac92] Ivar et *al* Jacobson. Object-oriented software engineering: a use case driven approach. *Addison-Wesley*, 1992.

[Jac01] Michael Jackson. Problem Frames: Analyzing and structuring software development problems. *Addison-Wesley*, 2001.

[KSR] H. Kefi, B. Safar, and C. Reynaud. Alignement de taxonomies pour l'interrogation de sources d'information hétérogènes.

[LBD02] Torsten Lodderstedt, David A. 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 (UML '02)*, pages 426–441. Springer-Verlag, London, UK, 2002.

[Let01] Emmanuel Letier. Reasoning about Agents in Goal-Oriented Requirements Engineering. May 2001.

[LNI$^+$03a] L. Lin, B.A. Nuseibeh, D.C. Ince, Mickael Jackson, and J.D. Moffett. Introducing Abuse Frames for Analysing Security Requirements. 2003.

[LNI$^+$03b] Luncheng Lin, Bashar Nuseibeh, Darrel Ince, Michael Jackson, and Jonathan Moffett. Introducing Abuse Frames for Analysing Security Requirements. In *Proceedings of the 11th IEEE International Requirements Engineering Conference*, 2003.

[Maytz] Nicolas Mayer. Cours : La gestion des risques de sécurité des SI : présentation du domaine et des concepts, Metz.

[MCN97] J Mylopoulos, L Chung, and B Nixon. Representing and using nonfunctional requirements: a process-oriented approach. *IEEE Trans. Softw. Eng., 18(6), 448-497*, 1997.

[MF99] John Mcdermott and Chris Fox. Using abuse case models for security requirements analysis. In *ACSAC '99: Proceedings of the 15th Annual Computer Security Applications Conference*, Washington, DC, USA, 1999. IEEE Computer Society.

[MG04] Haralambos Mouratidis and Paolo Giorgini. Enhancing secure Tropos to effectively deal with security requirements in the development of multiagent systems. *the 1st International Workshop on Safety and Security in Multiagent Systems*, AAMAS 2004.

[MG07] Haralambos Mouratidis and Paolo Giorgini. Secure Tropos: Extending i* and Tropos to model security throughout the development process. *accepted for publication in MIT Press*, 2007.

[MGGP02] Haralambos Mouratidis, Paolo Giorgini, Manson Gordon, and Ian Philp. A Natural Extension of Tropos Methodology for Modelling Security. *the Proceedings of the Agent Oriented Methodologies Workshop (OOPSLA 2002), Seattle-USA, November*, 2002.

[MGM] Haralambos Mouratidis, Paolo Giorgini, and Gordon Manson. Using Tropos Methodology to Model Integrated Health Assessment System.

[MGM03] Haralambos Mouratidis, Paolo Giorgini, and Gordon Manson. Integrating Security and Systems Engineering: Towards the Modelling of Secure Information Systems. *n the 15th Conference On Advanced Information Systems Engineering (CAiSE*03). Austria*, 16 - 20 June 2003.

[MH06] Nicolas Mayer and Jean-Philippe Humbert. La gestion des risques pour les systèmes dínformation. *Misc 24*, 2006.

[MHM06] Nicolas Mayer, Patrick Heymans, and Raimundas Matulevičius. Design of a Modelling Language for Information System Security Risk Management. Technical report, CRPHT Luxembourg and FUNDP Namur, October 2006.

[MHM07] Nicolas Mayer, Patrick Heymans, and Raimundas Matulevičius. Design of a modelling language for information system security risk management. In Colette Rolland, Oscar Pastor, and Jean-Louis Cavarero, editors, *Proceedings of the First International Conference on Research Challenges in Information Science (RCIS'07), Morocco*, pages 121–132. Ouarzazate, Morocco, 2007.

[MHO07] Raimundas Matulevičius, Patrick Heymans, and A.L. Opdhal. Comparing GRL and KAOS using the UEML Approach. In *Enterprise Interoperability II. New Challenges and Approaches*, pages 77–88. Springer-Verlag, 2007.

