Universidade do Minho
Escola de Engenharia

Roberto Manuel Dias Machado

Validating Common Criteria Documentation using Alloy

Outubro de 2011

Validating Common Criteria Documentation using Alloy
Roberto Manuel Dias Machado
UMinho | 2011

Universidade do Minho
Escola de Engenharia

Roberto Manuel Dias Machado

Validating Common Criteria Documentation using Alloy

Tese de Mestrado
Mestrado em Engenharia Informática

Trabalho efectuado sob a orientação do
Professor Doutor Manuel Bernardo Barbosa

Outubro de 2011

Declaração

Nome: Roberto Manuel Dias Machado

Endereço Electrónico: rmdmachado@gmail.com

Telefone: 916748994

Bilhete de Identidade: 13441167

Título da Dissertação: Validating Common Criteria Documentation using Alloy

Orientador: Doutor Manuel Bernardo Barbosa

Ano de conclusão: 2011

Designação do Mestrado: Mestrado em Engenharia Informática

Este trabalho for realizado no âmbito de uma bolsa de investigação financiada através do projecto de prestação de serviços de consultoria STORK, estabelecido entre a empresa Multicert - Serviços de Certificação Electrónica, S. A. e a Universidade do Minho.

Universidade do Minho, 31 de Outubro de 2011

Roberto Manuel Dias Machado

Acknowledgements

First of all I want to thank my supervisor Professor Manuel Bernardo Barbosa for the research guidance and also for all the suggestions that improved this dissertation. I am grateful to Professor Alcino Cunha for the advice with Alloy.

Many thanks to the Multicert company for allowing me to participate in the CESeCore Project and to the CESeCore Technical Committee for the always interesting meetings. A special thank you to Nuno Santos who was always available to help. Also thanks to European Union Seventh Framework Programme, for granting me with a scholarship which made most of this work possible.

I take this opportunity to thank my colleagues at CeSIUM, that have helped me direct this student association for the last two years. I must also thank my colleagues at AAUM for their understanding when I was less available. A special thank you as well to my mates at Group Buddies.

I want to give a special thank you to all my friends that always support me as well, and at the same time I want to apologize for not being so present in the last year. A special acknowledgement to João that helped me with my doubts regarding the english language.

Thank you to my friend and partner of long work nights André. A special thank you to Marta for all the kind support.

Family, I don't have words to describe how much I'm thankful to all of you, especially my parents and my brother. Thank you for all the support since day zero of my life.

Last but certainly not least, I would like to thank to my aunt and uncle André and Angelina, and my cousins Ricardo and Luís, for their support in this five years that I have spent in Braga. You not only gave me a roof, you also gave me a second family.

"Rejoice with your family in the beautiful land of life!"
- Albert Einstein
Dedico este trabalho aos meus tios André e Angelina.

Resumo

O software criptográfico é frequentemente associado a componentes críticos de sistemas que requerem esforços adicionais para melhorar o seu nível de segurança. A certificação de software criptográfico é necessária para garantir que este vai de encontro às garantias de segurança desejadas. Esta certificação tem que seguir *standards* tais como Common Criteria (CC) para assegurar que a avaliação foi feita de uma forma rigorosa.

CC é um *standard* internacionalmente reconhecido que permite a certificação de software de segurança com relevo para a produção de provas do correcto desenvolvimento do software. Estas provas geralmente tomam a forma de uma série de documentos que podem custar tempo e dinheiro às organizações que pretendem conseguir certificações de software.

Esta tese de mestrado trata da análise de todo o processo de certificação CC. Em específico, trata da documentação produzida como prova para uma avaliação. Pesquisa foi efectuada para compreender o processo de certificação, a documentação produzida, o *Protection Profile* e o *Security Target*, que mereceram especial atenção e, por último, técnicas que possam melhorar o desenvolvimento de documentação e a sua validação.

Geralmente os métodos formais são utilizados em procedimentos de certificação para obter fortes garantias de correcção para os níveis de segurança mais elevados. No trabalho aqui apresentado mostramos que os métodos formais também podem ser usados para aumentar o rigor do processo de certificação ao nível da análise documental. Usamos a linguagem de especificação *Alloy* para modelar os conceitos fundamentais de uma certificação de segurança de acordo com CC, e subsequentemente para mostrar como estes modelos podem ser usados para validar a consistência da documentação do *Protection Profile* e do *Security Target*.

Durante este trabalho estivemos envolvidos no projecto de certificação de um *core* criptográfico num pacote de software para certificação digital, o projecto CESeCore[1]. Olhamos para a documentação do CESeCore para desenvolver os nossos modelos e para validar a dita documentação. Para este projecto escrevemos ainda um documento que descreve todo o processo de certificação do CESeCore.

Finalmente, desenvolvemos um protótipo daquilo que pode vir a ser uma ferramenta para ajudar na criação de documentos para a certificação CC. Este protótipo usa os modelos *Alloy* como bases e permite-nos carregar documentos produzidos pela certificação CC, modificá-los, analisá-los e validá-los. Esta pode vir a ser uma ferramenta interessante para a indústria envolvida em certificação CC.

[1] www.cesecore.eu

Abstract

Cryptographic software is typically associated with critical components of systems that require additional efforts in order to improve their level of assurance. Therefore, the certification of cryptographic software is necessary to ensure that it meets the desired guarantees. This certification must follow standards such as Common Criteria (CC) to ensure that the evaluation was conducted in a rigorous way.

CC is an internationally recognized standard that allows for security software certification with the focus on producing evidence of the software development. This evidence usually takes the form of a series of documents that can cost the organizations that intend to achieve software certifications time and money.

This master thesis focuses on the analysis of the whole CC process of certification. Specifically, all the documentation produced as evidence for the evaluation. Research is conducted to understand the process of certification, the documentation produced, the Protection Profile and the Security Target, both of which deserved special attention, and lastly techniques that could improve the development of documentation and its validation.

Typically, formal methods are enforced in certification procedures in order to obtain strong correctness guarantees for the highest levels of assurance. In the presented work we show that formal methods can also be used to increase the rigor of the certification process at the documentation analysis level. We use the Alloy specification language to model the fundamental concepts of a security certification according to the CC, and then show how these models can be used to validate the consistency of Protection Profile and Security Target documentation.

During this work we have been involved in a cryptographic certification process in a software package for digital certification, project CESeCore[2]. We look at the documentation for CESeCore to develop our models and to validate said documentation. For this project we have also written a document that describes all the process of the CESeCore certification.

Finally, we have developed a prototype of what could be a tool to aid in the creation of an ion document for the CC certification. This prototype uses the Alloy models as bases and allows us to load documents produced for CESeCore certification, modify these documents, analyze them and validate the models. This could be an interesting tool for the industry involved in CC certifications.

[2] www.cesecore.eu

Contents

List of Figures

Listings

List of Tables

Acronyms

AA	Alloy Analyzer
Alloy	Alloy Modeling Language
CA	Certification Authority
CC	Common Criteria
CCRA	Common Criteria Mutual Recognition Agreement
CEM	Common Evaluation Methodology
CESeCore	CESeCore Project
CIMC	Certificate Issuing and Management Components Family of Protection Profiles
CRL	Certificate Revocation List
CTCPEC	Canadian Trusted Computer Product Evaluation Criteria
DOD	Department of Defense
EAL	Evaluation Assurance Level
ED	Embedded Device
FC	Federal Criteria
IDE	Integrated Development Environment
IT	Information Technology
ITSEC	Information Technology Security Evaluation Criteria
JVM	Java Virtual Machine
GUI	Graphical User Interface
Keon CA	RSA Keon CA System
Keon ST	RSA Keon CA System version 6.7 Security Target
MIT	Massachusetts Institute of Technology
MLS	Multi Level Security

NIAP	National Information Assurance Partnership
NIST	U.S. National Institute of Standards and Technology
NSA	National Security Agency
OSP	Organizational Security Policie
PCA	Polymorphous Computing Architecture
PDF	Portable Document Format
SAR	Security Assurance Requirement
SFR	Security Functional Requirement
SPD	Security Policies Definition
SPM	Security Policy Modeling
TCSEC	Trusted Computer System Evaluation Criteria
TLS	Top Level Specification
TOE	Target of Evaluation
TSF	TOE Security Function
TSFI	TOE Security Function Interfaces
UML	Unified Modeling Language
XML	Extensible Markup Language

Chapter 1

Introduction

Software plays a major role in critical systems such as cars, airplanes, health care devices, telecommunication and several other mainstream appliances. However, much of the software present in modern society does not give us correctness or security guarantees. Historically, this has led to incidents of disastrous proportions [11].

Over the years, great efforts have been made in order to find the best practices for achieving security, safety and dependability certificates to be delivered to users along with software systems. This work is related to this effort of establishing a consistent regulation to ensure that critical software is built in agreement with safety, security and reliability standards [31].

Common Criteria (CC) [22] is a standard developed for Information Technology (IT) products evaluation, ensuring a consistent way to build confidence in the security of software products. There are other standards oriented to software security certification, but CC are widely recognized and a requisite for several governmental institutions. CC standards contains a security vocabulary and a set of security methodologies so that customers, vendors and evaluators can all use the same terms and proceed in conformity with all the evaluation processes. It is important for the vendors that the customers have the software products validated by independent third-parties, confirming his claims about the product.

Security testing and evaluation standards provide a way to standardize product comparisons. This also introduces development and documentation methodologies that will improve the development of the products and the customers' level of assurance. CC is an internationally recognized standard that allows security software certification with focus on producing evidence on the software development.

The evaluation process in a CC certification can be divided into three phases: the preparation phase and project launch; the phase in which evidence for the evaluation is produced and the validation and certificate emission phase. Figure 1.1 transposes this process division. The second phase is a series of cycles of evidence production, evaluation, commentary, modification, resubmission and re-evaluation iterated until the evaluator is satisfied that the requirements for evaluation are complete.

This evidence usually takes the form of a series of documents, depending on the cycles presented before, that could cost the organizations that want to achieve software certifications time and money [42]. The number of iterations that one requirement could take depends on the quality of the documents and whether they are correctly formulated so as

to be presented to the evaluators.

Figure 1.1: CC Evaluation process

The amount of evidence that CC requires depends on the level of assurance desired with the evaluation. There are seven levels of assurance called Evaluation Assurance Level (EAL). In the highest levels of assurance, CC uses formal methods in some of its requirements to obtain rigorous correctness guarantees [38].

Formal methods are typically applied in software production to achieve a higher level in terms of correctness or security guarantees [9]. However, in this thesis, and following other works [5], we apply formal methods to the analysis of CC documentation.

The Alloy Modeling Language (Alloy)[1] is a language for describing structural properties [26]. The models written in Alloy are usually much smaller than the systems they model and yet they can be used to verify important properties of their underlying systems [44]. Alloy models can be analyzed through the Alloy Analyzer (AA) [25], a fully automatic tool built on top of a SAT solver to simulate models and check their properties. Alloy can be seen as a formal methods technique. In the end of this chapter we introduce Alloy in more detail.

1.1 Motivation

In software certification a great number of documents have to be produced as evidence of the evaluation process. In the CC's case, where the evaluation focuses mainly on the production of evidence in the form of documents, this work is always tedious and can become expensive with a great number of human resources occupation, even though the production of these documents is mostly a systematic process. The documentation produced for the CC evaluation process is also complicated to understand and vague on how its concepts relate [39].

CC concepts, sometimes abstract in their definitions, are an ample catalogue of threats, security objectives, security requirements, etc. When we are producing evidence for CC, in the form of a set of interrelated documents, it is necessary to clarify how all the concepts are transposed from each document to the others. To do this work manually can result in uncertainty regarding the correctness and sufficiency of the security objectives that are being certified [36].

The most important documents in the CC certification are the Protection Profiles and the Security Target. These two documents are interrelated since the Security Target is a subset of the Protection Profiles. This is an instantiation of the General Model concepts present on the CC, as we can see in Figure 1.2.

The use of Protection Profiles in the process of certification was pioneered by the CC as

[1] `http://alloy.mit.edu/`

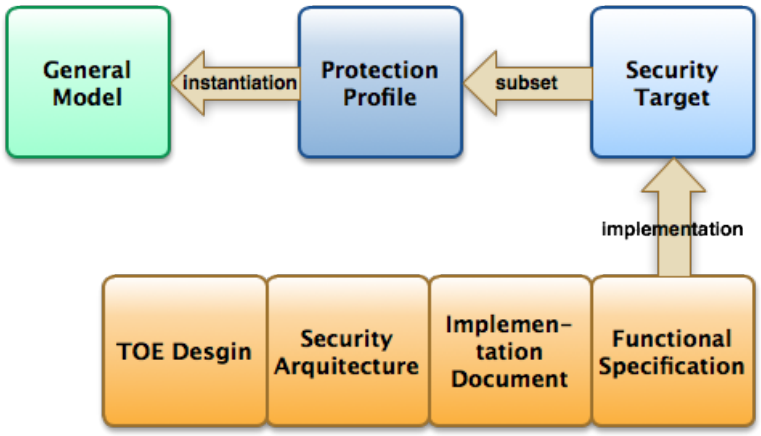

Figure 1.2: Documentation for Common Criteria Evaluation

it is a document produced for a family of products with the same characteristics. In it, all the elements used in a CC certification for that family of products are specified as an instantiation of CC concepts present in the General Model. Protection Profiles are implementation independent.

The Security Target is a subset of the Protection Profiles oriented towards software product implementation. In it all the concepts used in the CC certification are specified. All the others documents relative to the software product development are based on the information stated in the Security Target. We use Alloy to model this documentation structure and minimize the problems presented in this section.

A little More About Alloy

Specification languages are a type of formal language used in system analysis, requirements analysis and systems design [43]. As opposed to normal programming languages which are directly executable, formal languages are used along the whole system development process. With a specification language we can describe the system at a high level of abstraction, allowing representations of system properties in a way that is more understandable to humans. If the level of abstraction at which we use the specification language isn't high enough we cannot take advantage of these techniques. Some specification language examples are Alloy, Z, VDM, CASL, etc [29]. We present the Alloy language, which is used in this work to model and validate CC documentation.

Alloy was developed in the Massachusetts Institute of Technology (MIT) by Daniel Jackson in the late 90s for the purpose of abstract software design. It is a lightweight modeling language based on Z. Alloy supports the description of basic structure (graphically, or as textual declarations), as well as more intricate constraints and operations describing how structures change dynamically (both expressed as logical formulas) [26].

An Alloy model consists of a type declaration part, a number of formulas (facts), and an assertion. The AA checks the validity of the assertion taking into account a user-provided scope and an upper bound on the number of elements considered for each type [4]. In case the assertion is not valid, the analyzer produces a counterexample with symbolic values for each type and relation [12].

In Alloy types represent sets of atoms and are introduced using the signature construct. Atoms are something abstract and their type means nothing to the AA. These atoms establish relations between them and that is the key point in Alloy: every Alloy expression is a relation. Alloy is equipped with a set of operators similar to set theory operators and a set of quantifiers. Alloy also has a set of commands, non-parameterized constraints (assumptions) of the system, which are expressed as *facts*. These constraints are considered to be true at all times. Parameterized or reusable formulas expected to be used in different contexts are expressed as *predicates* and reusable expressions are expressed as *functions*. The properties to be checked are expressed as *assertions*. More details can be found elsewhere [27]. We explain in Section 4.1 how to build models in Alloy.

We use Alloy to model the various levels of abstraction present in the CC documentation and validate the main documents used as evidence in software certifications with CC. We chose Alloy for our work because it is a simple specification language that, when complemented with the AA, allows us to quickly build and analyze models.

1.2 Objectives

The main objectives for this work are several. Firstly, to study the CC standard in a way that would allow us to explain the whole certification process and how to elaborate a CC certification. Also a main objective is to understand how all the evidence is produced. With the objective of understanding how the evidence is produced comes the objective of understanding the problems associated with the elaboration of all the documents that need to be presented for validation during a CC certification. During this analysis we also intend to study ways to minimize this problem.

Another objective for this work is the use of Alloy in the software product certification process that could help its development in a correct way and that could help reduce the costs in the always expensive certifications. We want to use these techniques to analyze the CC certifications model and documentation. We also intend to use these same techniques to validate these documents.

This work's last assignment is to specify a prototype for a tool that could aid the process of document development for the CC. We also want to develop a proof of concept for this tool, since the development of the complete application is out of the scope of this masters dissertation.

1.3 Contributions

In this work we present an overview of the CC security certification standard and how we can use Alloy to help in the evidence development process. This work is included in the project developed by Multicert[2] in the CESeCore Project (CESeCore)[3]. In this project, Multicert, in a consortium with three more european companies, are conducting a software security certification, more specifically, a CC certification. As an example, along with the work for this dissertation, we look at the documentation of the RSA Keon CA System (Keon CA) digital certification and the CESeCore with CC.

[2] www.multicert.com
[3] www.cesecore.eu

In this work we analyze the Protection Profiles [35] used in the Keon CA and in the CESeCore, the Security Target of both certification, [30] and [37]. To analyze these documents we have built models of the concepts in CC and used them to represent the documentation used in CC certification. We have used Alloy to build these models.

We use Alloy to model the fundamental concepts of a security certification according to the CC. We model the information that we consider sensitive and validate it in the CC certification main documents. We want to be able to assertively answer several questions that have come up when analyzing the CC documents and validate the consistency of the main documents of CC: the Protection Profile and the Security Target. Other documents can be partially analyzed with these models and we can also infer information to include in the documents created from the Security Target. With this we are using formal methods techniques in a non-conventional way so as to analyze the rigor of the certification process at the documentation analysis level.

AA allows us to use the models and change them, quickly making changes and analyzing the documentation again. This, however, requires some knowledge of the Alloy language that not everyone has. With this in mind we believe that there was space for a tool to aid in the use of the alloy models. Another aspect that makes the need for a tool evident has to do with the difficulty felt in the elaboration of the evidence for the evaluation.

We have developed a prototype of what could be a tool to aid in the creation of documentation for the CC certification. This prototype uses the Alloy models as basis and allows us to load documents produced for CC certification, modify these documents, analyze them and validate these models. This could be an interesting tool for the industry involved in CC certifications.

1.4 Dissertation Outline

This work is divided into 6 Chapters. In the first chapter, the current one, we present an introduction to all the work undertaken for this dissertation.

Chapter 2:
presents an overview of the CC standard. We begin by presenting a brief history of the standard and its main goals. Afterwards, we present CC's framework and, in more detail, the EALs and the General Model. This chapter finishes with the CC's standard vision on formal methods and some works with this techniques on CC's certifications.

Chapter 3:
presents the CC documentation that is used as evidence in the certification processes. Firstly, we present the documentation structure of all the documents used in the CC certification. Afterwards, we present the documents considered for this work and what is interesting to analyze in the context of our modulation in Alloy in detail. The documents presented in detail in this chapter are the Protection Profile, the Security Target and the Functional Specification.

Chapter 4:
presents the modulation and validation of the CC documents presented in the previous chapter. Firstly, a brief tutorial on Alloy and the main characteristics of this

specification language is presented. To continue, a set of Alloy models that are used in the validation of the points specified in the chapter before for the main CC documents is presented.

Chapter 5:
presents the case studies that were used in this work. Initially, we present the case study for the RSA Keon CA System followed by the CESeCore Project. We also present the application of the Alloy models presented in the previous chapter in the CESeCore Project. This chapter ends with the presentation of an application to support our work with Alloy. We also present XMLs templates for the CC documents and parsers to work with this templates in this chapter.

In the last chapter we present this work's conclusion and what we expect to be future work, especially that which concerns with the development of the tool to aid in the production of theCC documentation.

Related Publications

A portion of the work presented in this dissertation has been previously published in the form of conference papers:

- Roberto Machado. Cryptographic Software Certification with Common Criteria Evaluation Assurance Level 4+. In *MI-STAR 2011*. 2011.
- Roberto Machado. Modeling and Validating Cryptographic Software Documentation for Common Criteria Certification. In *INForum*. 2011.

Chapter 2

The Common Criteria Standard

In this chapter we present a detailed overview of the CC standard, its process, all the concepts related with CC certification and an overview of the use of formal method techniques in the context of software certifications. CC is an internationally recognized standard for security certification and this very fact is one of its strengths.

So as to begin, we introduce a very brief history of the CC standard in Section 2.1. There are others standards for security certification. In Section 2.2 we present some advantages and disadvantages of using the CC.

The CC standard is a framework divided into three parts. We explain these three parts in Section 2.3. All the components needed for a CC certification are in them. The first part of the standard introduces all the concepts and terminology used during a CC certification. The second part introduces the security functional components that are available to specify the security requirements of a certification. The third part introduces the security assurance components that define what kind of guarantees a certification has to assure in terms of evidence to be delivered for evaluation. We also present the document used by the evaluators to understand how to make a proper evaluation.

CC evaluations are ranked in levels of assurance. In CCs, they are called Evaluation Assurance Levels (EALs). This is useful to comprehend how deep of an evaluation was made in terms of certification requirements. In Section 2.4 we present those levels.

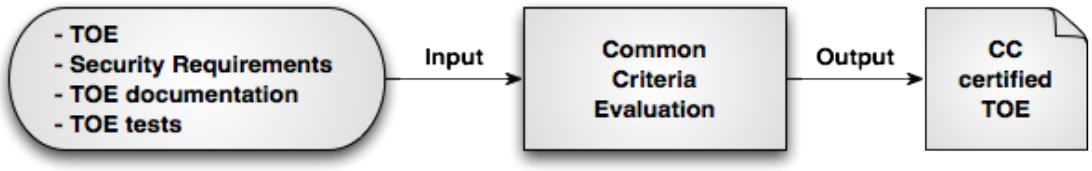

Figure 2.1: CC evaluation process overview

We also introduce an overview of the CC evaluation process. The process is based on the evaluation of evidence that serves as input and results in a certificate, which ensures that a certification was completed successfully. Figure 2.1 shows the flow of this evaluation [34].

In the second part of this chapter we present a brief overview of formal methods and how these techniques are included in the CC standard. We also present some work related with

the use of formal methods in the certification of software. In Section 2.7 we present an example of how formal methods are used in a CC certification successfully.

2.1 Brief History

For many years, people related to software security development tried to elaborate a standard to certify the security of software. The U.S. Defense Department, the main driver since the early days, launched the Trusted Computer System Evaluation Criteria (TCSEC) in the late 70's, also known as the *Orange Book*, the first standard for evaluating IT security. Since the release of the *Orange Book*, several attempts have been made to standardize such descriptions, especially for operating systems platforms. The trend has been toward increasing power and complexity [21]. A major result of this standardization activity is the Common Criteria, that was developed out of these previous approaches:

- Trusted Computer System Evaluation Criteria (TCSEC) - United States Department of Defense (DOD) 5200.28 Standard also know as the *Orange Book*;
- Federal Criteria (FC) a draft approach by the U.S. National Institute of Standards and Technology (NIST) and the National Security Agency (NSA) to replace TCSEC in 1992;
- Information Technology Security Evaluation Criteria (ITSEC) - developed in the early 1990s by France, Germany, the Netherlands and the U.K.;
- Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) - the Canadian standard first developed in 1993 from the U.S. DOD standard.

The *Common Criteria for Information Technology Security Evaluation* (abbreviated as Common Criteria or CC) is an international standard for computer security certification. It is currently in version 3.1, which was released in 2009. The rationale behind CC is to formalize the language used by customers, developers and security evaluators to have the same understanding when security requirements are specified and evaluated.

At this moment, 26 countries are participating in the CC program, Portugal isn't one of these countries. Participating countries sign the Common Criteria Mutual Recognition Agreement (CCRA) which basically states the conditions for participation including the requirement that each member country recognize certificates issued from the other member countries. This is probably the greatest advantage of CC for vendors, since they certify their products only once and all CCRA members will recognize that certification. This was not possible with previous standards. Each CCRA member nation has a CC governing entity or *Scheme* responsible for managing the implementation and use of the CC in their country.

There are two types of CCRA member nations, the *certificate-authorizing* countries and the *certificate-consuming* countries. The first ones are countries that have been approved to issue the CC certificates, the second ones are countries that recognize the certificates but can not issue them.

The CC standards documents are downloadable from the Common Criteria Portal[1] website free of charge.

[1] `http://www.commoncriteriaportal.org/`

2.2 Common Criteria Goals

CC standards have several characteristics that define them as the leaders on the certification of security products. However, some critics define CC as nothing more than a pile of documentation, even though it can be proved to be valuable for vendors that want to make their products' security claims certified and profit from that certification.

As we present in the previous section, CC was developed from a set of other existing security evaluation standards, including the two more popular, the TCSEC from the USA and the standard European/Australian ITSEC. Its other main characteristic is its international recognition. Right from the beginning fourteen countries have signed an agreement to recognize this standard for high-quality IT security evaluations. As stated in Section 2.1, today there are 26 countries participating in the CC program.

The CC specify methods for the evaluation criteria as well as to conduct evaluations. The stringent standards that CC sets and maintains imply that all people involved in a certification follow that standard in a rigorous way. Any laboratory that wishes to conduct CC evaluations must be certified through a rigorous process, and must maintain that certification through periodic reinspection.

Another important aspect is that CC is an independent evaluation of security assurance. Evaluation testing results are submitted to the CC Scheme in the lab's host country for validation so as to insure that the evaluation process has been followed correctly, to enforce consistency across labs, and to prevent any financial motives from influencing the evaluation outcomes.

This set of characteristics already allows us to realize the value of CC, even though it also has some disadvantages. The most relevant is the cost of CC evaluations. An evaluation with CC can sometimes take years to complete and can also fail. A vendor, when deciding to evaluate his software product, must analyze it's viability and what EAL (see Section 2.4) to follow. Otherwise, the certification can be commercially non-compensatory.

2.3 The Common Criteria Framework

CC standard provides a framework to describe and evaluate security attributes of a product. In this standard, we find a language that provides us with a vocabulary to express customer security requirements and vendor product claims. Allied to this, CC standard also has guidelines for evaluators to perform consistent security evaluations. CC standards documents capture these descriptions in four parts:

- CC Part 1: Introduction and General Model
- CC Part 2: Security Functional Components
- CC Part 3: Security Assurance Components
- Common Evaluation Methodology

Reading the CC standard is a bit like reading a dictionary. It is important to have some guidance on how to look to at the documents and on what is contained in them. In the following section we provide an overview of each of the parts presented above.

2.3.1 CC Part 1: Introduction and General Model

CC Part 1 explains how the CC works. It is a document that aims to introduce people to the standard and should be read by everyone involved in CC certification. This document explains the objectives of the CC, the basic philosophy of the evaluation process, and definitions for components related to CC. An introduction to the basic security concepts necessary for evaluation of IT products is also given.

The introduction to Part 1 explains that CC provides a standard set of IT security characteristics for functionality and assurance. These security characteristics are used during security evaluations allowing comparisons of the independent evaluations results. CC evaluations provide a specified level of confidence that the claimed security functionality will meet customer needs. With this, customers have more information so as to make an informed purchasing decision.

Part 1 contains a chapter dedicated to define the terms which are used in a specialized way throughout the CC. We can visit this chapter when we want to understand any concept that we find in the CC standard. This is an extensive dictionary of CC terms, 15 pages of terms, acronyms and abbreviations used in the standards. Some concepts aren't explained here, leaving its definition to appear in the specific context.

The CC is flexible in what to evaluate and is therefore not tied to the boundaries of IT products as they are commonly understood. Therefore, in the context of evaluation, the CC uses the term Target of Evaluation (TOE). TOE is defined as a set of software, firmware and/or hardware possibly accompanied by guidance. After defining the TOE, CC Part 1 also presents the evaluation context and describes the audience to which the evaluation criteria are addressed.

In the last chapters, it defines the various operations by which the functional and assurance components given in CC Part 2 and CC Part 3 may be tailored through the use of permitted operations, and introduces the General Model of CC, the Protection Profile and Security Targets. These are explained in detail bellow.

Security Functional and Assurance Requirements

The Security Functional Requirements describe the desired behavior expected of a TOE and are intended to meet the security objectives stated in a Protection Profile or Security Target. The Security Assurance Requirement (SAR) are requirements to establish a standard way of expressing the assurance requirements for the TOEs. The scale for rating assurance for TOEs is called Evaluation Assurance Levels and is presented bellow.

General Model

The general model descriptions in CC Part 1 allows us to understand the philosophy behind the CC evaluation process. CC is based on providing customers assurance by evaluating products and evidence rather than through theoretical modelling and simulation. In the CC process, products are evaluated after it has been developed, since CC uses "after-the-fact" evaluation.

The General Model is a main component for our work. The building of models to analyze

the documentation of the CC evaluation is based on them. In Section 2.6 we explain the General Model and the concepts involved in detail.

Protection Profile

The Protection Profile's main goal is to state a security problem rigorously for a given collection of systems or products, known as the TOE and to specify security requirements to address the problem without dictating how these requirements will be implemented. A Protection Profile is a combination of threats, security objectives, assumptions, Security Functional Requirement (SFR), SAR and rationales. It also specifies generic security evaluation criteria to substantiate a given family of information system products vendor's claims. In order to get a product evaluated and certified according to the CC, the product vendor has to define a Security Target which may comply with one or more Protection Profile.

Vendors can evaluate their products against the requirements present in one of these applicable Protection Profile or they can make their own claims and have those claims evaluated in the CC evaluation process. Nevertheless, some governmental agencies have established policies requiring some products to only be evaluated against an applicable Protection Profile.

In Section 3.3 Protection Profiles are presented in more detail.

Security Target

The Security Target is a security specification for a software product. This is the central document, typically provided by the developer of the product, that specifies security evaluation criteria to substantiate the vendor's claims for the product's security properties. A Security Target defines information assurance security requirements for the given information system product, which is called the TOE. A Security Target is a complete and rigorous description of a security problem in terms of TOE description, threats, assumptions, security objectives, SFRs, SARs, and rationales [23].

In Section 3.4 we present the Security Target in more detail.

2.3.2 CC Part 2: Security Functional Components

As already mentioned, the CC standards have a standard vocabulary. This allows customers to express their security needs and vendors to manage the security characteristics of their products in a consistent manner. In this CC Part 2, Security functional components are defined as the basis for the security functional requirements expressed in a Protection Profile or a Security Target.The SFR are standard identifiers and descriptions of security features about the product itself. These requirements describe the desired security behavior expected of a TOE and are intended to meet the Security Objectives as stated in a Protection Profile or an Security Target. These security features are the countermeasures to the threats.

SFRs are organized into eleven major groups called Security Functional Classes. We can easily understand that these classes define all of the potential security functions a product

Table 2.1: Security Functional Classes

Class ID	Class Name	Description
FAU	Security Audit	Security event audi record handling
FCO	Communications	Non-repudiation or origin and receipt
FCS	Cryptographic Support	Cryptographic operation and key management
FDP	User Data Protection	Protecting user data transferred within the TOE
FIA	Identification and Authentication	User identification and authentication
FMT	Security Management	Management of TOE security functions, attributes and data
FPR	Privacy	Includes anonymity, pseudo-nymity, unlink ability and unobservability
FPT	Protection of the TSF	Protect TSF data. Recovery and self-test
FRU	Resource Utilization	Includes fault tolerance, service priority and resource allocation
FTA	TOE Access	Restricts access to TOE
FTP	Trusted Paths/Channels	Trusted communications paths and channels with the TSF

could provide. The Security Functional Classes from CC Part 2 are listed in Table 2.1:

The SFR classes contain several families, components and elements. All of this information is contained in CC Part 2, and are the result of decades of effort to identify, describe and categorize effective countermeasures to a wide variety of IT threats. Using the "dictionary" of security functions we can remove the ambiguity of using different terms for the same requirement. To accommodate a wide variety of product types and security features, SFRs may include operations that further refine the definitions.

The SFRs may have dependencies. If an SFR has any dependence on another SFR and you include this SFR, you must also include those dependencies.

2.3.3 CC Part 3: Security Assurance Components

The CC standard's main objective is to improve customer confidence (assurance) in the security of the IT products they purchase and use. Security features are one way to provide that confidence, but providing security in the product development process improves it even more. This CC Part 3 defines the assurance requirements of the CC, with this we can gain assurance through the evaluation of development process.

CC uses a set of independent evaluation techniques to provide assurance. We apply these techniques through the production of evidence for the evaluation. These evaluation techniques include:

- Examination and analysis of the soundness of vendor development procedures and

Table 2.2: Security Assurance Classes

Class ID	**Class Name**	**Description**
ADV	Development	Product architecture, functional specifications, internals, implementation and design
AGD	Guidance Documents	Operation and installation guides
ALC	Lifecycle Support	Configuration management, delivery, development, flaw remediation security
ATE	Test	Test coverage, depth. Functional and independent testing
AVA	Vulnerability Assessment	Vulnerability analysis
ACO	Composition	Composition evidence, testing, vulnerability analysis and rationale

processes

- Verification of the application and adherence to those processes
- Analysis and comparison of the various TOE design evidence
- Comparison of the evidence against security requirements
- Validation of claims
- Analysis of user guidance
- Analysis of product tests
- Performance of independent functional tests
- Vulnerability analysis
- Penetration testing

As in CC Part 2, where SFR are used to describe security features about the product itself, CC Part 3 and the security assurance components relate to the product development process. SAR cover areas regarding the secure development, delivery and deployment of the product. As in the CC Part 2 with the SFRs, the Security Assurance Classes are organized in eight classes. The Security Assurance Classes are listed in table 2.2.

As the SFRf, the SAR classes are segmented into families, components and elements. For example, the ADV, Development class is broken down into the following families:

- Security Architecture (ADV_ARC)
- Functional Specification (ADV_FSP)
- Implementation Representation (ADV_IMP)
- TOE Security Function Internals (ADV_INT)

- Security Policy Modeling (ADV_SPM)
- TOE Security Design (ADV_TDS)

Each family is further decomposed into components and elements that provide the details of the specific requirements. The notation for SAR also has D, C and E element identifiers, that respectively denote requirements for Developers, Content and Evaluators. Developers are the vendors or actual product developers, Content is the evidence documentation produced for evaluation and finally Evaluators are the actual, independent, third-party evaluators.

For example, ADV_ARC1.1, Security Architecture description, requires that the developer design and implement the TOE so that the security features of the TSF cannot be bypassed. A TSF can be something like a user authentication mechanism to detect or prevent unauthorized access to the system. Developers that use structured design methodologies will be able to claim that they support this requirement. The content and presentation elements are the evidence (documentation) presented to the evaluators to prove that the requirements have been met. With this the evaluators shall confirm the claims made by developers. In Section 2.3.4 we will present the Common Evaluation Methodology (CEM) document that goes into this in much more detail, from the evaluator's perspective.

It's important to understand this SAR requirements. When defining with assurance requirements what you want to achieve in your certification, you must keep in mind that the evaluator will analyze every component of that requirement. So you may save yourself some time and expense if you read the evaluator's requirements in not only CC Part 3 but in CEM as well to find out precisely what the evaluator is going to be looking for to satisfy each requirement. With this you will be prepared to correspond with all the evidence necessary.

EAL is a set of collections of SARs with seven different levels of evaluation assurance. In Section 2.4 we explain the EALs in more detail.

2.3.4 Common Evaluation Methodology

The Common Methodology for Information Technology Security Evaluation (CEM) is a companion document to the Common Criteria for Information Technology Security Evaluation (CC). CEM is the standardized set of instructions to the evaluation labs on how to evaluate the different assurance requirements. The CEM defines the minimum actions to be performed by an evaluator in order to conduct a CC evaluation using the criteria and evaluation evidence defined in it. The CEM guides the evaluator on what to look for and to what depth should he evaluate the evidence. According to the EAL, the evaluator will examine the evidence with different depths. The CEM does not define evaluator actions for certain high assurance CC components, where there is as of yet no generally agreed on guidance.

Evaluators are instructed to check, examine, record and report. These terms are defined by the CEM as:

check - generate a verdict by simple comparison;

examine - generate a verdict by analysis, using evaluator expertise;

Table 2.3: EAL Description

EAL	Description
EAL1	Functionally Tested
EAL2	Structurally Tested
EAL3	Methodical Tested and Checked
EAL4	Methodically Designed, Tested and Reviewed
EAL5	Semi-formally Designed and Tested
EAL6	Semi-formally Verified, Designed and Tested
EAL7	Formally Verified and Tested

record - retain a written description of procedures, events, observations, insights and results in sufficient detail to enable the work performed during the evaluation to be reconstructed at a later time;

report - include evaluation results and supporting material in the Evaluation Technical Report or in an Observation Report.

These definitions may be subjected to interpretations and variatons based on the evaluators' perspective and experience. They could be based on the background of the evaluator. Even though the CC Part 3 describes evaluator actions for each SAR, the CEM provides much more detailed instructions on what they have to do to satisfy the evaluation requirement. This level of detail helps ensure greater consistency across evaluators and across inter nation Schemes.

2.4 Evaluation Assurance Levels

The assurance requirements define the level of the evaluation effort applied to the secure development, delivery and deployment processes by the product developer. This can refer both to the amount of evidence produced and reviewed to prove the security claims and to the scope of aspects of the product development processes..

Each EAL embodies a recommended set of assurance requirements: the higher the EAL, the more assurance the product has. The intent of higher levels is to provide higher confidence that the system's main security features are reliably implemented. We can use EALs to choose which assurance requirements we want to satisfy. Table 2.3 has a description of the seven levels. A customer can look at the EAL of a certain product and see what kind of effort was put into the assessment of the security claims made by the vendor. The range of EAL usually used to certify products goes from EAL2 to EAL4. EAL4 is the highest level that is commercially advantageous and that is mutually recognized internationally. The belief is that evaluations higher than EAL4 require unique and proprietary evaluation techniques that cannot be duplicated or standardized internationally. It is also complicated to find laboratories that will evaluate these levels.

In some cases, the evaluation may be augmented to include assurance requirements beyond the minimum required for a particular EAL. Officially this is indicated by following the EAL number with the word augmented and usually with a list of codes to indicate the additional requirements. To simplify, vendors will often simply add a "plus" sign (as in

EAL4+) to indicate the augmented requirements. For example, in the project CESeCore, that is evaluated with EAL 4+, the SAR ALC_FLR.2, Flaw reporting procedures were added. As such, the EAL 4 would be denoted EAL4 augmented with ALC_FLR.2 or more informally EAL4+. Table 2.4 shows the requirements presented in standard EAL4.

EAL4 certification

In this section we present an overview of an EAL4 certification of a software product with CC. This is the assurance level which processes an interesting set of requirements that assure us that the whole process of product improvement was developed with proper security concerns. The choice of EAL4 is due to its balance between price level and security certification guarantee [24]. Here we present this EAL because it was the one that we studied in more detail along the work for the dissertation.

With EAL4 certification a software product was methodologically designed, tested and reviewed, which means that good practices of development where followed, the software was tested against the identified threats and the whole process was reviewed.

EAL4 provides assurance by a full Security Target and an analysis of the SFRs in that Security Target, using a functional and complete interface specification, guidance documentation, a description of the basic modular design of the TOE, and a subset of the implementation, to understand the security behavior. In what SARs are concerned, as we can see in Table 2.4, EAL4 covers six class of assurance.

The analysis is supported by independent testing of the TOE Security Function (TSF), evidence of developer testing based on the functional specification and TOE design, selective independent confirmation of the developer test results, and a vulnerability analysis.

EAL4 also provides assurance through the use of development environment controls and additional TOE configuration management including automation, and evidence of secure delivery procedures. This EAL represents a meaningful increase in assurance from EAL3 by requiring more design description, the implementation representation for the entire TSF and improved mechanisms and/or procedures that provide confidence that the TOE will not be tampered with during development.

In Chapter 5 two examples of real-case CC certifications with EAL4+ are presented.

More than EALs

It is important to note that not all families and components from CC Part 3 are included in the EALs. This is not to say that these do not provide meaningful and desirable assurances. Instead, it is expected that these families and components will be considered for augmentation of an EAL in those Protection Profiles and Security Targets to which they prove to be useful.

This notion of EALs represents the belief that assurance in terms of customer confidence is driven by evaluation and that more in-depth evaluation will give customers greater confidence. This means that, in theory, the higher the EAL, the greater the confidence the customer should have that the claims made by the vendor are true. The evaluations with higher EALs require more detailed evidence and are subjected to careful evaluation. These activities should represent greater proof of the vendor's claims.

Table 2.4: EAL4 Components

Assurance Class	Assurance Components
ADV: Development	ADV_ARC.1 Security architecture description ADV_FSP.4 Complete functional specification ADV_IMP.1 Implementation representation of the TSF ADV_TDS.3 Basic modular design
AGD: Guidance documents	AGD_OPE.1 Operational user guidance AGD_PRE.1 Preparative procedures
ALC: Life-cycle support	ALC_CMC.4 Production support, acceptance procedures and automation ALC_CMS.4 Problem tracking CM coverage ALC_DEL.1 Delivery procedures ALC_DVS.1 Identification of security measures ALC_LCD.1 Developer defined life-cycle model ALC_TAT.1 Well-defined development tools
ASE: Security Target evaluation	ASE_CCL.1 Conformance claims ASE_ECD.1 Extended components definition ASE_INT.1 Security Target introduction ASE_OBJ.2 Security objectives ASE_REQ.2 Derived security requirements ASE_SPD.1 Security problem definition ASE_TSS.1 TOE summary specification
ATE: Tests	ATE_COV.2 Analysis of coverage ATE_DPT.1 Testing: basic design ATE_FUN.1 Functional testing ATE_IND.2 Independent testing - sample
AVA: Vulnerability assessment	AVA_VAN.3 Focused vulnerability analysis

The security of a product is not measured by the EAL with which it was evaluated. A product evaluated with EAL3 isn't necessarily more secure than another evaluated with EAL4. This only means that the product evaluated with EAL4 will have more evidence collected during its evaluation process. There does seem to be a need for customers to more easily grade the security of products, but EALs are not the way to achieve this.

2.5 Common Criteria Process

We can see the the CC certification process in five phases, as is presented in Figure 2.2. The first phase that could be called Phase 0, has to do with the Pre-Evaluation Preparation. It is a key element for a successful evaluation. The activities for that phase include: researching customer requirements, understanding the CC standards, developing a compelling business case, managing the project's scope, allocating resources and selecting partners.

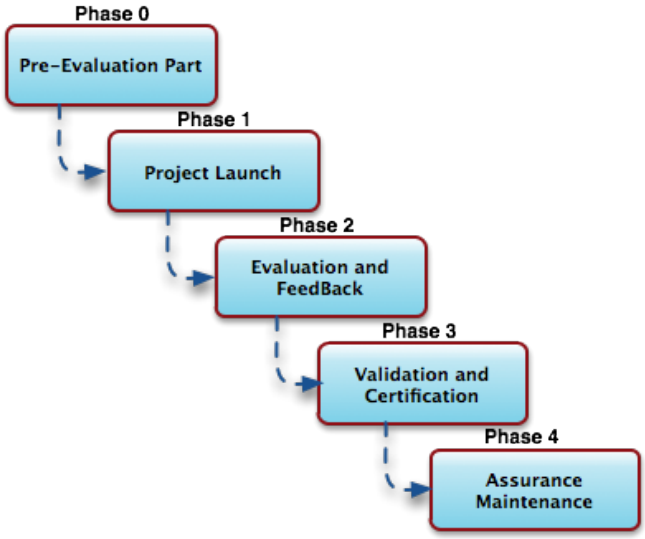

Figure 2.2: CC evaluation process overview

The second phase, Phase 1, the Project Launch has to do with the kickoff of the evaluation, here the meetings with the evaluation entities and the terms of the evaluation are defined. In this phase the Protection Profile to be used is defined and the Security Target is elaborated. The third phase, Phase 2, the Evaluation and Feedback phase is a series of cycles of interaction with the evaluators that include activities such as evidence production, evaluation, modification, resubmission and re-evaluation. Figure 2.3 shows the workflow of these cycles, as something deliverable starts, is produced and submitted as evidence. This evidence is evaluated and an observation report is produced by the evaluators. That report announces if the evidence is accepted or rejected. If it is rejected, then it must be updated and submitted again. Otherwise, in case of acceptance, the work unit is complete. This phase is the longest of the whole and the more critical.