[MJF06] H. Mouratidis, J. Jurjens, and J. Fox. Towards a Comprehensive Framework for Secure Systems Development. In E. Dubois and K. Pohl, editors, *Proceedings of the 18th International Conference on Advanced Information Systems Engineering (CAiSE'06)*, pages 48–62. Springer-Verlag, 2006.

[MMHD07] Nicolas Mayer, Raimundas Matulevičius, Patrick Heymans, and Eric Dubois. A Framework for Analysing the Interoperability of Security Modelling Languages with Risk Management Methods. *submitted for publication*, June 2007.

[Moo06a] D. Moody. Dealing with "map shock": A systematic approach for managing complexity in requirements modelling. In *in Proceedings of the Twelth International Workshop on Requiremetns Engineering Foundations for Software Quality (REFSQ'06), IEEE Computer Society, Washington, DC, USA pp. 148-157*, 2006.

[Moo06b] D. Moody. What makes a good diagram: improving the cognitive effectiveness of diagrams in IS development. In *Proceedings of the Fifteenth International Conference in Information Systems Development, IEEE Computer Society, Washington, DC, USA pp. 148-157*, 2006.

[Mou04] Haralambos Mouratidis. *A Security Oriented Approach in the Development of Multiagent Systems: Applied to the Management of the Health and Social Care Needs of Older People in England.* PhD thesis, University of Sheffield, U.K., 2004.

[Mou06] Haralambos Mouratidis. Integrating Security and Information Systems Engineering: The Secure Tropos approach, October 2006. University of Namur Seminar.

[MPL06] Raimundas Matulevičius, Heymans Patrick, and Opdahl A. L. *Comparison of Goal-oriented Languages using the UEML Approach.* ISTE, 2006.

[MPL07] Raimundas Matulevičius, Heymans Patrick, and Opdahl A. L. *Comparing GRL and KAOS using the UEML Approach.* Springer-Verlag, 2007.

[Pie07] Frank Piessens. The challenge of building secure software. Presentation, 2007.

[SGF02] Gary Stoneburnern, Alice Goguen, and Alexis Feringa. Risk Management Guide for Information Technology Systems: Recommendations of the National Institute of Standards and Technology. *NIST Special Publication 800-30*, 2002.

[SHF04] Gary Stoneburnern, Clark Hayden, and Alexis Feringa. Engineering Principles for Information Technology Security (A Baseline for Achieving Security), Revision A. *NIST Special Publication 800-27 Rev A*, 2004.

[SO04] Guttorm Sindre and Andreas L. Opdahl. Eliciting security requirements with misuse cases. *Springer-Verlag London*, 2004.

[Som01] Ian Sommerville. *Software Engineering Sixth Edition.* Addison Wesley, 2001.

[vL03] Axel van Lamsweerde. The KAOS Meta-model: Ten Years After. *Technical report, Université Catholique de Louvain*, 2003.

[vL04] A. van Lamsweerde. Elaborating Security Requirements by Construction of Intentional Anti-Models. 2004.

[vLL] A. van Lamsweerde and Emmanuel Letier. Handling Obstacles in Goal-Oriented Requirements Engineering.

[VML$^+$07] F. Vraalsen, T. Mahler, M. S. Lund, I. Hogganvik, F. den Braber, and K. Stülen. Assessing enterprise risk level: The coras approach. *in Advances in Enterprise Information Technology Security, D. Khadraoui and Francine Herrmann, Idea Group Reference*, March 2007.

[Yu95a] E Yu. Modelling Strategic Relationships for Process Reengineering. *Ph.D. Thesis, University of Toronto, Departement of Computer Science*, 1995.

[Yu95b] Eric Yu. *Modelling Strategic Relationships for Process Reengineering.* PhD thesis, University of Toronto, 1995.

[Yu97] E.S.K Yu. Towards modeling and reasoning support for early-phase requirements engineering. *In RE'97: Proceedings of the 3^{rd} IEEE International Symposium on Requirements Engineering (RE'97) (p.226)*, Los Alamitos, CA: IEEE Computer Society, 1997.