The fourth phase, Phase 3, Validation and Certification, is the last phase directly related with the evaluation. This is when the whole certification is reviewed by the validators and if everything is evaluated successfully the product is certified. The last phase mentioned here, Phase 4, Assurance Maintenance, has to do with the problem that arises from each certification only being valid for one version of the software product. As such, the vendor,

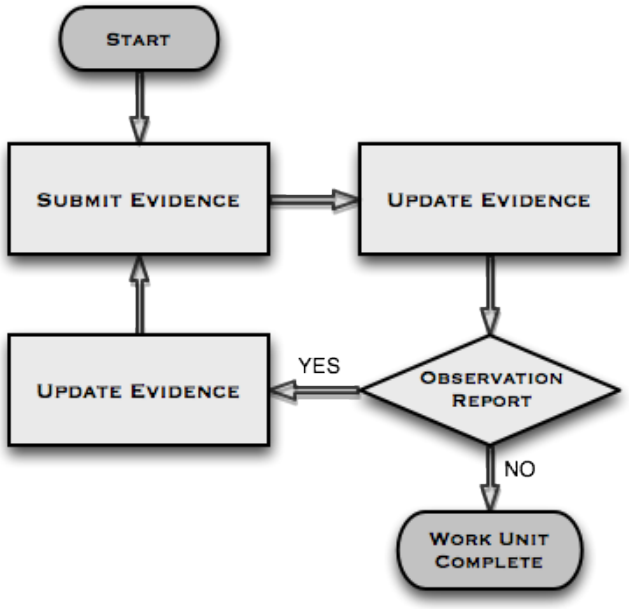

Figure 2.3: Evidence Evaluation Process

when revising his product, also has to revalidate it's certification, providing new evidence along with the new claims for the product. In the context of this dissertation we focus on Phases 1 and 2.

2.6 General Model

This is the model that explains how the concepts in CC are related. CC discusses security using a set of security concepts and terminology. To work with CC, submitting evaluations or evaluating them, an understanding of these concepts and the terminology is a prerequisite.

These concepts are quite general and not intended to restrict the class of IT security problems to which the CC is applicable. The General Model is a fundamental component for our work, since it is based on it that we build the models to analyze the CC evaluation documentation.

In order to understand the CC language, it is important to realize that IT organizations have assets they want to protect; those assets are subjected to threats; those threats can be opposed by countermeasures. We present the security concepts below and the relations between them are presented in Figure 2.4.

Assets Entities that someone places value upon. This could mean the contents of a file or a server, the authenticity of votes cast in an election, the availability of an electronic commerce process, etc.

Owners Place value on those assets. It is their responsibility to safeguard assets of interest.

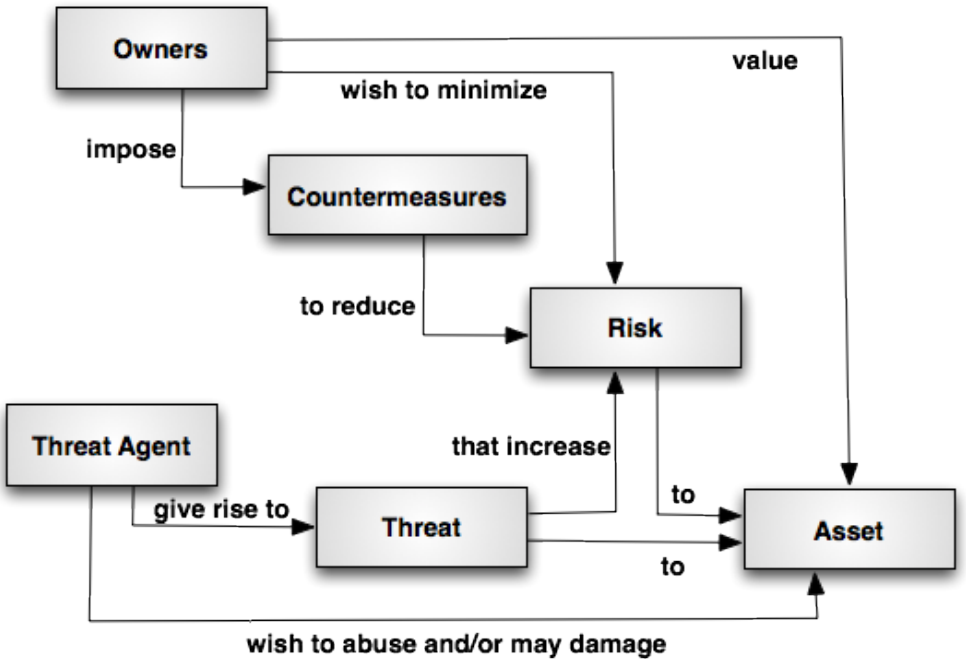

Figure 2.4: The General Model of the CC evaluation process

Threat Agents May also place value on the assets and seek to abuse assets in a manner contrary to the interests of the owner.

Threats Potential for impairment of assets such that the value of the assets to the owners would be reduced.

Risk Comes from the threats inflicted by Threat Agents on the Assets, based on the likelihood of a threat being realized and the impact on the assets when that threat is realized.

Countermeasures Subsequently, countermeasures are imposed to reduce the risks to assets. These countermeasures may consist of IT countermeasures (such as firewalls and smart cards) and non-IT countermeasures (such as guards and procedures). In our work, only IT countermeasures are considered.

Two important elements about countermeasures:

- **the countermeasures are sufficient:** if the countermeasures do what they claim to do, the threats to the assets are countered;
- **the countermeasures are correct:** the countermeasures do what they claim to do.

Evaluation of these countermeasures is important to see if they are being used in a sufficient and correct way.

2.6.1 Sufficiency of the Countermeasures

The sufficiency of the countermeasures is one of the main points in an evaluation. Countermeasures are exposed in the Security Target. In Section 3.4 this document is explained in more detail. Here we present a brief description of the parts related to the sufficiency of the countermeasures.

The Security Target as of now contains a description of the assets and the threats to those assets. Subsequently, it describes the countermeasures and demonstrates that these countermeasures are sufficient to counter these threats: if the countermeasures do what they claim to do, the threats are countered. Countermeasures are shown in the Security Target as the Security Objectives. The countermeasures are divided into two groups in the Security Target: the security objectives for the TOE, which describe the countermeasure(s) for which correctness will be determined in the evaluation and the security objectives for the Operational Environment, that describe the countermeasures for which correctness will not be determined in the evaluation.

The reason for this division is that in the CC only the IT countermeasures' correctness is suitable to be assessed. As such, the non-IT countermeasures (e.g. human security guards, procedures) are always in the Operational Environment. Assessing correctness of countermeasures costs time and money. This could possibly make it infeasible to assess the correctness of all IT countermeasures. The correctness of some IT countermeasures may already have been assessed in another evaluation. It is therefore not cost-effective to assess this correctness again.

The IT countermeasures whose correctness will be assessed during the evaluation are the ones that come from the Security Objectives for the TOE. A detailing of this Security Objectives for the TOE is necessary and is accomplished with the introduction of the Security Functional Requirements (SFRs). These SFRs are formulated in a standardized language (described in CC Part 2) to ensure exactness and facilitate comparability.

With this stated we can say that a correct TOE, with all the SFRs covered, in combination with a correct operational environment, with all the security objectives for the operational environment, will counter all the threats. In the next two sections correctness of the TOE and correctness of the operational environment are discussed separately.

2.6.2 Correctness of the TOE

An attacker can exploit vulnerabilities, damaging and/or abusing the assets. This becomes possible because a TOE can be incorrectly designed and implemented and can therefore contain errors that lead to the vulnerabilities used by attackers. Accidental errors made during the development, poor design, intentional addtion of malicious code, poor testing, among others, cause these vulnerabilities to arise.

Some of the tasks that can be performed to reduce risks are:

- testing the TOE;
- examining various design representations of the TOE;
- examining the physical security of the development environment of the TOE.

These activities are presented in the context of the CC in the form of Security Assurance Requirements (SARs) and are defined in the Security Target. These SARs are formulated in a standardized language (described in CC Part 3) to ensure exactness and facilitate comparability. If the SARs are met, there is assurance in the correctness of the TOE and the TOE is therefore less likely to contain vulnerabilities that can be exploited by attackers. The level of assurance that exists in the correctness of the TOE is determined by the EALs, which are explained in Section 2.4.

2.6.3 Correctness of the Operational Environment

As in the TOE, the operation environment can also be incorrectly designed and implemented, and may therefore contain errors that lead to vulnerabilities. By exploiting these vulnerabilities, attackers may still damage and/or abuse the assets. However, in the CC, no assurance is obtained regarding the correctness of the operational environment. Or, in other words, the operational environment is not evaluated. As far as the evaluation is concerned, the operational environment is assumed to be a 100% correct instantiation of the security objectives for the operational environment.

2.6.4 Common Criteria Evaluation

The CC has two types of evaluation: a Security Target/TOE evaluation and an evaluation of the Protection Profiles. In general the CC uses the term evaluation to refer to a Security Target/TOE evaluation.

The Security Target/TOE evaluation has the following steps defined in the CC:

a) a Security Target evaluation, where the sufficiency of the TOE and the operational environment are determined;

b) a TOE evaluation, where the correctness of the TOE is determined. As stated earlier, the TOE evaluation does not assess correctness of the operational environment.

In the case of the Security Target evaluation, the criteria applied are defined in CC Part 3, where the ASE assurance requirement is explained. The precise method to apply the ASE criteria is determined by the evaluation methodology that is used. The TOE evaluation is more complex. The main inputs to a TOE evaluation are: the evaluation evidence, which includes the TOE and Security Target, but will usually also include input from the development environment, such as design documents or developer test results. The TOE evaluation consists of applying the SARs (from the Security Target) to the evaluation evidence. The precise method to apply a specific SAR is determined by the evaluation methodology that is used.

The evaluation scheme under which the evaluation is carried out and the evaluation methodology followed states how the results must be presented. The results of the TOE evaluation process are either:

- all SARs have been met and therefore we have the specified level of assurance that the TOE meets the SFRs as stated in the Security Target;

- not all SARs have been met and therefore we do not have the specified level of assurance that the TOE meets the SFRs as stated in the Security Target.

As we have stated before, the TOE evaluation may be carried out after TOE development has finished, or in parallel with TOE development.

Below we present two other methodologies defined by the CC standard: how to counter *Threats* and the relation between Security Objectives and SFRs.

Countering Threats

Countering a threat does not necessarily mean removing that threat, it can also mean sufficiently diminishing that threat or sufficiently mitigating that threat. Examples of removing a threat are:

- removing the ability to execute the adverse action from the threat agent;
- moving, changing or protecting the asset in such a way that the adverse action is no longer applicable to it;
- removing the threat agent (e.g. removing machines that frequently crash it from a network).

Examples of diminishing a threat are:

- restricting the ability of a threat agent to perform adverse actions;
- restricting a threat agent opportunity to execute an adverse action;
- reducing the likelihood of an executed adverse action being successful;
- reducing the motivation to execute an adverse action of a threat agent by deterrence;
- requiring greater expertise or greater resources from the threat agent.

Examples of mitigating the effects of a threat are:

- making frequent back-ups of the asset;
- obtaining spare copies of an asset;
- insuring an asset;
- ensuring that successful adverse actions are always detected in a timely fashion, so that appropriate action can be taken.

These are examples of countermeasures that can be used to remove, reduce or mitigate a *Threat*.

Relation between SFRs and Security Objectives

In the Security Target we have a chapter that contains a security requirements rationale where two sections about SFRs are present: a tracing that shows which SFRs address and which Security Objectives for the TOE are relevant and a set of justifications that show that all Security Objectives for the TOE are effectively addressed by the SFRs. In terms of relation carnality we can say that each SFR traces back to at least one Security Objective and that each Security Objective for the TOE has at least one SFR tracing to it. Multiple SFRs may trace to the same security objective for the TOE, indicating that the combination of those security requirements meets that security objective for the TOE.

The security requirements rationale demonstrates that the tracing is effective: if all SFRs tracing to a particular security objective for the TOE are satisfied, that security objective for the TOE is achieved. This demonstration should analyze the effects of satisfying the relevant SFRs as far as achieving the security objective for the TOE is concerned and lead to the conclusion that this is indeed the case. In cases where SFRs very closely resemble security objectives for the TOE, the demonstration can be very simple.

2.7 Formal Methods in Common Criteria

This section introduces the other theme used in this dissertation. Even though formal methods are already recognized in CC certification and explicitly used in the higher EAL, other approaches to the use of formal methods have been used to help in software certification [1]. The contribution that formal methods can provide for the certification of software is achieved through software specification and verification [40]. Using formal methods to assure the correctness of cryptographic software makes even more sense due to the connection that this type of software has with critical components [2].

Formal methods are used today for various purposes in software systems. The myth that they can only be used in critical software, that they are extremely expensive and not worth the hassle is no longer valid [16]. In this work we show that these techniques can be used to validate documentation, increase assurance in documents, help construct documentation, etc [13]. We present some examples in Section 2.8.

We show how CC consider formal methods in the more rigorous certification levels to improve assurance. They advise the use of formal methods as a way to improve the software development process. This means that in CC, formal methods are used during normal software development process. In this dissertation work we will use Alloy, a particular formal methods technique, to model and validate documentation used as evidence in the CC certification process.

The highest level of CC assurance, EAL7, requires a formal specification of a product's security functions, its security model and formal proof of correspondence between the two.

Although the CC is just a framework that can be used in a variety of circumstances, its documentation does provide some guidance on the use of formal methods. This is in Annex 5 of CC Part 3 [24] Supplementary material on formal methods.

Although it does not recommend any particular formal methods it does give four examples:

- **The Z specification language** is highly expressive and supports many different

methods or styles of formal specification. The use of Z has been predominantly directed towards model-oriented specification, using schemas to formally specify operations.

- **ACL2** is an open-source formal system comprising a LISP-based specification language and a theorem prover.
- **Isabelle** is a popular generic theorem proving environment that allows mathematical formulae to be expressed in a formal language. It also provides tools for proving those formulae within a logical calculus.
- **The B method** is a formal system based on the propositional calculus, the first order of predicate calculus with inference rules and a set theory

If the developer uses a formal system which is already accepted by the evaluation authority the evaluator can rely on the level of formality and strength of the system and focus on the instantiation of the formal system to TOE specifications and correspondence proofs [20].

Below, we present a new process already used to obtain a CC certification with EAL7. We end this chapter with an overview of cases that use formal methods out of the traditional scope of CC to increase the success of the certifications.

CC evaluation of Embedded Device with EAL7

In this section we will follow the approach used in [19], where formal methods are used practically to certify a secure software system. This serves as an example of a CC evaluation using formal methods. This work is defined as innovative, to increase the use of formal methods in software certification. This approach was formulated to support a CC evaluation of an Embedded Device (ED) software piece's security. The CC requires a formal proof correspondence between a formal specification of ED's security functions and its required security properties as well as a demonstration that ED's implementation satisfied the formal specification.

The process presented in [19] is divided into five steps. First of all, 1) given source code annotated with Floyd-Hoare preconditions and postconditions and 2) a security property of interest, the problem is how to establish that the code satisfies the property. The Top Level Specification (TLS) purpose is to provide a precise yet understandable description of the allowed security-relevant external behavior of ED's separation kernel and to make the assumptions on which the TLS is based explicitly. The five steps of the process are listed as follows, with all explanation available to be consulted in [19]:

1. Formulate a TLS of the code as a state machine model;
2. Formally express the security property as a property of the state machine model. Confirm that the property is preserved under refinement;
3. Translate the TLS and the property into the language of a mechanical prover and formally prove that the TLS satisfies the property;
4. Given source code annotated with preconditions and postconditions, partition the code into three categories - Event, Other, and Trusted Code - based on criteria determined by the property of interest;

5. So as to demonstrate that the Event Code does not violate the property of interest, construct:

   (a) A mapping from the Event Code to the TLS events and from the code states to the states in the TLS;

   (b) A mapping from preconditions and postconditions of the TLS events to the preconditions and postconditions that annotate to the corresponding Event Code.

   Demonstrate separately that both Trusted Code and Other Code are benign. Based on these results, conclude that the code refines the TLS.

This work has introduced a novel and affordable approach to verifying security down to the source level. Tools such as checkers and theorem provers are already available for verifying that a formal specification satisfies a security property of interest [18]. This type of tools could help bring formal methods to the software certification process even more.

2.8 Other applications of Formal Methods in CC Certifications

We now present some works that use formal methods in software certification. These works are related with CC certifications, some of them directly related to the highest levels of EALs, others simply use these techniques aiming to improve the level of assurance in an evaluation component. In the previous section we have already presented an example of formal methods applied in the CC evaluation process.

The work by Hashii et al [17] presents the lessons learned using Alloy to formally specify an Multi Level Security (MLS) on the Darpa Polymorphous Computing Architecture (PCA), while it was under a high assurance certification process. PCA is a multi-processor architecture that allows a processor to morph during operations to provide the best type of processor for the job at hand. The goal of MLS-PCA is to create a high assurance security infrastructure for multi-processor distributed applications, with a certification like CC EAL7. In this work, we discuss the benefits of the formal specification being threefold. Firstly, the act of writing the specification forces one to examine the details of the design at a higher, more abstract level, than one would get from writing code. This allows the designers to focus on the issues that are important and gain distance from those that are not. In addition, both the writing and analysis of formal specifications allows one to encounter and deal with problems in the design phase. Eliminating problems after coding is extremely costly, mainly due to the need for regression testing. Finally, the ability to mathematically verify that specification holds to some criteria provides confidence that the design is correct.

Park et al [36] presents an Security Policy Modeling (SPM) requirement in the higher EALs of CC, using Z notation for CC version 3. EALs 5 to 7 are referred to as the high-assurance levels. In the highest levels such as EAL6 and EAL7, a formally specified and verified security policy model is needed in evaluation. SPM provides increased assurance that TOE satisfies security functional requirements. SPM establish a correspondence between security policy models and functional specifications. A security policy model may be used as guidance throughout the design, implementation, and review processes. It may be difficult to develop the formal SPM because the CC doesn't specify how to make the document

and how to use formal methods in modeling security policy. In addition, industries have rarely published the modeling result. In this work, a guideline for developing formal SPM for CC v3.1 with the example of a smart card operating system is provided.

Singh et al [41] present a formal specification of an Access Control Policy model also in Z notation for CC. The Insider Threat Study [3][7] provided the first comprehensive analysis of the insider threat problem. As part of the Insider Threat study, the lack of an effective access control mechanism is identified as one of the major issues that facilitated IT sabotage. Ninety three percent of the insiders in the IT sabotage cases exploited insufficient access controls. Other causes of Insider threat include System misconfigurations, disgruntled employees and overloaded system administrators etc. In this paper we focus on access control element of the insider IT sabotage problem. In this a formal security policy model for a security evaluation CC is developed, so as to provide a formal framework to allow the implementation of an Internal threat protection security solution against unauthorized access in network computing environments. The paper concludes with a case study along with model verification.

Téri et al [45] uses the B Method to formalize the java Card Runtime Security Policy for a CC evaluation. This work provides an overview of a set of techniques required to obtain high EALs CC for a Java Card. The motivation for a Java Card evaluation is to reach the same security level in the new open smart card as in traditional embedded platforms. They introduce Unified Modeling Language (UML) and the B method to illustrate the semi-formal and formal models required for a high level evaluation. With the B method they intend to formally model the security mechanisms of the Java Card: the bytecode verifier, the interpreter and the firewall.

In our case, we apply formal methods to increase the rigor of the certification process at the documentation analysis level in the CC certification process. We use the Alloy specification language to model the fundamental concepts of a security certification according to the CC, and then show how these models can be used to validate the consistency of Protection Profile and Security Target documentation.

Chapter 3

Common Criteria Documentation

We focus our work on the modeling and validation of documentation used in the certification process of cryptographic software. As we have presented previously, in Chapter 2, CC has several documents that are used as evidence in the certification process. In Section 2.6 we present the CC's General Model that includes the relations between security concepts. This will be the basis for the evidence documents. The documents that are more important in the certification process, the Protection Profile and the Security Target, are the focus of our work. These documents have a structure and properties that can be modeled as we show in the current chapter. Our global approach aims to validate the consistency of low level documentation, instances of Protection Profiles and Security Targets, against restrictions imposed in the very Protection Profile and in the concepts of CC.

In this chapter we present the CC documentation structure and explain in detail the most important documents in the certification process. The documents are interrelated, with the Protection Profile being followed by the Security Target as the documents that are in the CC documentation structure head. In the context of this work we also focus on documents for the development of evidence.

The concepts and terminology present in CC translate how security is discussed during a CC evaluation. We must understand these concepts and terminology for a correct use of the CC. The concepts are quite general and are not intended to restrict the class of IT security problems to which the CC is applicable. The restrictions that are present in the CC standard are defined in the General Model.

We initiated this chapter by introducing the documentation structure for CC. Next, we will present the restriction imposed by the General Model. After this, we will introduce the Protection Profile and the CC in detail, presenting the relevant characteristics to be analyzed in this type of documents. To conclude this chapter we also introduce another document, the Functional Specification. This is the document where the software product's external interfaces are defined, with the SFR stated in the Security Target. We want to be able to generate and validate information from the Security Target to be used in other documents.

3.1 CC Documentation Structure

To achieve a CC certification we need to produce a considerable amount of documents. We can see the documentation structure for CC in Figure 3.1. We can see the CC standards as readable guidelines to produce the documents in the top. The next document is the Protection Profile that gives origin to the Security Target. Some certifications may not use a Protection Profile. This way, we can go from the CC standards to the Security Target. This is explained in Section 3.3. The Sensitive Information present in the Security Target is a subset of this document that represents assets considered more sensitive in the context of the certification. This is explained in Section 3.4.

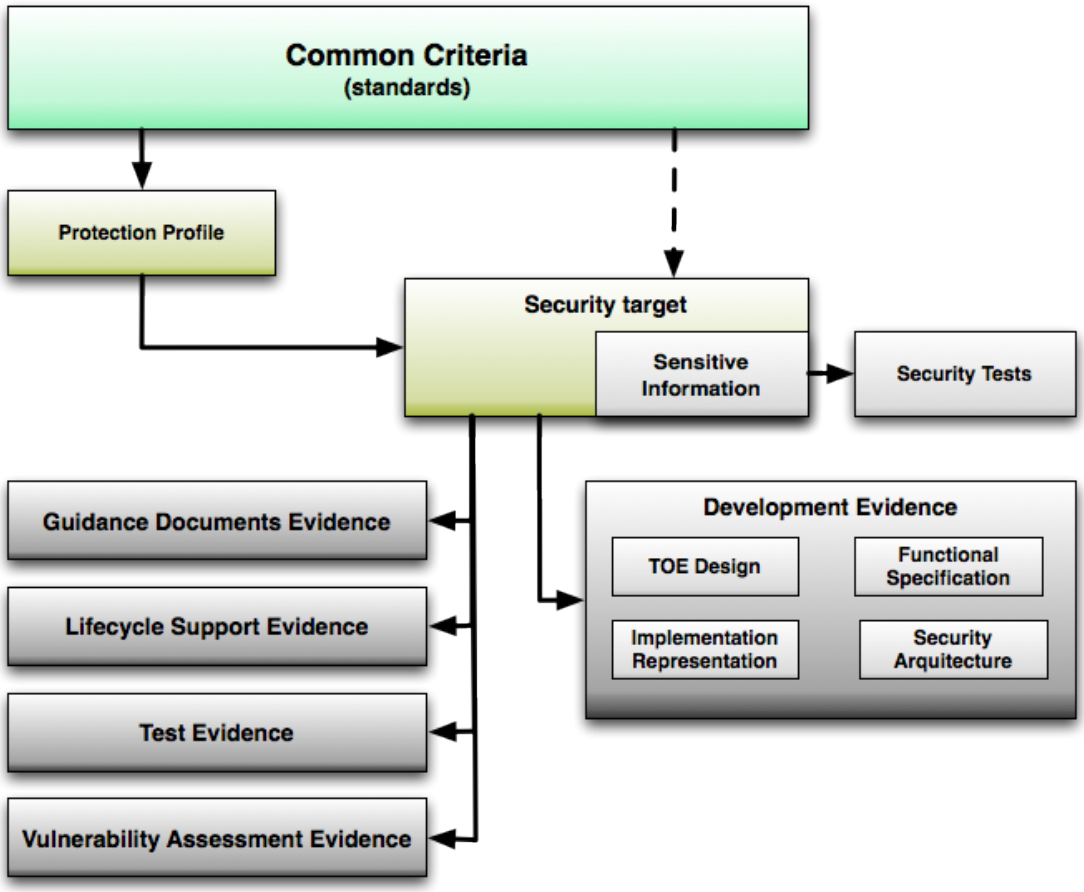

Figure 3.1: Common Criteria Documentation Structure

These documents are divided in evidence categories represented in Table 2.2. In this case we present the Development Evidence in more detail because we are mostly interested in this set of documents to perform our analysis. In the figure we only show the development documents for an EAL4 evaluation.

For the purpose of this work we focus on four levels of documentation. Figure 3.2 presents all four levels: General Model, Protection Profile, Security Target and the implementation documents level represented with the Functional Specification document. All the security concepts are represented in the General Model. This is the most abstract level of the certification process and is valid for all type of software products. The Protection Profile

is an instantiation of the General Model and is related to a specific type of product. The Security Target is the document where elements for a certification are defined and it is related to a unique product as a security specification. It is usually a subset of one or many Protection Profiles but it could also be a direct instantiation of the General Model. All the other documents are produced in relation to the Security Target. In our work we also considered the Functional Specification document, where the product implementation for the security functional requirements are stated.

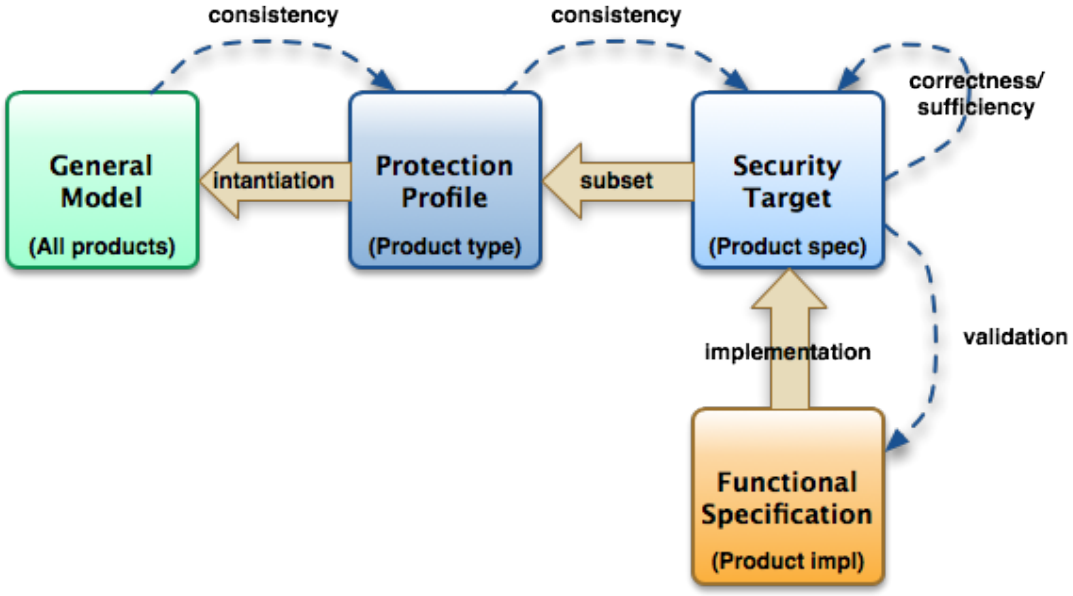

Figure 3.2: Documentation for analyzes

The purpose of this work, as we can see in Figure 3.2 with the blue arrows, is to go through this set of documents and perform an analysis of the most relevant security concepts. We analyze the Protection Profile for consistency against restrictions imposed in the General Model and with that information we can validate the document for all concepts included in the analysis process. Much in the same way, we can check the Security Target against restrictions imposed by the CC and in the Protection Profile validating the document for all concepts included in the analysis process. In the case of the Security Target, we can also analyze the security requirements in terms of sufficiency and correctness.

With the information present in the Security Target, we can generate contents present in other documents of CC with a higher abstraction level. This can be used to fill in parts of those documents. Alternatively, we can use use it to validate these parts comparing what is in the Security Target and what is in the other CC documents. We use this process in the Functional Specification document. This idea can be put into practice throughout the process of producing documents for the CC certification process, thus enabling the people that produce documents to quickly fill systematic parts of the process and save time.

Documents for evidence are very important in software certification with CC. In the following sections we present each of the documents defined here as crucial for this work. For each one of them we will introduce its purpose, which is important to understand its utility in the certification process; its structure and components, mentioning what is important for our work and which are the characteristics that make it important. We will also analyze so as to perform consistency checks and to understand what should be

generated, according to its relevance. With this we aim to contribute to the CC certification process and to help in the always tedious work that is producing the evidence for the certification.

3.2 Restrictions Imposed in the General Model

From the beginning, we have understood that some concepts are too generic and that detailing them would be necessary to fit the documents' contents. This brought about our work reformulating some things in the model presented in Figure 2.4. We are looking for a model that approximates us to the reality present in the CC documents.

In the context of CC certifications we have the Security Policies Definition (SPD) concept that includes the Threats, Organizational Security Policies (OSPs) and the Assumptions. We don't include all the SPD in our work because we are only interested on elements related to TOE. As such, we will only consider the security objectives for the TOE and the Threats. OSPs and Assumptions that are also defined in the SPD will not be considered, so we will only refer to the Threat as the whole SPD. Figure 3.3 shows the relation between Security Objectives and the SPD. However, even though OSP are also related to the Security Objectives for the TOE we do not consider them relevant for our analysis.

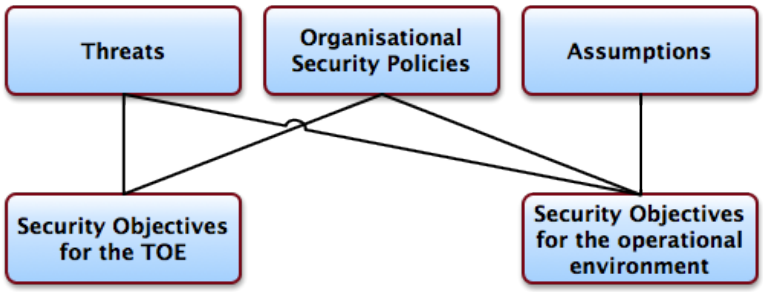

Figure 3.3: Security Objectives and Security Police Definition

In Figure 3.4 we present a new representation of the General Model, that we will follow as the basis for our work. In this model we have dropped the concept of the Owner as it is irrelevant for security concerns.

The Assets were also divided into two new concepts. We understand that an Asset has to have Sensitive Information, which holds the value, the information valuable to the owner. Besides the Sensitive Information, the Asset is also represented by a Security Information Goal. This represents the security goals that we need to consider for the Sensitive Information represented by the Asset.

The Countermeasures are also detailed in the model into two other concepts. These are key concepts in the CC certification process. In the context of the model, Security Objectives are high level statements which target the Threats to the Assets. The Risks are the link between Security Objectives and Threats, being the Security Objective the opponent to the Risk provoked by the Threat.

Related to the Security Objective is the SFR which is certainly the most important concept present in the CC standard. SFRs are a translation of the security objectives for the TOE into a standardized language. CC provide a long catalogue of SFRs that can be used in

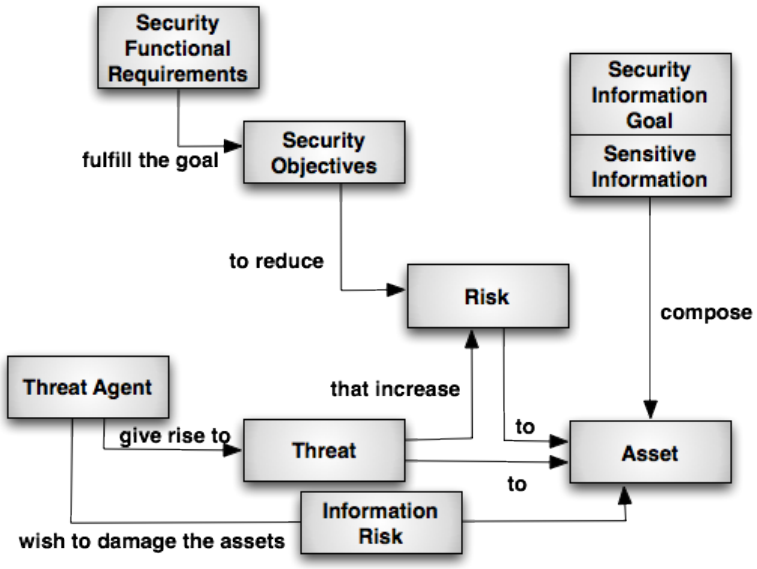

Figure 3.4: Detailed General Model

certifications to counter Threats. We can see the SFRs as a set of countermeasures that will fulfill the goal of the Security Objectives.

To analyze the restrictions imposed by the General Model we must look to Figure 3.4. We can quickly understand that for a Threat to exist, a Threat Agent must give origin to it. Therefore, we can claim that a Threat always has at least one Threat Agent. Following the same reasoning, we can claim that an Asset always has at least one Threat whenever there is Risk associated to it.

In terms of the Asset, it will have at least one Threat, a Threat Agent that gives origin to said Threat and one Risk. We will need to verify if the Threat Agents associated with the Threat in the Asset are the same Threat Agents of the Asset.

In the case of the Risk, they are the connection between the Threats and the Countermeasures. We understand the Risks as something that results from a succeeded Threat to an Asset. That damage must have origin in at least one Threat and can be prevented by a countermeasure. These countermeasures are presented in the context of the CC as Security Objectives that are supported by the SFR. Below we present examples of how to counter Threats and the relation between Security Objectives and SFRs.

3.3 Protection Profile

The Protection Profile was an important innovation of the CC. Its main goal is to state a security problem rigorously for a given collection of systems or products, known as the TOE, and to specify security requirements to address that problem without dictating how these requirements will be implemented [10].

A Protection Profile is a combination of threats, security objectives, assumptions, SFR, SAR and rationales. It also specifies generic security evaluation criteria to substantiate a given family of information system products vendor's claims.

In order to get a product evaluated and certified according to the CC, the product vendor has to define a Security Target which may comply with one or more Protection Profiles. Also notice that a Protection Profile may inherit requirements from one or more other Protection Profiles [22].

Another functionality of a Protection Profile is to specify the Evaluation Assurance Level (EAL), a number 1 through 7 [24], indicating the depth and rigour of the security evaluation, usually in the form of supporting documentation and testing, that a product meets the security requirements specified in the Protection Profile. A Security Target that follows a certain Protection Profile must define its assurance level according to the Protection Profile. A large number of Protection Profile have been developed and can be found in the CC portal[1].

The bulk of certified Protection Profiles are from the U.S. government ant the Smart Card Industry. The following are some of the aproved Protection Profiles:

- Anti-Virus
- Certificate Management
- Databases
- Intrusion Detection System/Intrusion Prevention System
- Firewall
- Operatin System
- Router
- USB Encryption
- Virtual Private Network
- Wireless LAN
- Enterprise Security Management
- Enterprise Firewall
- Security IC Platform (Smart Card)

Vendors can evaluate their products against the requirements present in one of these applicable Protection Profiles or they can make their own claims and have those claims evaluated in the CC evaluation process. However, some governmental agencies have established policies requiring some products to only evaluate against applicable Protection Profiles.

[1] `http://www.commoncriteriaportal.org/pps/`

3.3.1 Contents of a Protection Profile

CC states the contents for the Protection Profile. Each Protection Profile contains a set of mandatory parts. These parts must be integrated in the document following a structural outline like the one in Figure 3.5. Nevertheless, alternative structures are allowed. As an example, if the security requirements rationale is deemed complicated to explain and to introduce in the middle of the document, then it could be included in an appendix of the Protection Profile instead of the security requirements section.

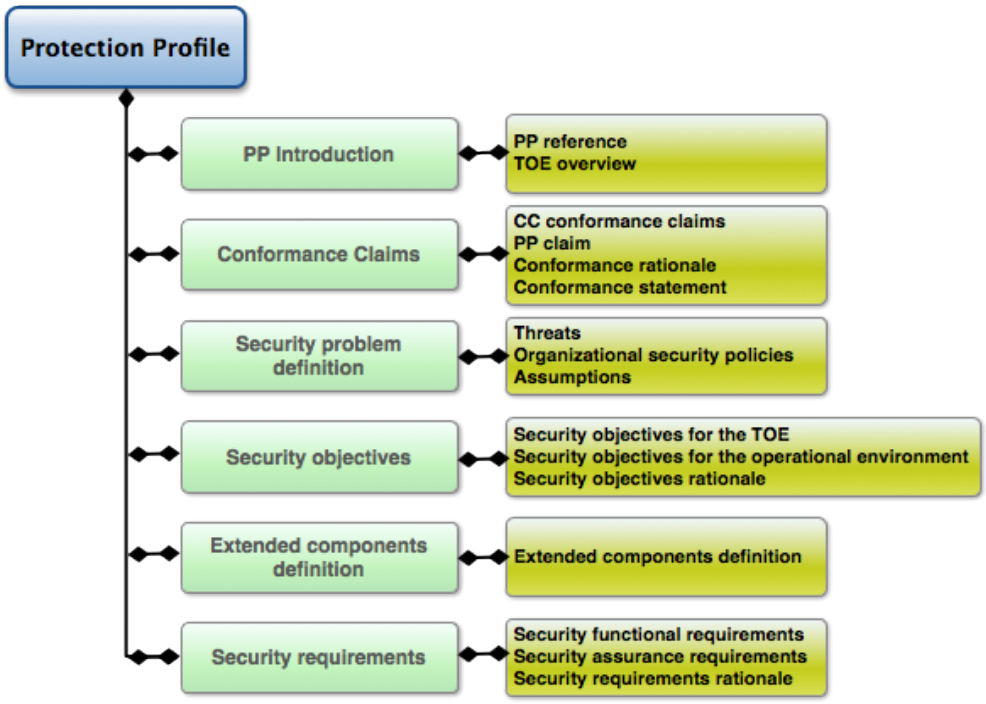

Figure 3.5: Contents of a Protection Profile

We present a brief explanation of all the contents below.

Introduction The Protection Profile introduction describes the TOE for each Protection Profile. Two levels of abstraction are intended: the Protection Profile reference, which provides identification material for the Protection Profile and a TOE overview where the TOE are briefly described.

The reference typically consists of a title, version, authors and publication date, facilitating the index and reference of the Protection Profile and its inclusion in lists. To achieve this, the reference must be unique so that it is possible to tell different Protection Profiles and different versions of the same Protection Profile apart.

The TOE overview means to allow potential consumers of a TOE who are looking through lists of evaluated products to find TOEs that may both meet their security needs and be supported by their hardware, software and firmware. This also aims to allow developers who may use the Protection Profile to design TOEs or to adapt existing products. We can see the TOE overview as a brief description of its usage and of its major security features, that identify the TOE type and any major non-TOE hardware/software/firmware made

available to the TOE.

Conformance Claims This section includes the CC conformance claims that specify which version of CC is being followed. Conformity with other Protection Profiles and with packages is written in the Protection Profile's conformance claims. The conformance rationale explains the rationale behind the claims. The conformance statement in the Protection Profile states how Security Targets and/or other Protection Profiles must conform to that Protection Profile.

The author of the Protection Profile also has to specify if conformance with the Protection Profile is "strict" or "demonstrable". Demonstrable conformance is orientated to the Protection Profile-author who requires evidence that the Security Target is a suitable solution to the generic security problem described in the Protection Profile. Strict conformance is oriented to the Protection Profile-author who requires evidence that the requirements in the Protection Profile are met, that the Security Target is an instantiation of the Protection Profile, even though the Security Target could be broader than it. In essence, the Security Target specifies that the TOE does at least the same as in the Protection Profile, while the operational environment does at most the same as in the Protection Profile.

Security Problem Definition In the Security Problem Definition the security usage assumptions, the Threats and the Organizational Security Policies (OSP) for the TOE are defined. As we have said previously we only considered the Threats for our analysis.

Security Objectives This section of the Protection Profile shows the solution to the security problem is divided between security objectives for the TOE and security objectives for its operation environment.

Extended Components Definition This is the section where new components (i.e. those not included in CC Part 2 or CC Part 3) may be defined. This allows new requirements do the catalogue of functional and assurance requirements of the CC to be added.

Security Requirements In this section the TOE's security objective's translation into a standardized language is provided. This standardized language is in the form of SFRs. The SARs is also included in this section defining which EAL must be followed with the Protection Profile.

3.3.2 Choice of Protection Profile

The step of choosing the Protection Profile in the evaluation process is certainly one of the most important. The Protection Profile is an improvement of CC and aims to serve as guidance to the evaluation of software product types. The vendors choose a Protection Profile that fits the security requirements of their product and will build its evaluation against that Protection Profile. National Information Assurance Partnership (NIAP) has stated in 2003 the following policies for acquiring products by the Department of Defense (DOD) [21]:

- If a Protection Profile for a given product type exists and some product have been validated against that Protection Profile, then only those products must be procured;

- If a Protection Profile for a given product type exists but no products have been validated against that Protection Profile, then the vendors are required to certify their products and validate them against that Protection Profile in order the product to be sold to the DOD;

- If no Protection Profile exists for the given product type, then the vendor must certify their products submitting them for evaluation and validation against the vendor-provided Security Target.

With these policies, any vendor that wants to see their products procured by the DOD has to evaluate it with CC and if possible against a recognized Protection Profile. In Figure 3.6 the decision process described above is illustrated.

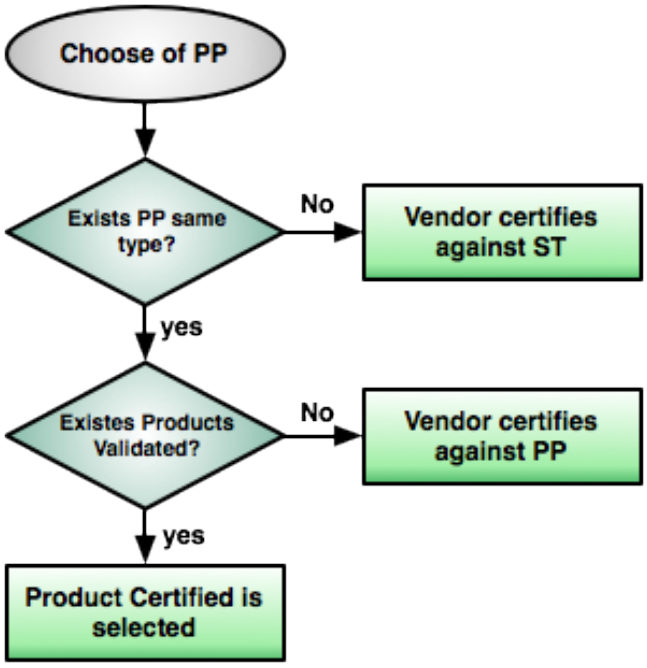

Figure 3.6: Protection Profile Decision Tree

Despite these policies, the CC don't require that software products are evaluated against Protection Profile and admit that sometimes this could be negative for the process of evaluation. The Protection Profile imposes certain requirements that may not be appropriate for the software product, even if the product is of the same type, since it may have particular features.

Developers can also choose a Protection Profile that is of the same type as the product and, in the Security Target, claim to follow the Protection Profile with some restrictions or with new requirements retrieved directly from the CC standard. They may even create or establish new requirements themselves. This liberalization of requirement choice is also an advantage of CC [21].

3.3.3 Consistency Concerns for the Protection Profile

The concerns over Protection Profile consistency are based on the restrictions imposed in the General Model. We need to check if the information present in the various chapters of the Protection Profile follows the rules imposed by the CC standard in terms of relations between the CC concepts. From the Protection Profile structure presented before we will look at Threats, Security Objectives, Threat Agents, Assets and SFRs to analyze if these elements are all in conformity with the restrictions present in the General Model.

We want to see if all the Threats are being countered by a Security Objective, that, in turn is related to at least one SFR. We want to check if all Assets are taken into consideration and are properly related to their Threats, Threat Agents and one Risk. We can also check if we don't have duplicate cases, for example, two Assets with the same Risk and same Threats.

In Protection Profile consistency checking we are mainly concerned with the information being properly placed in the document and if that information is sufficient and correct for all the Assets addressed by the Protection Profile.

3.4 Security Target

In this section we present the Security Target document. We start by stating the importance of said document, after which we follow with the contents that a Security Target has and a brief description of these contents. A perspective for using the Security Target is given in Section 3.4.2. We also introduce the low assurance Security Target.

As we have previously stated, the Security Target is a security specification for a software product. This is the central document, typically provided by the developer of the product, and it specifies security evaluation criteria to substantiate the vendor's claims for the product's security properties [10].

A Security Target defines information assurance security requirements for the given information system product, which is called the TOE. A Security Target is a complete and rigorous description of a security problem in terms of TOE description, threats, assumptions, security objectives, SFRs, SARs, and rationales [23]. The SARs are typically given as a number 1 through 7 called EAL, which, as stated above, indicate the depth and rigor of the security evaluation, usually in the form of supporting documentation and testing, to ensure that the product meets the SFRs [24]. Unlike Protection Profiles, Security Targets are implementation dependent [22].

A Security Target is the basis for the agreement between the developers, evaluators and, where appropriate, users on the TOE security properties and on the scope of the evaluation. A Security Target is usually derived from a given Protection Profile by instantiation; in general, each Security Target corresponds to a particular Protection Profile definition. A Security Target may then claim conformance to a Protection Profile by providing the implementation details concerning the security requirements defined by that Protection Profile [6]. Also, a Security Target may augment the requirements derived from the Protection Profile. Indeed there might be cases where there is no Protection Profile that matches the security properties of a specific product. In this case, the product developer could create his own Security Target without claiming conformance to any Protection Pro-

file. A Security Target may also claim conformance to more than one Protection Profile [22].

The Security Target document frames the CC evaluation effort by answering the question: What is being evaluated?. A large number of Security Target for different types of products have been developed and can be found in the CC portal[2].

3.4.1 Contents of a Security Target

The CC states the contents for the Security Target. Each Security Target contains a set of mandatory parts. These parts must be integrated into the document following a structural outline like the one in Figure 3.7. However, alternative structures are allowed. As an example, if the security requirements rationale is complicated to explain and to introduce in the middle of the document, it could be included in one of the Security Target's appendixes instead of being included in the security requirements section.

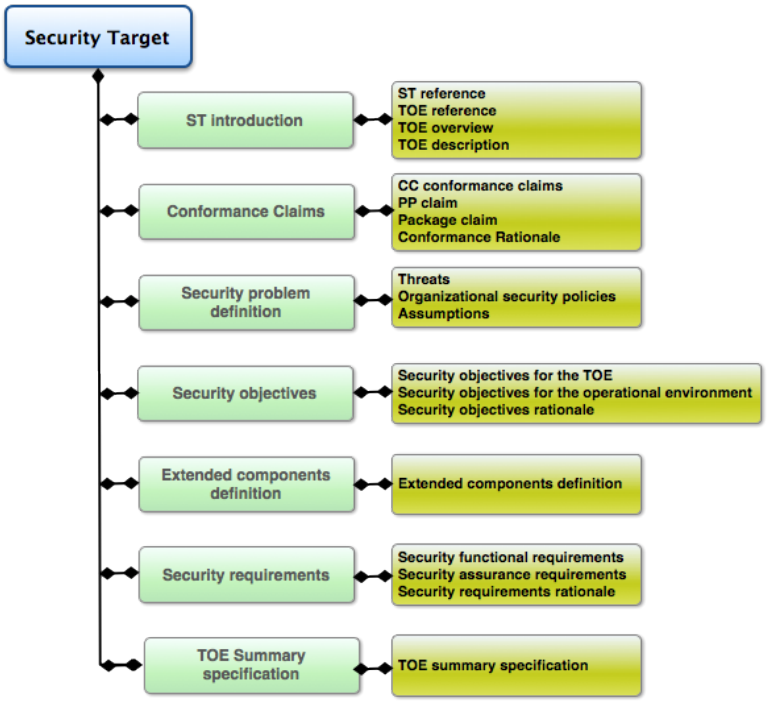

Figure 3.7: Contents of a Security Target

We present a brief explanation of all the contents of a Security Target below.

Introduction Security Target introductions describe the TOE for each Security Target and are intended to be in three levels of abstraction: the Protection Profile reference, which provides identification material for the Protection Profile, a TOE overview where the TOE are briefly described and the TOE description which describes the TOE in mode detail.

Any given Security Target contains a clear Security Target reference that identifies that particular Security Target. A typical Security Target reference consists of a title, version,

[2] `http://www.commoncriteriaportal.org/products/`

authors and publication date. The Security Target also contains a TOE reference that identifies the TOE that claims conformance to it. A typical TOE reference consists of the developer's name, TOE name and TOE version number. The Security Target reference must be unique. As a single TOE may be evaluated multiple times, for instance, by different consumers, and therefore have multiple Security Targets, this reference is not necessarily unique. The Security Target reference and the TOE reference facilitate indexing and referencing the Security Target and TOE and their inclusion in summaries of lists of evaluated TOEs/Products.

TOE overview intends to allow potential consumers of a TOE who are looking through lists of evaluated products to find TOEs that may both meet their security needs and be supported by their hardware, software and firmware. We can see the TOE overview as a brief description of its usage and its major security features, which identify the TOE type and any major non-TOE hardware/software/firmware available to it. The TOE overview identifies the general type of TOE, such as: firewall, VPN-firewall, smart card, crypto-modem, intranet, web server, database, web server and database, LAN, LAN with web server and database, etc.

TOE description explains the TOE in detail, usually occupying several pages of a Security Target. The TOE description should provide evaluators and potential consumers with a general understanding of the security capabilities of the TOE in more detail than that which was provided in its overview. The TOE description may also be used to describe the wider application context into which it will fit.

This part of a Security Target is usually important as it is the explanation of the physical and logical scopes. They describe the TOE in such a way that there remains no doubt over whether a certain part or feature is in the TOE or whether this part or feature is outside of it. The physical scope of the TOE consists of: a list of all hardware, firmware, software and guidance parts that constitute the TOE. This list should be described at a level of detail that is sufficient to give the reader a general understanding of those parts. The logical scope of the TOE consists of the logical security features offered by the TOE at a level of detail that is sufficient to give the reader a general understanding of those features. This description is expected to be more detailed than the major security features described in the TOE overview.

Conformance Claims This section includes the CC conformance claims that specify which version of CC is being followed and whether the Security Target contains extended security requirements or not. The conformity with Protection Profile (if any) and with packages (if any) is written in the Protection Profile conformance claims. The Security Target's description of conformance to Protection Profiles means that the Security Target lists the packages said conformance is being claimed to. The Security Target's description of conformance to packages means that the Security Target lists the packages said conformance is being claimed to. The conformance rationale explains the rationale behind the claims.

Security Problem Definition This section of the Security Target defines the security problem that is to be addressed. The process of deriving the security problem definition is not included in the scope of CC. However, the success of an evaluation strongly depends on the Security Target, and the usefulness of the Security Target strongly depends on the quality of the security problem definition. It is therefore often worthwhile to spend

significant resources and use well-defined processes and analysis to derive a good security problem definition.

The security Problem Definition is composed by threats, OSPs and assumptions. However, according to CC Part 3 it is not mandatory to have statements in all sections. A Security Target with threats does not need to have OSPs and vice versa. Also, any Security Target may omit assumptions. It is also normal to discuss the relevant threats, OSPs and assumptions separately for distinct domains of the TOE operational environment and for the cases where the TOE is physically distributed.

Security Objectives After presenting the problem, it is necessary to present a solution. In the Security Target, the section that presents the solution for the problem defined by security problem definition is the Security Objectives section. The security objectives are a concise and abstract statement of the intended solution to the problem defined by the security problem definition. The security objectives are divided into three parts:

- provide a high-level, natural language solution to the problem
- divide this solution into two parts: security objectives for the TOE and security objectives for the operation environment.
- demonstrate that the solution covers all of the presented problems.

With this section and based on the security objectives and the security objectives rationale we can reach a conclusion: if all the security objectives are achieved then the security problem as defined in security problem definition is solved. This means that all threats are countered, all OSPs are enforced and all assumptions are upheld. This is one of the objectives that we want to guarantee with this work.

Extended Components Definition In this section all the new components that were created for the evaluation presented in the Security Target are included. In most cases, the requirements are taken from CC Part 2 or CC Part 3, or are based on those components. In some cases, there may be requirements in a Security Target that are not based on components of the CC. In this case, new components (extended components) must be defined, and this definition should be done in the Extended Components Definition. This section is only for the new components and not for new requirements derived from other components. These must be defined in the security requirements section.

Security Requirements The security requirements consist of two groups of requirements, as demonstrated before: SFRs, a translation of the security objectives for the TOE into a standardized language; and the security assurance requirements (SARs), a description of how assurance is granted that the TOE meets the SFRs. The Security Target also contains a security requirements rationale in this section that explains why this particular set of SARs was deemed appropriate. There are no specific requirements for this explanation. The goal for this explanation is to allow the readers of the Security Target to understand the reasons why this particular set was chosen.

TOE summary specification This is the last mandatory piece of content for the Security Target and aims to provide a description of how the TOE satisfies all the SFRs. The TOE's summary specification should allow potential consumers to understand the general technical mechanisms that the TOE uses for this purpose. This should have a level of detail sufficient for a plain understanding of the SFR's covering of the TOE security objectives.

3.4.2 Using a Security Target

The roles that a Security Target has can be divided in two: before and especially during the evaluation, the Security Target specifies "what is to be evaluated" and after the evaluation the Security Target specifies "what was evaluated". During the evaluation the Security Target serves as a point of agreement between the developer and the evaluator on the exact security properties of the TOE and the exact scope of the evaluation. The point of correctness and completeness of all the evidence are always the Security Target and what the Security Target specifies. After the evaluation, the Security Target will serve for the seller or the developer to present to the potential consumer of the TOE the security claims achieved with the certification of the product. The Security Target describes the exact security properties of the TOE in an abstract manner, and as such, the potential consumer can rely on this description to prove that the TOE has been evaluated to meet the Security Target. This means that the Security Target must be easy to use and easy to understand.

It is also important to understand for what the Security Target must not be used. The Security Target must not be used essentially for two roles. Others exist, but we refer two that are present in the CC: a detailed specification and a complete specification. A Security Target is designed to be a security specification on a relatively high level of abstraction. A Security Target should not, in general, contain detailed protocol specifications, detailed descriptions of algorithms and/or mechanisms, long descriptions of detailed operations etc. The Security Target was also designed to be a security specification and not a general specification. Unless security-relevant, properties such as interoperability, physical size and weight, required voltage etc. should not be part of a Security Target. This means that, in general, a Security Target may be a part of a complete specification, but not a complete specification itself.

3.4.3 Sensitive Information

The Sensitive Information appears in the certification process as an optional document that works as a subset of the Security Target to clarify some of its elements. In some cases, it is complicated to explain how threats are treated in the software development. This document aims to further explain how the concepts are related in the *Assets* assumed as sensitive to the system.

This document is presented as a listing of several *Assets* where for each of them we have: description, threat, rationale, related SFR, protection required, control, access rights and location. This can be traced to fit the General Model presented in the Figure 2.4. Having this document as a subset of the Security Target we can perform analysis to validate the Sensitive Information as well.

3.4.4 Consistency Concerns for the Security Target

The Security Target is similar to the Protection Profile in terms of structure, so the consistency checking is also similar. As we have seen before the Security Target is a subset of the Protection Profile. This means that we can apply all of the consistency assumptions from the Protection Profile to the Security Target.

In this case we do this analysis based in the General Model but, if applicable, with the restrictions imposed by the Protection Profile. Suppose that it is stated in the Protection Profile that all *Threats* must be countered with at least two *Security Objectives*. This must be introduced in the Security Target analysis.

In the Security Target we are also concerned with the information being properly placed in the document and if that information is sufficient and correct for all the *Assets*. We also want to generate information to aim the verification of the Security Target in terms of correction and sufficiency for the TOE, as presented in Section 2.6.1 and Section 2.6.2, explained below.

3.4.5 SFR Dependencies

Some components of CC have dependencies between them. SFRs are one of those components, where when an SFR isn't self-sufficient for some functionality, it is dependent on another functionality, or interacts with another component. We can consult the list of dependencies in CC Part 2 component definitions. In order to ensure completeness of the TOE security requirements, dependencies should be satisfied when requirements based on components with dependencies are incorporated into Protection Profiles and Security Target. We have prepared our analysis for the Security Target case. However, this can also be used for Protection Profile.

Each SFR has its own lists of dependencies. However, these lists can be empty. There are three list of dependencies: required, indirect required or optional. These lists of dependencies are normative. This means that they should be followed through the Protection Profile/Security Target. In other words: if component A has a dependency on component B, this means that whenever a Protection Profile/Security Target contains a security requirement based on component A, the Protection Profile/Security Target shall also contain one of :

a) a security requirement based on component B;

b) a security requirement based on a component that is hierarchically higher than B;

c) a justification on why the Protection Profile/Security Target does not contain a security requirement based on component B.

In cases a) and b), when a security requirement is included because of a dependency, it may be necessary to complete operations (assignment, iteration, refinement, selection) on that security requirement in a particular manner to make sure that it actually satisfies the dependency. This operations are explained in the CC Part 2 and are out of the scope of this work. In case c), the justification that a security requirement is not included should address either:

- why the dependency is not necessary or useful;
- that the dependency has been addressed by the operational environment of the TOE, in which case the justification should describe how the security objectives for the operational environment address this dependency;
- that the dependency has been addressed by the other SFRs in some other manner.

We are interested in analysing this process in the Security Target, which could be complicated to follow. It may also prove difficult to ensure that all the dependencies have been taken into consideration.

	FAU_GEN.1	FAU_SAA.1	FAU_SAR.1	FAU_STG.1	FIA_UID.1	FMT_MTD.1	FMT_SMF.1	FMT_SMR.1	FPT_STM.1
FAU_ARP.1	-	X							-
FAU_GEN.1									X
FAU_GEN.2	X				X				-
FAU_SAA.1	X								-
FAU_SAA.2					X				
FAU_SAA.3									
FAU_SAA.4									
FAU_SAR.1	X								-
FAU_SAR.2	-		X						-
FAU_SAR.3	-		X						-
FAU_SEL.1	X				-	X	-	-	-
FAU_STG.1	X								-
FAU_STG.2	X								-
FAU_STG.3	-			X					-
FAU_STG.4	-			X					-

Figure 3.8: Dependency table for Class FAU: Security audit

In Figure 3.8 we show a table retrieved from the CC standard. In that table we show their direct, indirect and optional dependencies of a class of SFR. Each of the components that is a dependency of some functional component is allocated a column. Each functional component is allocated a row. The value in the table cell indicates whether the column label component is *directly required* (indicated by a cross "X"), *indirectly required* (indicated by a dash "-"), or *optionally required* (indicated by an "o") by the row label component. An example of a component with optional dependencies is FDP_ETC.1 Export of user data without security attributes, which *requires* either FDP_ACC.1 Subset access control or FDP_IFC.1 Subset information flow control to be present. So if FDP_ACC.1 Subset access control is present, FDP_IFC.1 Subset information flow control is not necessary and vice versa. If no character is presented, the component is not dependent upon another component.

Table 3.1: Functional Specification Component Requirement

EAL	Component requirement
EAL1	Basic functional specification
EAL2	Security-enforcing functional specification
EAL3	Functional specification with complete summary
EAL4	Complete functional specification
EAL5	Complete semi-formal functional specification with additional error information
EAL6	Complete semi-formal functional specification with additional formal specification

3.4.6 Generation of Information from the Security Target

As we have stated many times through the course of this document, the Security Target is the most important document in the CC certification process. It is based on this document that all the other documents produced for evidence will retrieve their information. There is information that can be generated from the Security Target and used to fill other documents and validate them, such as the Functional Specification with the SFR Tracing presented bellow. We can also generate information that will be used in the very CC. In this case we are talking about the Correction and Sufficiency analysis that is made in the rationale present in the Security Target.

SFR Tracing

SFR are the most important components in the software certification with CC. These components are defined in the CC standard in the form of a catalogue of functional requirements. They are picked for the Protection Profile or the Security Target, where the authors of these documents can create other components and insert them into the documents to be used during CC evaluations. This results in an extensive catalogue of SFRs that is hard do structure and relate to during an evaluation.

We want to validate the path that the SFRs do from the CC standards to the Functional Specification. This means that we will trace the SFRs from the CC to the Protection Profile, from the Protection Profile to the Security Target and from the Security Target to the Functional Specification. We want to generate this information to help construct these documents and validate them.

3.5 Functional Specification

Functional Specification is a document included in the development evidence class for the CC process of certification. The objective of this document is to describe the functional specifications of the software product. It contains the information necessary to fulfill the requirements of CC component ADV_FSP. This document has several iterations, from 1 to 6 as can be seen in Table 3.1. According to the EAL chosen for the certification, with this, the contents of the document vary. In the lower EALs this document is essentially a specification of the interfaces for the software product, while in the higher EAL this document includes formal presentations of the functional specifications for the TSF.

This document provides a high level description of the external interfaces of TOE, which are necessary to satisfy the security functional requirements of the Security Target. Therefore, all externally available interfaces are documented in the functional specifications. Additionally, interfaces that provide security functions (directly or indirectly) are called TOE Security Function Interfaces (TSFI). All effects, exceptions and error messages of TSFI are described in this document, which also includes:

- Outputs of the TOE (e.g. certificates and Certificate Revocation Lists (CRLs) issued, event log entries, etc.);
- Reasons for an interface to be irrelevant in terms of security (e.g. no reconfiguration possible).

However, this document is not meant to describe how the TSF processes the requests received by each interface, nor does it describe the communication flow when the TSF uses its operational environment services. This information is addressed by the TOE design (ADV_TDS) document.

There are different types of interfaces that need to be clarified in this document:

SFR-Enforcing - an interface that can be traced directly to an SFR;

SFR-Supporting - an interface that supports the TOE security policies;

SFR-non-interfering - an interface where no SFR-enforcing functionality has any dependence.

This document is drawn up according to the contents of the Security Target. This contents defines all the security requirements for the certification and must be followed through the CC documents. In the case of the Functional Specification document, we want to generate information, such as the SFR Tracing, to be able to validate this document or help build it.

3.5.1 Contents of a Functional Specification for EAL4

In the case of Security Target there isn't any structure defined in the CC standards, so we will present the structure of the document produced for the CESeCore certification. In Figure 3.9 we show the structure of an Functional Specification document. The main component is the chapter interfaces where all the TSFI for the software product are specified.

Below we present a brief explanation of all the contents.

Introduction The introduction presents an introduction of the document in terms of its purpose and objective in the context of the certification. It also presents the related documents, that usually include the Security Target used in the same certification.

TOE Functional Architecture In this chapter an overview of the TOE is introduced, in terms of its architecture. A set of diagrams representing different levels of abstraction of the TOE architecture is usually presented.

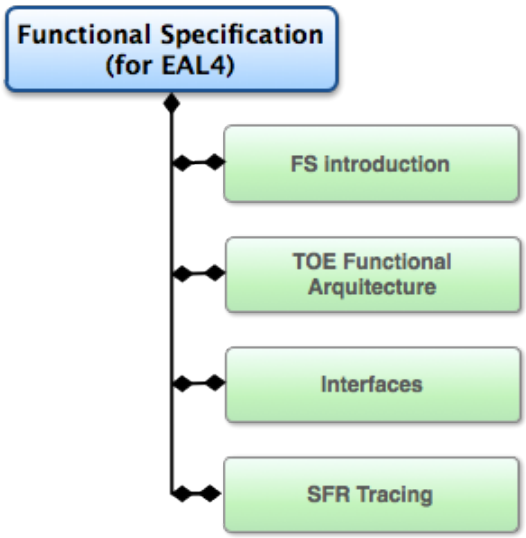

Figure 3.9: Contents of a Functional Specification

Interfaces As the assurance levels increase we need to provide greater description and proof that the interface fits the designated category. Each TSFI is described in terms of its:

- **Purpose** - general goal of the interface;
- **Method of Use** - how the interface is to be used;
- **Actions** - what the interfaces does;
- **Paramaters** - inputs and outputs;
- **Parameter descriptions** - meaningful descriptions;
- **Log Events** - events that need to be registered in the log;
- **Related SFRs** - SFR related with the interface.

SFR Tracing This chapter demonstrates that the SFRs trace to TSFIs is defined in this document. All the SFRs are present in the Security Target. They have their presence traced in the Functional Specification and also in what TSFIs they fit.

3.5.2 Generate Information for Functional Specification

We are mainly interested in the generation of SFR Tracing presented in Section 3.4.6. The main objective of this document is to map the SFRs from the Security Target to the several external interfaces for the TOE. We want to generate the information from the Security Target and use it to validate this document.

We can also make some consistency checking in terms of seeing if all the SFR are being mapped to at least one interface, and if all interfaces have SFRs associated.

Chapter 4

CC Documentation Modeling and Validation in Alloy

Up to this point we have presented the context and motivation for this work. We have presented an overview of CC, an overview of formal methods in the context of software certification and we have presented the structure of CC documentation, with the documents necessary for this work in detail. From this point on we will present the work done to improve the elaboration of CC documents.

The elements of a CC certification presented in the previous chapter were modeled using a formal specification language and were checked for consistency against restrictions imposed by the CC standards or other documents used during the certification. With this we hope to give a higher level of assurance to the documents produced in certifications and improve their production process. We use Alloy Modeling Language (Alloy) in the context of this work.

In the first place, we introduce an overview of the Alloy language. Subsequently, we present a model that represents the security concepts retrieved from the CC documentation with a few changes that we consider necessary to improve their application. This is based in the General Model presented in Section 2.6. The model elaborated from the General Model is the basis of all the other models.

For the two most important documents in CC certification, the Protection Profile and the Security Target, which are similar in their structure, we have developed a model that includes the elements relevant for analysis. After the inclusion of the necessary information in the model we are able to perform consistency verifications in these models and to validate them. As stated before, we also want to be able to generate information that could be included in higher level CC documents as the Functional Specification.

In Figure 4.1 we show how the models are structured in documents. Henceforth, we shall address each of them as individual models. In the top we have the most abstract level, that represents the General Model, which we have called *Abstract Model*. Below this, we have the *Full Model*. This is the model where we include the information present in the Protection Profile and in the Security Target to perform consistency analyses. In the lowest level we have two models: the *SFR Tracing* where we can generate the SFRs for each interface and the *SFR Dependencies* where we can validate them.

To finalize, we perform consistency checking along these models in order to validate the CC

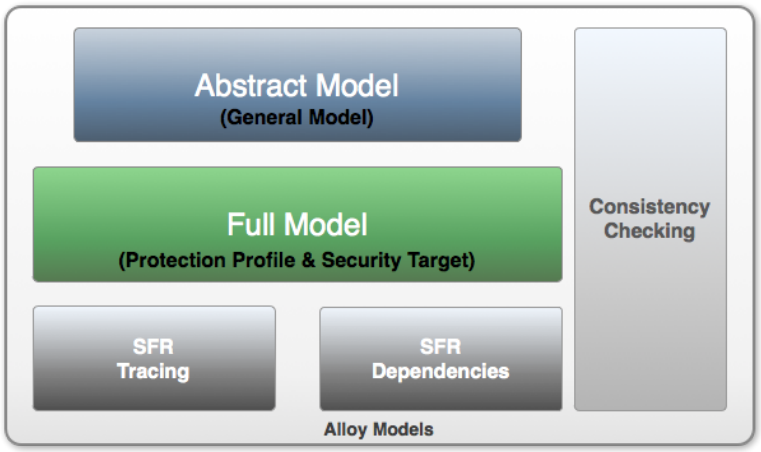

Figure 4.1: Structure of the Alloy Models

documentation identified for analysis. This consistency is based in restrictions, norms and characteristics present in the CC standards and in other documents that are part of the certification process, disposed hierarchically as presented in Figure 3.1. In the following sections we present these Alloy models.

4.1 Modeling with Alloy

We have already introduced Alloy in Section 1.1 when we discussed formal techniques. This was addressed more specifically when formal specification was discussed. We will use Alloy to model the concepts used in the CC certification process, and with this verify the consistency so as to validate the documents presented as evidence for evaluation. Alloy's approach is divided into three key elements that we describe below [27][32]:

logic Alloy uses first-order relational logic, where reasoning is based on statements written in terms of atoms and relations between atoms. Any property or behavior is expressed as a constraint using set, logical and relational operators.

language The Alloy language supports typing, sub-typing and compile-time type-checking, giving more expressive power on top of the logic. It also allows for generically-described modules to be re-used in different contexts. Also very useful is syntax support for three styles of writing Alloy model specifications that users can mix and vary at will: predicate-calculus style, relational-calculus style and navigational-expression style. Navigational-expression style, where expressions are sets and are addressed by "navigating" from quantified variables, is the most common and natural.

analysis The Alloy Analyzer is a model-finder that tries either to find an instance of the model or a counter-example to any property assertions specified in the model. An instance is literally an instantiation of the atoms and relations in the model specification. It performs a bounded-analysis by requiring a user-specified bound on the number of atoms instantiated in the model. With this bound, it translates the model into a boolean satisfiability (SAT) problem. Then, it hands the SAT problem off to a commercial SAT solver such as Berkmin [15]. The resulting solution is then

interpreted under the scope of the model and presented to the user in a graphical user-interface as show in Figure 4.2.

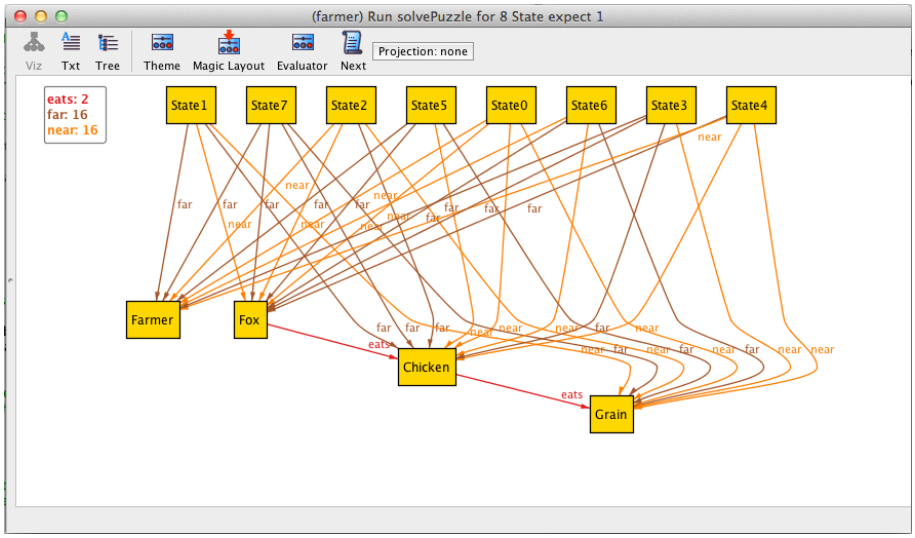

Figure 4.2: Alloy analyzer instance for a farm problem

In the following sections we present the three approach in more detail. This introduction to Alloy serves as a tutorial to understand the logic, language and analyses on related to working with CC documentation.

4.1.1 Relational Logic

In Alloy we build the models using atoms, tuples and relations. An atom is something that is indivisible, a tuple is an ordered sequence of atoms and a relation is a set of tuples. Relations in Alloy can only contain tuples of atoms and not tuples of other relations. This is because relation logic used by Alloy is of a first-order variety [32].

We can think of relations as if they were tables. In a table we have multiple rows, where each row corresponds to a tuple and each column in a tuple corresponds to an atom. Subsequently, we can have an unary relation that is a table with one column, a binary relation that is a table with two columns and so on. In a table, we also have the number of rows in a relation that represents its size and the number of columns that represents its arity. A scalar quantity is represented by a singleton set containing exactly one tuple with one atom [28].

To express constraints and manipulate relations, we use operators. Alloy operators fall into three classes: the standard set operators, the standard logical operators and the relational operators. The standard set and logical operators listed in Table 4.1.

The relational operators require a little more treatment. Let p be a relation containing k tuples of the form $\{p1, \cdots, pm\}$ and q be a relation containing l tuples of the form $\{q1, \cdots, qn\}$ [27].

- $p -> q$ - the relational product of p and q gives a new relation $r = \{p1, \cdots, pm, q1, \cdots, qn\}$ for every combination of a tuple from p and a tuple for q (kl pairs).

Table 4.1: Set and logical operators used in Alloy

Symbol	Operator
+	union
-	difference
&	intersection
in	subset
=	equality
!	negation
&&	conjunction
||	disjunction

- $p.q$ - the relational join of p and q is the relation containing tuples of the form $\{p1, \cdots, pm-1, q2, \cdots, qn\}$ for each pair of tuples where the first is a tuple from p and the second is a tuple from q and the last atom of the first tuple matches the first atom of the second tuple.
- $\hat{}p$ - the transitive closure of p is the smallest relation that contains p and is transitive. Transitive means that if the relation contains (a, b) and (b, c), it also contains (a, c). Note that p must a binary relation and that the resulting relation is also a binary relation.
- $*p$ - the reflexive-transitive closure of p is the smallest relation that contains p and is both transitive and reflexive, meaning that all tuples of the form (a, a) are present. Again, p must be a binary relation and the resulting relation is also a binary relation.
- $\sim p$ - the transpose of a binary relation r forms a new relation that has the order of atoms in its tuples reversed. Therefore, if p contains (a, b) then r will contain (b, a).
- $p <: q$ - the domain restriction of p and q is the relation r that contains those tuples from q that start with an atom in p. Here, p must be a set. Therefore, a tuple $\{q1, \cdots, qn\}$ only appears in r if $q1$ is found in p.
- $p :> q$ - the range restriction of p and q is the relation r that contain those tuples from p that end with an atom in q. This time, q must a set. Similarly, a tuple $\{p1, \cdots, pm\}$ only appears in r if pm is found in q.
- $p++q$ - the relational override of p by q results in relation r that contains all tuples in p except for those tuples whose first atom appears as the first atom in some tuple in q. Those tuples are replaced by their corresponding ones in q.

Now, let us explore a brief tutorial on writing model specifications in Alloy.

4.1.2 Alloy Models

To explain the basic modeling techniques in Alloy, we will consider a classic model based on the song "I'm My Own Grandpa". In this problem we intended to prove that no one can be their own Grandpa. In this case we only show that no one can be their own Father. For the rest of the problem we advise consultation of the book "Software Abstractions" [27].

Defining object types

An Alloy model consist of a number of signatures (sig), or sets of atoms that describe basic types in the model. Within these signatures, we can define relations that associate these basic types to one another. Firstly, we define a abstract type called Person. This is abstract because no one can simply just be a Person, seeing as we humans are divided into Men and Women. In that person we are defining two relations, father and mother, and saying that a Person can have zero or one Man as father, and zero or one Woman as their mother. Next we declare the Man and the Woman that have a relation to declare if they have a husband or a wife. All this is expressed with the following:

Listing 4.1: Declaration Part

```
1 abstract sig Person {
2    father: lone Man,
3    mother: lone Woman
4 }
5
6 sig Man extends Person {
7    wife: lone Woman
8 }
9
10 sig Woman extends Person {
11    husband: lone Man
12 }
```

The lone multiplicity states that the relation could have zero or one connections. As we have said before, we only have at most one father or mother. In the case of the wife and husband relation for our purpose we only consider at most one wife or one husband.

Specifying constraints

After defining object types, we want to be able to specify properties about their behavior by specifying constraints. Although the possible relations have been sufficiently defined, we may need to add certain constraints to prevent Alloy from producing trivial instances or counterexamples during its model-finding. This is made through facts. A fact records a constraint that is assumed to always hold. In this example its a fact that no one can be their own ancestor and that if someone is the wife of other, the other is their husband. This can be written with the following:

Listing 4.2: Example of a fact in Alloy

```
1 fact {
2    no p: Person | p in p.^(mother+father)
3    wife = ~husband
4 }
```

In the first line we are saying that no person (p) can be joined with the same p through the mother or father relation. In the second line, with the relational operator transposition, we state that if a Person is the wife, the other is her husband.

Running analyses

With the model in place, we can proceed to perform some analyses. We can simulate the model by defining predicates on it which can either be true or false depending on the instance. These are different from factual constraints, which are always required to be true in any instance. Simulation of a predicate is the process of instructing Alloy to find an instance for which the predicate is true. To obtain an instance of the model without imposing any further constraints, we instruct Alloy to simulate the empty predicate show(), which is effectively a non-constraint. In this case, Alloy just has to find an instance that satisfies the object definitions and their factual constraints. Since Alloy only performs bounded-analyses, we specify a maximum of 10 atoms in each object type (i.e. each sig).

Listing 4.3: Running the model to find a valid instance

```
1 pred show() {}
2 run show for 10
```

Figure 4.2 shows Alloys graphical output for a model related with a farm. Alternatively, we can also define assertions on properties of the model that we believe to be true. Checking is the process of instructing Alloy to find an instance satisfying the model but not the assertion. For example, we want to ensure that nobody is their own father, which is valid, and no counterexamples are found. We write this as:

Listing 4.4: Exemple of a assertion in Alloy

```
1 assert NoSelfFather {
2    no m: Man | m = m.father
3 }
4 // This should not find any counterexamples
5 check NoSelfFather
```

Alloy will then proceed to find an instance that satisfies the model but acts as a counterexample to the assertion, an instance where someone is their own father. All of Alloy analyses have to be run within a size bound on the number of atoms in each object type. Alloy will check all models within this binding until a satisfying instance or counterexample is found. Given that the running time of the analysis increases exponentially with size, we can only check small models (8-12 atoms per object type) in reasonable time. However, the philosophy behind Alloy is that if there are any inconsistencies with the model, they are extremely likely to appear even in small models.

This section serves as a introduction to Alloy Modeling Language. We will now present the models that we have developed in the context of this work.

4.2 Abstract Model

Our work began with the analysis of how the concepts presented in Section 2.6 are interrelated. Along this section we will present our modulation in Alloy of the General Model.

All the components are taken in consideration except the owners since we don't use these components for our verification. As we said, some alterations are introduced in the General Model's Alloy modulation. We introduce all the components, one by one, explaining their

relations and introducing new concepts that are created to detail the General Model model. In the Abstract Model we only have declarations, in here where we declare all the concepts as abstract so that only when we extend these declarations will they have meaning. At this point, Alloy generates abstract instances of the concepts.

We start to show the model by demonstrating the relation between the *Threat* and the *Threat Agent*. Naturally, *Threats* are triggered by a malicious user, defined in Alloy as:

Listing 4.5: Threat in Alloy

```
1 abstract sig Threat {
2    t_agent: some Threat_Agent
3 }
```

We state that a *Threat* will always have at least one *Threat Agent*. We have nothing to relate to the *Threat Agent*. We nevertheless have to connect it with the *Asset*.

Listing 4.6: Threat Agent in Alloy

```
1 abstract sig Threat_Agent {}
```

The most important concept in the general model is the *Asset*. This is what's important to protect and what is meaningful for *Threat Agents*. We divide the concept into two other concepts that we consider compose an *Asset*, so each *Asset* is composed by a pair. A *Security Information Goal*, which is the part of the *Asset* we mean to assure and the *Sensitive Information*, which is the information in the *Asset*. For example, some *Sensitive Information* could represent access to a restricted component of the system and if some *Threat Agent* has access to that area, he could damage that component, so we have a problem of integrity. The resulting pair will be (Access Control ComponentX, Integrity).

Listing 4.7: Asset in Alloy

```
1 abstract sig Asset {
2    sinfg: one Security_Information_Goal,
3    si: one Sensitive_Information,
4    threat: some Threat,
5    ta_motivation: some Inf_Risk,
6    risk: one Risk
7 }
```

In addition to these two relations, the *Assets* also have a relation with the *Threat*, since at least one is contained in it. We also have a relation with the *Threat Agent*, that passes by a new atom introduced in the model that we called *Information Risk*. A *Threat Agent* could abuse or/and damage an *Asset* with an attack that we called Inf_Risk. A single *Asset* can be attacked by many *Threat Agents*.

Listing 4.8: Information Risk in Alloy

```
1 abstract sig Inf_Risk {
2    t_agent: some Threat_Agent
3 }
```

Another relation with the *Asset* is Risk, the pair *Asset*, *Threat*, always has at least one *Risk* associated to it.

Listing 4.9: Risk in Alloy

```
1 abstract sig Risk {
2    threat: some Threat,
3    sec_objective: one Security_Objective
4 }
```

Risks are a central concept in the model as they relate *Threats* with *Countermeasures*. *Threats* are the origin of *Risks* and *Security Objectives* are *Countermeasures* to *Threats*. *Security Objectives* are defined in the CC as a statement of an intent to counter identified threats and/or satisfy identified organization security policies and/or assumptions. In the context of this work we only consider *Threats*, so we replace *Security Objectives* with *Countermeasures* in the General Model. This *Security Objectives* have a set of SFR that will counter the *Threats*.

Listing 4.10: Security Objective in Alloy

```
1 abstract sig Security_Objective {
2    srf: some SRF
3 }
```

The SFR are connected with the *Security Objectives*, and these are connected with *Risk*, making it possible to browse all of them. This SFR will have an important role in our work. They are the most important concept in the CC certification. It is necessary to ensure that they are correctly applied, and properly considered in the several documents of the CC evaluation.

Listing 4.11: Security Functional Requirement in Alloy

```
1 abstract sig SRF {}
```

This is the model used to represent the security concepts of CC. In Figure 3.4 we present a version of the General Model with the new concepts introduced here for the Alloy model. With this we can specify the contents of Protection Profiles and Security Targets making it possible to analyze that information.

The Abstract Model is the basis for all the other models. In the next section we present the Full Model where the information present in the Protection Profile/Security Target is modeled in Alloy.

4.3 Full Model

In the section above we present the concepts used in a CC certification, but that part of the model only allows us to understand how these concepts relate between them. We want to analyze information present in the documents of CC certification, so we need to introduce that information in our models to be able to perform analyses on these documents. The information that we want to include in our models is that which is present in Protection Profile and in the Security Target. Also, we can only include the information present in the Sensitive Information document, a subset of the Security Target that we present in Section 3.4.3.

To include the document's information, we have created a part for our Alloy models called Full Model. In that Full Model, we extend the declarations present in the Abstract Model with the information present in the documents. It is based on the Full Model that we will perform consistency analyses on the documents. As such, we need to include the information properly for each concept presented in the previous section. As an example, here we create one atom for each Threat Agent, and this will be the Threat Agent set available in the model:

Listing 4.12: Threat Agents for the TOE

```
1 one sig Administrator, Officer, Operator, Auditor, Other
2    extends Threat_Agent {}
```

There are restrictions associated to the security concepts that are important to validate. In the Abstract Model presented above, some are already taken into account and specified. These restrictions could be validated when we introduce the information present in documentation for validation in the Full Model. For example, when we say *some* (example above), there will be at least one Threat_Agent for each Inf_Risk, and since Inf_Risk is abstract, when we extend Inf_Risk we will always need to associate a Threat_Agent to it.

To represent the documentation we understand that we could use the Sensitive Information, a subset of the ST that contains all the information that must receive more attention in the product under certification, as opposed to using all of the information present in the ST and in the PP. With the Sensitive Information we have written a Full Model where methodologies are applied so as to validate the documentation. We will present some examples in Alloy of the Full Model. The examples presented here are retrieved from the case study CESeCore, presented in Section 5.2.

Firstly, we present the Security Information Goals present in the CESeCore certification. In this case, instead of *"abstract"* before the declaration, we have *"one"*. This means that only one instance of that declaration could be created, allowing us to ensure that when we perform some analyses on the model, we only will have one Confidentiality, one Integrity and one Availability atom. All of the declarations present in the Full Model will have "one".

Listing 4.13: Security Information Goals present in CESeCore

```
1 one sig Confidentiality, Integrity, Availability
2    extends Security_Information_Goal {}
```

The other concept that we use to complete the Asset pair the Sensitive Information. In it we present an example of a Sensitive Information item. In this case we have Access Rules Configuration, that will have the Security Information Goal Integrity associated to it. This will give origin to the Asset (Access Rules Configuration, Integrity) that we will present below.

Listing 4.14: Example of Sensitive Information

```
1 one sig Access_Rules_Configuration
2    extends Sensitive_Information {}
```

We have said before that a Threat is always associated with at least one Threat Agent. Here we show the declaration of a Threat with a set of Threat Agents. Note that these Threat Agents were already declared in the beginning of this section.

Listing 4.15: Threat in Alloy

```
-- Threat
one sig Tampering extends Threat {}{
    t_agent = Administrator + Officer + Operator + Auditor + Other
}
```

As we have stated previously in the presentation of the General Model in Alloy, for each Asset that has a Threat associated to it, there will also be a Threat Agent with a reason to attack that Asset. Here we present an example of an Information Risk for a set of Threat Agents that could damage an Asset.

Listing 4.16: Information Risk example for a set of Threat Agents

```
one sig Want_to_grant_his_role_more_access_rights_than_it_should_have
    extends Inf_Risk {}{
    t_agent = Officer + Operator + Auditor + Other
}
```

Moving on to Countermeasures, we have an example of a Security Objective, that is related to an SFR. This Security Objective is used to counter a Threat that is covered by that SFR.

Listing 4.17: Example of a Security Objective

```

one sig O_Configuration_management extends Security_Objective {}{
    srf = FMT_MOF_1
}
```

The Risks are something that we don't have information of on the documents and which will always be something abstract. They exist to connect the Threats to the Security Objectives when they are placed in the Asset declaration.

Listing 4.18: Example of a Risk

```
one sig Risk1 extends Risk {}{
    threat = Tampering &&
    sec_objective = O_Configuration_management
}
```

The SFR are the last concept that we need to introduce. The example below was already used in the declaration of the Security Objective above.

Listing 4.19: Example of a Security Functional Requirement in Alloy

```

one sig FMT_MOF_1 extends SRF{}{}
```

The last concept represented is the one that connects all the dots in the model. The Asset serves as point of collision for all the concepts presented before. The concept declarations above are necessary for the declaration of the Asset. We can see that the Security Information Goal (sinfg) is the Integrity, the Sensitive Information (si) is the Access Rules Configuration, the Threat (threat) is Tampering, the Information Risk (ta_motivation) is the same presented before and finally the Risk (risk) is the Risk that we presented above. This is repeated for all the Assets present in a CC certification process.

Listing 4.20: Example of a Asset

```
1 // SI#1 Access Rules Configuration
2 one sig A_Access_Rules_Configuration extends Asset {}{
3    sinfg = Integrity &&
4    si = Access_Rules_Configuration &&
5    threat = Tampering &&
6    ta_motivation =
         Want_to_grant_his_role_more_access_rights_than_it_should_have &&
7    risk = Risk1
8 }
```

In this section we presented the Full Model, with some examples of the case study for CESeCore. However, there is some other information that we will need to introduce in this model, and some other abstract declaration in the Abstract Model that we choose to only introduce in the following section. That information is relative to the SFR Dependencies and the SFR Tracing that we present in the next section.

4.4 Checking Consistency and Validation of Documents

In the two previous sections we have presented the Abstract Model and the Full Model respectively. These two sections represent the declaration part of our Alloy model. In the first case we have introduced the concepts present in the CC and in the second case we have introduced the information from the documents that we want to analyze. In those sections, although not directly, we have also introduced some restrictions to our possible model so as to do some consistency analysis.

In this section we present the part related with the documentation analysis.

4.4.1 Checking Consistency

We have started our consistency verification from the General Model. In Section 3.2 we have presented the consistency concerns for the General Model. In here we have stated that all the restrictions present in that model must be used to ensure that all the information present in the CC documents have their concepts correctly related. In the Abstract Model we have introduced those restrictions. For example, when we have said that an Asset must have a Threat and a Risk, we have guaranteed that in the Full Model we always have at least one Threat and one Risk associated to every Asset. With this, as the Risks always have a Security Objective associated, we guarantee that for all the Assets threatened, we always have a Security Objective countering every Threat.

However, we don't look for the model to see what we want to verify. We look at the CC standard to see what are the restriction present for the General Model. In the standard, as we state in Section 3.2, we can see, for example, that all Threats must be countered by a Security Objective. Here we will present these verifications in Alloy, for our example:

Listing 4.21: Security Objective countering the Threat in the Asset

```
1 assert SOminThreat {
2    all  a: Asset |
3    one a.risk.sec_objective &&
4    some a.risk.threat
```

```
5 }
6 check SOminThreat
```

In Alloy we can perform analyses on the model by using assertions. The AA uses its SAT solver to quickly verify all the components and generate, if there are any, counterexamples of the sentence checked. In this case the AA will look to all Assets and see if they have a Security Objective associated. If not, the AA will produce a counterexample with that instance.

Another example is to check if the Threat Agents that are responsible for a Threat to an Asset are contained in the set of Threat Agents that could inflict said Threat. This is a kind of information that could be incorrectly placed in the documentation and it is difficult to analyze.

Listing 4.22: Security Objective countering the Threat in the Asset

```
1 assert TAinT {
2    all a: Asset |
3    a.ta_motivation.t_agent in a.threat.t_agent
4 }
5
6 check TAinT
```

We use these assertions so as to afterwards, including the information in the Full Model, analyze the consistency of the model. Another example, to verify if there is only one Sensitive Information, Security Information Goal pair or to verify if Assets always have a Security Objective associated.

Listing 4.23: Threat Agent in Alloy

```
1 assert PairSIGSI {
2    no disj a,a': Asset |
3    (a.sinfg + a.si) == (a'.sinfg + a'.si)
4 }
```

In this example, we can validate if all the Assets have the Security Objectives necessary to guarantee that the Asset is protected against its Threats, or we can validate if the Threat Agents associated to an Asset are present in the Threat that is being inflicted on that Asset. We can also validate the uniqueness of the pairs that represent the Assets. With this analysis we can validate the Sensitive Information retrieved from the Security Target. The number of verifications is large and we can't enumerate them all. However, we can say that they cover the consistency concerns presented in the previous chapter for each document.

Another aspect that we can validate is the Sufficiency and Correctness of Countermeasures, explained in section 2.3.1. In order to achieve this, we can list combinations of atoms present in the Full Model and analyze if indeed, for example, the combinations of a Security Objectives set are sufficient to protect a specific Threat. In another example, we can see the sufficiency of the SFR to cover a certain Security Objective. For example, in the Protection Profile used in the CESeCore certification we can find the following example of a rationale:

"***O.Configuration Management*** *is provided by* ***FMT_MOF.1*** *(Management of security functions behavior) (iterations 1 and 2) which covers the requirement that only*

authorized users can change the configuration of the system. ***FMT_MOF_CIMC.2*** *(Certificate profile management)...'*

Here we can see that to cover the Security Objective presented we have two SFRs. With the Alloy models we can generate information that allows us to see beforehand what are the SFR that are associated with a Security Objective and validate them in terms of sufficiency. To do this we use the evaluator present in the AA. Figure 4.3 presents the example above.

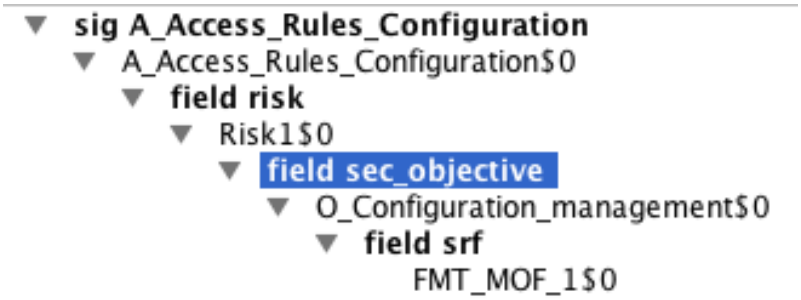

Figure 4.3: Evaluator of the Alloy Analyzer

4.4.2 SFR Dependencies

We are also using these models to check the dependencies that exist between the SFRs that are normative by the CC and see if these dependencies are correctly handled in the Protection Profile/Security Target. We have presented the SFR Dependencies in Section 3.4.5. In it, we have stated that the SFR have three dependency lists. As such, to begin our inclusion of this new concept in our models we need to change the declaration of an SFR in the Abstract Model.

Listing 4.24: New declaration of a SFRs

```
1 abstract sig SRF {
2    d_required: set SRF,
3    d_indrequired: set SRF,
4    d_optonally: set SRF
5 }
```

Now we have three lists of SFR in each SFR. In the Full Model we will fill this list with information retrieved from the CC standards or another other document, like a Protection Profile.

Listing 4.25: List of Dependencies in the SFRs

```
1
2 one sig FMT_MOF_1 extends SFR{}{
3    d_required = FMT_SMF_1 + FMT_SMR_1
4    d_indrequired = FIA_UID_1
5 }
```

After including this information in the Full Model for every SFR, we still have to include the information from the document which dependencies we are analyzing. Normally, an observation is included to all the components (SFR) that are included in the CC certification. We declare each component's dependencies and if we are including them or not.

To introduce that information we go back to the Abstract Model and declare two new concepts: WichIs and Component.

Listing 4.26: Components (SFRs)

```
abstract sig WichIs {}

abstract sig Component {
   srf: one SRF,
   dependencies: set SRF -> WichIs
}
```

With the WichIs we can introduce if the dependency is included or not. Dependencies and status at that point are included in the components for each Component (SFR). As such, in the Full Model we now have to declare the two WichIs statuses, and all the Components.

Listing 4.27: Declaration of the dependencies

```

one sig Included,NotIncluded extends WichIs {}

one sig Cfau_gen_1 extends Component {}{
   srf = FAU_GEN_1 &&
   dependencies = FPT_STM_1 -> Included
}
```

After including all the information in the Full Model we can perform some analyses and verify some properties. For example, we can verify if all the required dependencies are included in the Security Target.

Listing 4.28: Checking the SFR Dependencies

```
assert checkDependencies {
   all c : Component |
   c.srf.d_required in c.dependencies
}

check checkDependencies
```

This analysis allows the validation of a difficult component of the documents. This is usually presented in expenses tables, which makes it difficult to ensure that all the components are correctly placed.

4.4.3 SFR Tracing

Another type of verification that we can use this models for, with some alterations, is the generation and validation of the SFR in the Functional Specification of the CC certification. We can generate all the SFR for each Interface specified for the software product by looking at the Asset and mapping the SFR that are connected to it and to the interface that involves each Asset. We can validate the SFR present in the Functional Specification by doing the opposite, by looking to the SFR present in the document and validating it against the SFR related with each Asset. We introduce this in Section 3.4.6.

To perform these analyses we need to add some concepts and information to our models. Firstly, we need to add the concept of an interface to the Abstract Model:

Listing 4.29: Interface in the Abstract Model

```
1
2 abstract sig Interface{
3    asset: some Asset,
4    relatedsrf: set SRF
5 }
```

We declare an Interface as something that connects at least one Asset to a set of SFRs. After including the concept of an Interface in the Abstract Model we need to fill the Full Model with all the Interfaces for a certification. Once again we use examples from the CESeCore certification. In this case we include an Interface for the Roles.

Listing 4.30: Interface Roles in the Full Model

```
1
2 one sig Roles_ManageRoles extends Interface{}{
3    asset = A_Roles_Configuration +
4            A_Access_Rules_Configuration +
5            A_Roles_Configuration
6    relatedsrf = FMT_MOF_1 + FDP_ACF_1
7 }
```

Now that all the information is declared in the models we can run some analyses. We want to see if we have some Assets and SFRs for all the Interfaces. This has to do with consistency checking, to verify if the information is correctly placed in the documentation. The second assertion has to do with SFR Tracing. We want to see if the SFRs present in our previous Alloy model, when related to an Asset, are included in the Interface that contains that Asset. This for all Interfaces.

Listing 4.31: Assertions for the SFR Tracing

```
1 assert AssetinInterface {
2    all i: Interface |
3    some i.asset && some i.relatedsrf
4 }
5
6 check AssetinInterface
7
8 assert checkSRFtracing {
9    all i: Interface |
10   i.asset.risk.sec_objective.srf in i.relatedsrf
11 }
12
13 check checkSRFtracing
```

This will allow us to validate the documents Functional Specifications where the interfaces for a certification are defined. CC demand that a correct SFR Tracing is placed in that document.

Other analyses could be performed based in this models, this is left as future work. The main problem this work has presented until this point is the introduction of information in the models. We understand that a tool that could perform parsing in the CC documents is absolutely necessary to make the use of these models viable. In the next chapter we introduce a proof of concept for a tool to aid the CC documentation development based on

these models that will allow for information inclusion and for easier manipulation of these Alloy models as well.

Chapter 5

Case Studies

Along this document we have presented our contextualization of the CC standards and the use of formal methods techniques in the CC certification process. We have also presented our motivation to perform this work on the CC documentation and finally we have presented our work to improve document development and validation in the CC documentation process. In this chapter we will introduce the case studies used in this work.

To do this we have based our work with CC in two certifications. However, other examples of certifications were also occasionally analyzed. During this work we have been involved in the certification process of a cryptographic core in a software package for digital certification, CESeCore Project (CESeCore). This is the first case study. The other case is also related to the certification process of a cryptographic core in a software package for digital certification, project RSA Keon CA System (Keon CA). These two certifications are explained in the following sections.

In this chapter we show how we have applied the models presented in Chapter 4. In that chapter we already have presented some portions of the documentation present in the CESeCore certification. The two case studies are similar. In order to present the application on CC documentation we chose CESeCore because it is the project that we are more acquainted with. Also, we have easy access to the necessary documentation.

During the work in the study cases, we have realized the problem that most of CC documents do not have a machine-readable form that would allow for their easy interpretation in cases such as the one as the proposed in this work. To fill this gap we have developed templates in Extensible Markup Language (XML) and parsers to manage this information. In this chapter we will present the XML version of the CC documents. To finalize our work we specify a toolbox to aid in the development of CC documentation.

5.1 RSA Keon CA System

In this section, we examine the RSA Keon CA System (Keon CA)[1] CC certification in detail. Keon CA was certified in 2002 but, late in 2006, it obtained a revalidation of its EAL4+ certification to match the new version of CC. The certification analyzed here is from Version 2.3 of the CC with EAL4+ [30]. We will start by presenting the Protection

[1] http://www.rsa.com/

Profile and the Security Target used to obtained the certification. Afterwards, we will show the evaluation process.

The Protection Profile that Keon CA follows is the Certificate Issuing and Management Components Family of Protection Profiles (CIMC), which defines requirements for components that issue, revoke, and manage public key certificates, such as X.509 public key certificates [35]. Actually, in the CIMC, there are four Protection Profiles with different security levels (1-4). The Keon CA follows security level 3 [30]. The four Protection Profiles (i.e. security levels) are hierarchical.

The Security Target of Keon CA, RSA Keon CA System version 6.7 Security Target (Keon ST), the Certificate Manager conforms to the Security Level 3 protection profile. The TOE meets all the Security Level 3 Functional and Assurance Requirements. Additionally, RSA has elected to pursue a more rigorous Assurance evaluation and the TOE additionally conforms to all the Assurance Requirements for an EAL4 product. The resulting assurance level is therefore CIMC Security Level 3 with an overall EAL4, augmented with the Security Level 3 Assurance Requirement: ALC_FLR.2. The Assurance and other CC required documentation, specifically this Security Target, conform to the CC for Information Technology Security Evaluation, Version 2.3 part 2 extended, and to part 3 [23][24].

One of the strongest aspects of the CC certification process is that the mapping between the requirements is done in a clear and concise way from the Protection Profile to the Security Target. Bellow, we analyze both, the CIMC protection profile and the Keon ST security target, to highlight the strict correspondence between the different sections of these two documents. Before that, let us look closely at Keon CA's features.

5.1.1 Description of Keon CA

The Keon CA is a signing authority solution for large enterprises and public CAs. The Keon CA system is responsible for creating and issuing both authority and end-entity public-key certificates, creating and issuing CRLs, and responding to status requests. In addition to the basic CA functionalities, the Keon CA system provides:

- Audit recording and backup capabilities;
- Use of a FIPS140-1 or FIPS140-2 Level 3 cryptographic module to protect all private keys and additionally for key generation.

The RSA Certificate Manager is designed to meet the CIMC Security Level 3 requirements, which are appropriate where the risks and consequences of data disclosure and loss of data integrity are moderate [30].

5.1.2 Target of Evaluation

The Keon CA's TOE includes multiple components. The TOE boundary is indicated in Figure 5.1 by the darker shaded area. The TOE's explanation can be found in [30].

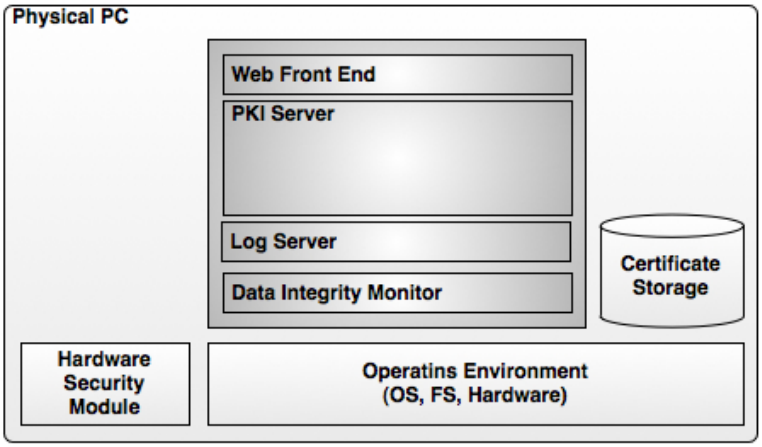

Figure 5.1: The TOE Boundary of Keon CA system

5.1.3 Security Environment

The security environment sets by the Protection Profile and Security Target documents specify all the security concerns that a software product has to deal with. This includes threats, organizational security policies and security assumptions.

5.1.4 Security Objectives

After defining the security environment, security objectives are defined to provide statements to address and counter the identified threats and maintain organizational security policies. The CIMC defines three types of security objectives: security objectives for the TOE, security objectives for the TOE Environment and for both. The Security Objectives described in the Keon ST are taken directly from [35].

5.1.5 Security Requirements

The security requirements section is the longest in the Security Target. This is because the security requirements need to be described clearly, including all the needed details to avoid any ambiguous interpretations. The requirements in CC are divided into two categories: SFRs and SARs [23]. In Keon ST's case, the SFR are divided into two categories: functional requirements for the TOE Environment and functional requirements for the TOE [30]. In SAR's case, all those that are specified in the CIMC Protection Profile for SL 3 are met. Additionally all the SAR specified in EAL4 are met. This increase is obtained including the *Flaw Reporting Procedures*, as required by the [35].

5.1.6 Evaluation Process

The evaluation process analyzes all the aspects presented above in this survey. It's done in conformance with all the documentation produced for achieving the certification. The documentation necessary for this certification can be found in [14]. The Evaluation Team conducted the evaluation in accordance with the CC v2.3 [22] and the CEM [8]. This

evaluation's report explains the whole process [14]. The testing process is not included in this dissertation because we believe that it's outside it's scope.

In the next section we present a similar certification process for the CESeCore Project.

5.2 The CESeCore Project

The CESeCore project aimed to develop a security core with common security functions for Certification Authority (CA)s and certify this core with a Common Criteria EAL4+ certification. The Common Criteria certification will make the CESeCore security core publicly available for integration in numerous security based applications. The CESeCore project (Apr 2009 - Out 2011) was ranked 15th amongst 111 positively evaluated EUREKA's Eurostars Program funded projects.

The CESeCore is a security core, in the form of a common security function Java library, that will provide a reusable base for third-party trustworthy systems. In order to make the CESeCore security core publicly available for integration in numerous security based applications, the CESeCore project aims to undertake a CESeCore Common Criteria EAL4+ certification.

CESeCore integrators will be able to correct, improve and extend their applications at any time without the need to perform frequent system re-evaluations, nor perform continuous checks of the security functions implemented by the security core (including features like digital signature creation/validation, digital certificate and CRLs creation, key management and maintenance of a secure audit log).

A consortium of four IT security companies is promoting the CESeCore project: PrimeKey (main partner, Sweden), MULTICERT (Portugal), E-Imza (Turkey) and Commfides (Norway). As a natural consequence of the project, each partner will integrate CESeCore with its products, such as PrimeKey's EJBCA or Multicert's Timestamp.

5.2.1 Protection Profile and Security Target for CESeCore

In the CESeCore project, as it was undertaking the first CC evaluation of any element of the consortium, there was no experience in choosing Protection Profiles. The core of CESeCore has features of a normal CA, but the applications that are intended to run above it have different purposes. For example, Multicert wants to apply CESeCore in its time stamping services. The Timestamping service allows a user to be sure about the time at which each particular document was created, modified, or came into someone's possession. This includes some features that are not usually present in a CA. With this in mind, in the beginning, the creation of a Protection Profile that covered all the needs for CESeCore Project was considered. This was set aside when they realized the complexity of creating a new Protection Profile and having it certified.

The process of creating a new Protection Profile is complex, as it is necessary to state the TOE for the new type of products, develop all the threats, security objectives, assumptions, OSPs, SFR and also rationales for every one of these items. However, the major problem associated with the creation of a new Protection Profile is to achieve its certification. Along with the huge amount of work, a great expense also comes with this. In terms of time, this will also significantly increase the project's duration. With this in mind, the idea of

creating a Protection Profile specific to the project was set aside.

After some research about Protection Profiles for CAs, the CIMC Protection Profile [35] was found. The CIMC is a Protection Profile for CA that already had some products certified. The CIMC has all the features required for the Project CESeCore and introducing some changes at the Security Target level is enough to serve all the purposes of every product suggested by the consortium that will interconnect with the core.

5.2.2 Description of CESeCore

The CESeCore is a software product that provides a security core for the development of trustworthy systems, gathering common security functions typically found on such systems into stable, well-defined and self-contained modules. A security vendor willing to develop a new trustworthy system (i.e., an application) may build its specific functionalities on top of the CESeCore security functions that are provided out of the box. By using a stable CC EAL certified core, the integrator is able to continuously extend its system without the need to perform a re-evaluation of its entirety, specifically the security functions.

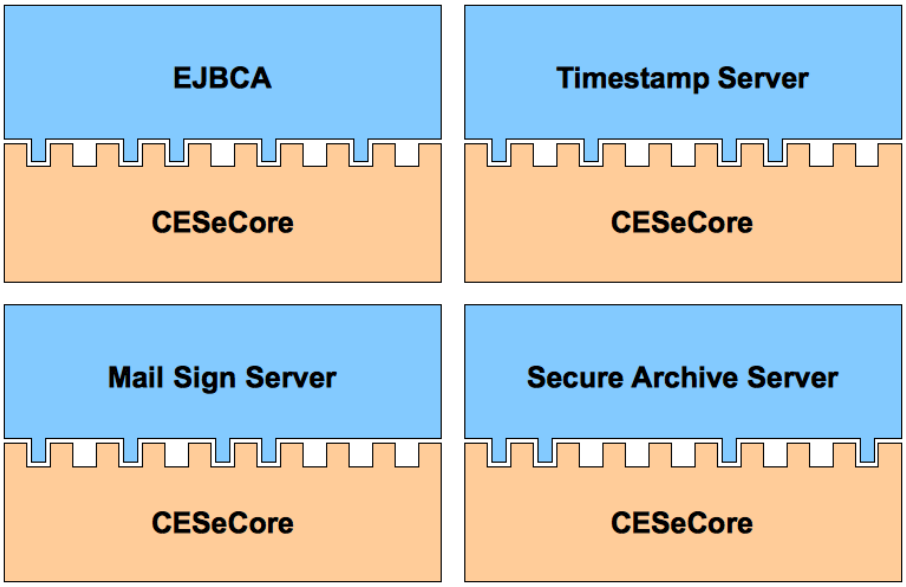

Figure 5.2: Integration of CESeCore with other applications

The CESeCore provides its functionalities to external applications by means of APIs and a combination of configuration parameters. Different applications may use different selected API methods and customized configurations to implement security functions, as illustrated in Figure 5.2. The main security functions provided in the CESeCore are:

- Electronic signatures creation;
- Creation of digital certificates and CRLs;
- OCSP support;
- Data integrity protection;
- Secure audit logs;

- Authentication and authorization management;
- Token management;
- Key generation and management;
- Backup of system data.

The rest of this document describes the CESeCore TOE that is in the scope of this CC evaluation and the corresponding Security Target.

5.2.3 Target Of Evaluation

As illustrated by Figure 5.3, the TOE includes: the CESeCore component and configuration artifacts. Excluded from the TOE are: hardware and operating system platform (abstract machine); application server and execution environment; hardware security module (HSM) and the database engine. The rationale for excluding components from the TOE is explained in the Security Target [37].

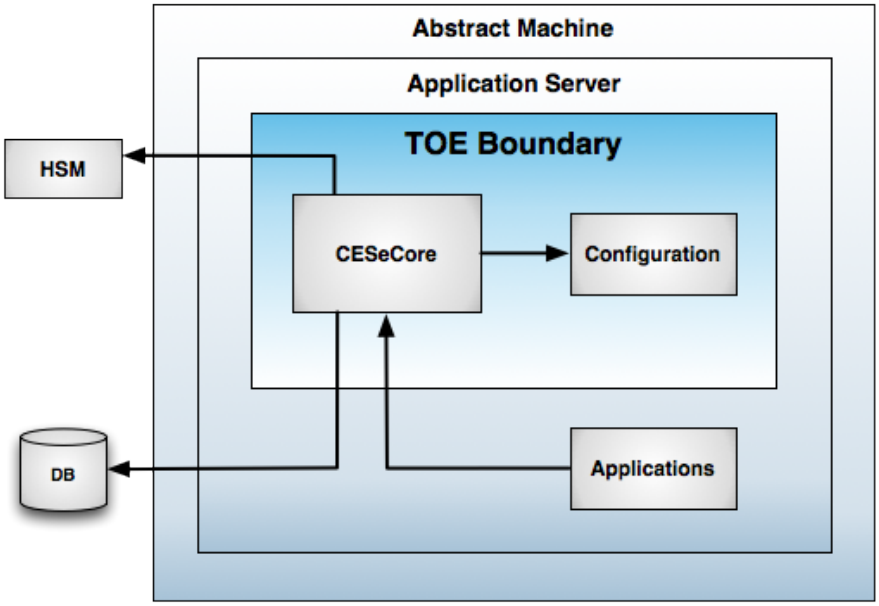

Figure 5.3: TheTOE Boundary of CESeCore

The usage of CESeCore relies not only on its implementation, but also on several other additional components described in the following subsections.

CESeCore The CESeCore component consists of a set of Java classes that provide the functionalities presented above.

Given CESeCore's modular structure, in order to provide more advanced functionalities, the TOE also includes a set of JEE components that can be deployed in any JEE compliant application server. Additionally, depending on the exact functionalities required by the client application, CESeCore can be configured to include or exclude certain parts.

Configuration artifacts Configuration artifacts are basic TOE configuration items provided by the CESeCore dependent application. The configuration artifacts define details on how the specific instance of the TOE works and consist of key-value pairs, stored in a configuration file or in a database. Examples of configuration artifacts are PKCS#11 library path for the hardware security module (HSM), key labels for cryptographic keys and modes for secure audit.

However, in order to run in a CC certified configuration, certain restrictions on the configuration artifacts may apply.

Java Virtual Machine CESeCore is developed in the Java programming language and, as such, runs in a Java Virtual Machine (JVM). Additionally, since the JVM specifications are public, it can be implemented by independent vendors.

Application server CESeCore can be (optionally) deployed on a JEE 5 compliant application server, which provides a number of resources and services to CESeCore, namely:

- Database connectivity services (e.g. object mappings and connection pooling);
- Component creation and management (e.g. session bean pooling and life-cycle management);
- Communication interfaces (e.g. HTTP and JEE).

These resources and services not only make development and maintenance more efficient, but also enable high performance, scalability and availability.

Database Data persisted by CESeCore is handled by a standard relational database, where the following information is kept:

- Key pairs and references to key pairs;
- Certificates and CRLs;
- Audit logs of all security relevant operations;
- Authentication data, such as TOE user information;
- Authorization data, such as which TOE user is authorized to which resources.

CESeCore enforces access control and maintains integrity of the data for which it is required.

Cryptographic module All cryptographic operations performed at the request of the TOE should take place in FIPS 140-1 (or higher) validated cryptographic modules, either in software or in a hardware (HSM). The interaction with the cryptographic module is performed through a standard PKCS#11 library provided by the respective vendor.

5.2.4 Security Environment

The security environment set by the Protection Profile and Security documents specifies all the security concerns that a software product has to deal with. This includes threats, organizational security policies and security assumptions. The CESeCore Security Target follows the security environment described in the CIMC. However, the Security Target is independent to choose its own security environment.

Threats The threats are organized into four categories: authorized users, system, cryptography, and external attacks. We present a example retrieved from the CIMC to the category authorized users: *T.Critical system component fails*: Failure of one or more system components results in the loss of system critical functionality.

Organizational Security Policies The Organizational Security Policies (OSPs) are all the procedures defined by the organization deploying the certified product to protect sensitive data. An example retrieved from the CIMC: *P.Authorized use of information*: Information shall be used only for its authorized purpose(s).

Security Assumptions The usage assumptions are organized into three categories: personnel (assumptions about administrators and users of the system as well as any threat agents), physical (assumptions about the physical location of the TOE or any attached peripheral devices), and connectivity (assumptions about other IT systems that are necessary for the secure operation of the TOE). An example of a security assumption for the Personnel category: *A.Auditors Review Audit Logs*: Audit logs are required for security-relevant events and must be reviewed by the Auditors.

Security Objectives

Based on the statements defined in the security environment, the security objectives provide statements to address and counter the identified threats and maintain the OSP. CIMC divides the security objectives into three types: security objectives for the TOE, security objectives for the environment, and security objectives for both the TOE and environment. In the case of the security objectives for the TOE, they are divided into three types as the threats presented before. The security objectives in the Security Target are retrieved from the CIMC.

Extended Components Definition

This section doesn't exist in the CIMC, but it does exist in the Security Target for CESeCore. In this section we present the components created in the CIMC that only exist for the propose of this Protection Profile. In this section it is also possible to create new components exclusively for the Security Target. In the case of CESeCore no component is created. We can see some examples of extended components requirements retrieved from the CESeCore Security Target in Table 5.1.

Table 5.1: Extended Security Requirements on CESeCore

Extended Security Requirements	CIMC Page Reference
FCO_NRO_CIMC.3	49
FCO_NRO_CIMC.4	51
FCS_CKM_CIMC.5	53
FDP_ACF_CIMC.2	52
FDP_ACF_CIMC.3	53
...	...

5.2.5 Security Requirements

The security requirements section is the one that occupies the largest portion of the Protection Profile and the Security Target. This happens because security requirements need to be described clearly, including all the needed details to avoid any ambiguous interpretations. As we have seen throughout this dissertation, security requirements can be classified in two categories, namely SFRs and SARs. In the case of the CIMC the security requirements are divided into several chapters: the security functional requirements that are applicable to the IT environment, the security requirements that are applicable to the TOE and the TOE assurance requirements. The security requirements are divided into classes for the SFR and the SAR, that we have presented respectively in Table 2.1 and Table 2.2. The CIMC has the following SFR classes that are retrieved for the CESeCore Security Target:

- Security Audit (FAU)
- Security Management (FMT)
- User Data Protection (FDP)
- Identification and Authentication (FIA)
- Communications (FCO)
- Cryptographic Support (FCS)
- Protection of the TSF (FPT)
- Trusted Paths/Channels (FTP)

An example of an SFR that is included in the CESeCore from the CIMC is the *FCO_NRO_CIMC.3* that is stated in the CIMC as *Enforced proof of origin and verification of origin.*

In the case of the assurance components, the CESeCore certification includes all the components present in Table 2.4 with the addition of the *ALC_FLR.2* component that is used for *Flaw reporting procedures.*

In conclusion, the goal of the CESeCore Project is to receive a CC certification EAL4 augmented by ACL_FLR.2, which requires the TOE to be "Methodically Designed, Tested and Reviewed" (Table 2.3). EAL4 analyzes the TOE using functional specifications, design

documentation, as well as all security related documentation. To increase the level of assurance, EAL4 looks also to aspects related to the development environment, configuration management and delivery procedures.

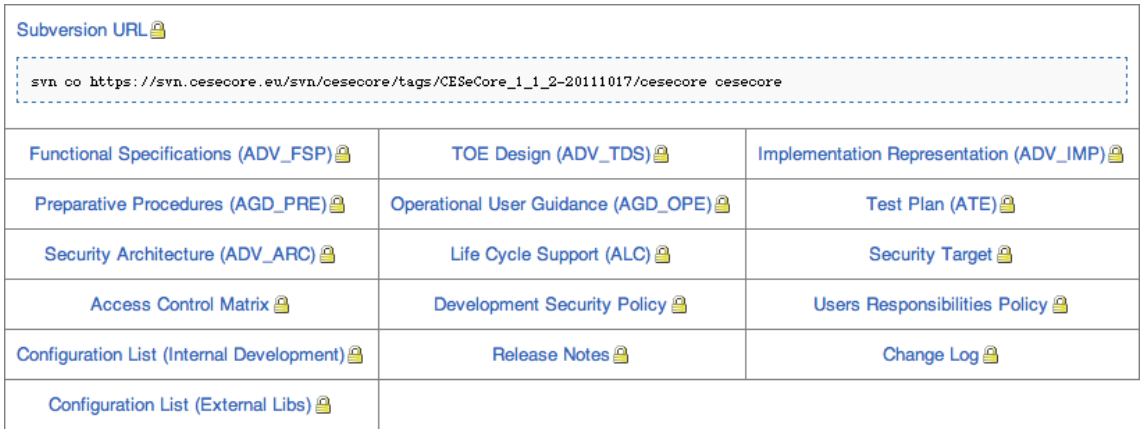

Figure 5.4: Deliverables for the CESeCore Project

We can see all the evidence produced for the certification in Figure 5.4. This evidence fulfills the assurance requirements that we have presented before for the EAL4 augmented with ACL_FLR.2.

5.3 Validation of the CESeCore Documentation

In the section above we have introduced the two case studies that we have used in the context of this work. In this section we will present how we have used the models presented in Chapter 4 to validate the CESeCore documentation. We have applied the models to both case studies. However, seeing as they are similar, we only show the CESeCore application in this work . CESeCore was founded in 2008, and when we developed the models in Alloy to aid in document production, all CESeCore documents were already completed and approved by the evaluation lab. As such, we did not contribute to their production. However, these results are useful contributions for future certifications processes.

To demonstrate our application of the Alloy models we will follow the same structure of their presentation in Chapter 4. For starters we will look to the Abstract Model application for a real case study. To fill in the information in the Full Model we use the Sensitive Information developed for CESeCore. After including all the information present in the Full Model we can validate it by applying the consistency checking assertions. To finalize, we apply the validation of the SFR Dependencies present in the CESeCore Security Target and generate the SFR Tracing for the Functional Specification submitted as CESeCore evidence.

5.3.1 Abstract Model Application

To begin the presentation of the Alloy models in a real case study, lets look at the Abstract Model. In this case we have already presented the whole Alloy specification in Section 4.2 and, as we have stated before, the Abstract Model is the most abstract level of our modulation that is good for all CC certifications. So in this case we have nothing to prove for

CESeCore. However, we present in Figure 5.5 an instance of the Abstract Model where we can see how the concepts are interrelated in these models.

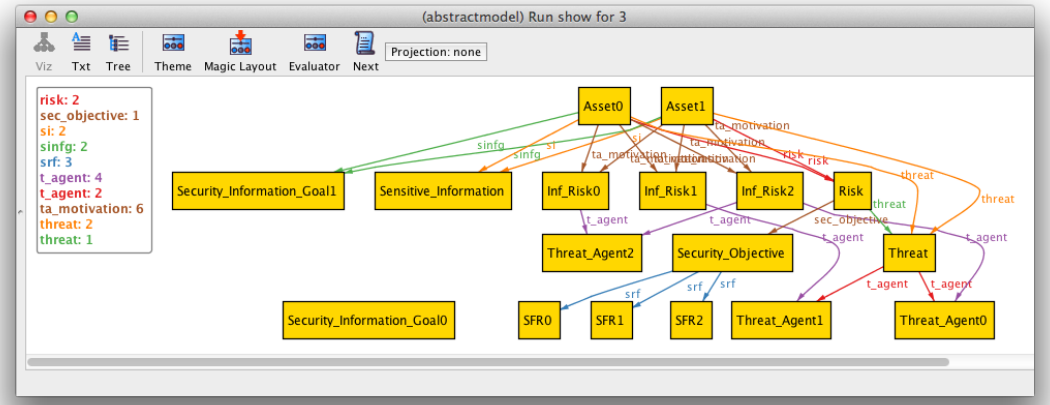

Figure 5.5: Abstract Model Instance for a Run with 3 of Scope

We can notice that the AA generates atoms for each concept following the rules imposed by us. For example, we can see that two assets are generated *Asset0* and *Asset1*. This instance is generated using a *show()* predicate.

5.3.2 Sensitive Information in the Full Model

The Full Model is the part of the Alloy model where we include information from the documents. To perform our demonstration we have chosen information from the Sensitive Information for CESeCore. Sensitive information contains all the concepts present in the Abstract Model. As such we need to fill in the Full Model with that information. If we *run* the *show()* predicate for the Full Model, the AA will generate an instance of the model as we can see in the Figure 5.6.

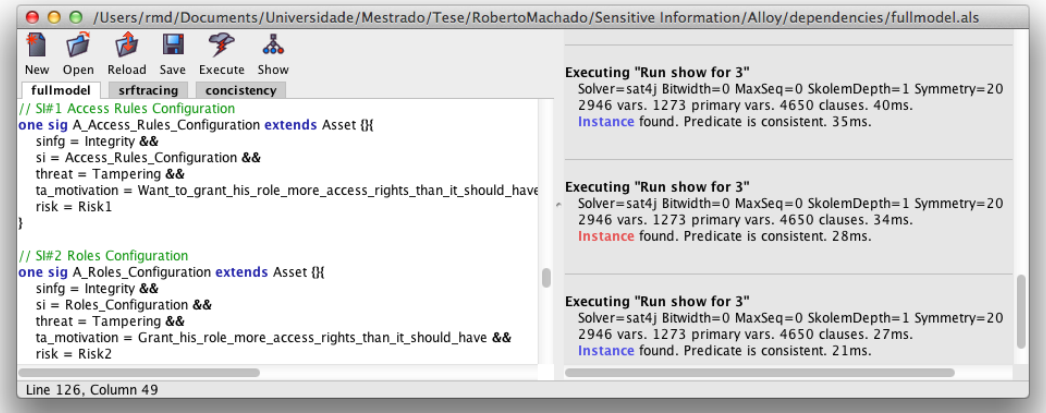

Figure 5.6: Sensitive Information included in the Full Model

Figure 5.6 shows part of the Alloy model and the last commands executed in the AA. We can see that "Predicate is consistent", so an instance is provided for this execution.

However, if the information introduced does not respect the restrictions imposed by the Abstract Model, an instance is not founded by the AA. Figure 5.7 shows an example of a *run* execution that result in a "Predicate may be inconsistent". We can see in Alloy that in the definition of the Asset we have through the *sinfg* relation that we have two Security Information Goals, when in the Abstract Model we have stated that each Asset have only **one** Security Information Goal.

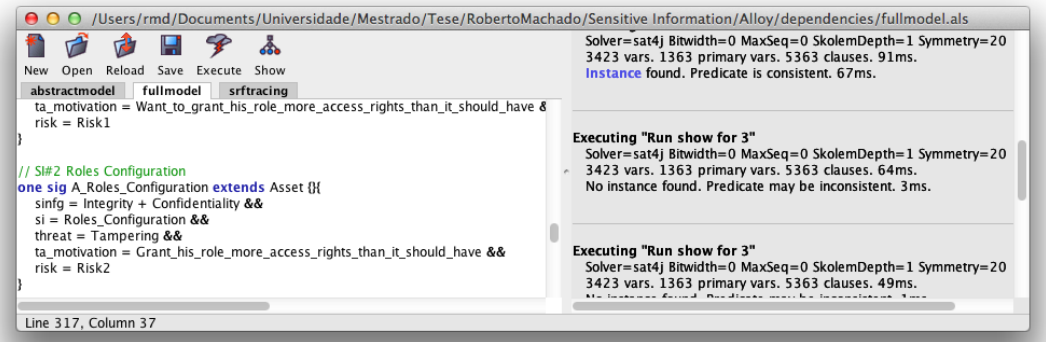

Figure 5.7: Validation of the Full Model - inconsistent

We use the *show()* predicate to see if the model is consistent with the restrictions presented in the Abstract Model. In the next section we will present the consistency checking for properties that we retrieve from the documentation.

5.3.3 Consistency Checking for the Sensitive Information

Now that we have introduced the information that we want to analyze in the Full Model, we can perform the validation of the documents included there. The validation is made through a set of assertions included in the consistency checking part of the Alloy model.

For example, we want to be sure that the Full Model doesn't have two Assets with the same Sensitive Information and Security Information Goal. It doesn't make sense for two Assets with the same definition to exist. Figure 5.8 shows Alloy defining this assertion. We also can see the *check* command executing the assertion.

Figure 5.8 also shows, on the right side, that "No counter example found". This means that no counterexample for the assertion exists in the information present in the Full Model.

In case the assertion is not valid, the AA generates an instance of a counterexample. It is necessary to remember that the AA's SAT solver only verifies a certain scope given by the user. This means that if no counterexample is found we can only guarantee that the assertion is valid for that scope.

Another assertion example is that in an Asset we have Threat Agents associated with two relations. We have Threat Agents that are related to the Information Risk and to the Asset through the *ta_ motivation*; and we have Threat Agents associated to the Threat

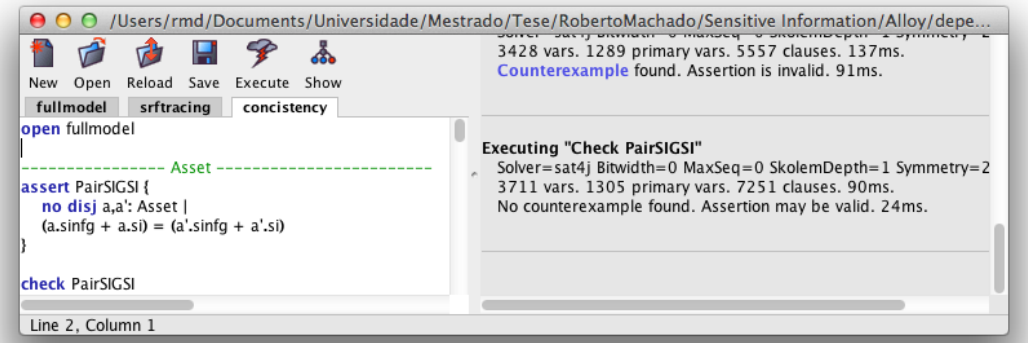

Figure 5.8: Example of a correct assertion in the model

that are also related to the Asset. Figure 5.9 shows the Alloy for an assertion that tries to find, in the Full Model's whole Asset, if there is a case where the Threat Agents associated with the Information Risk are not associated with the Threat that is related to the Asset. We must verify that all Threat Agents related to the Information Risk are also related to the Threat. The contrary, however, is not necessary.

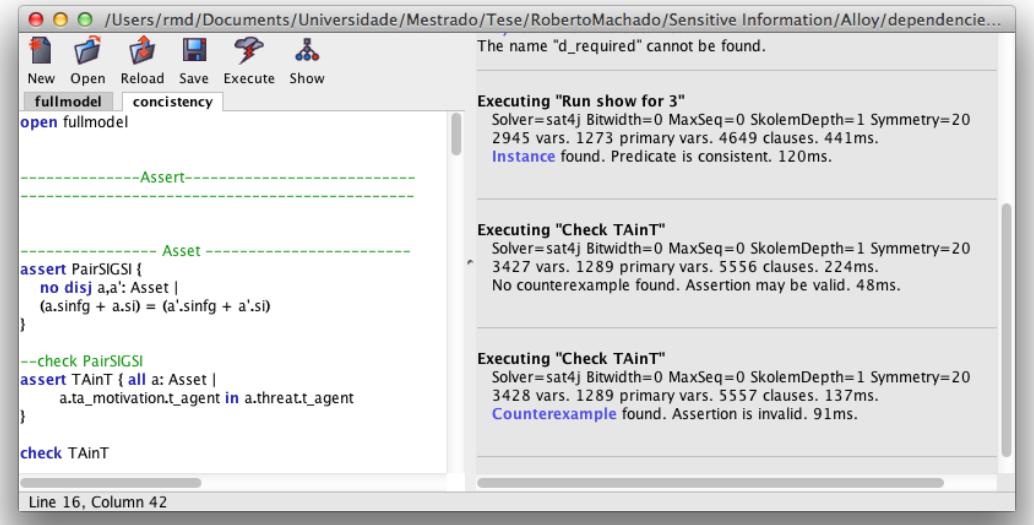

Figure 5.9: Example of a assertion that creates a counterexample

We executed the *check TainT* command and the result was "Counterexample Found". In this case, the AA have found a counterexample for these assertions. This means that somewhere in the Full Model we have an Asset where the Threat Agents associated to the Information Risk which is related to an Asset are not all present in the Threat that is related to that Asset as well.

We can open the counterexample and see that instance. We have different possibilities to

analyze the counterexample. We have chosen a way that fits inclusion in this document better. Figure 5.10 shows a view of the counterexample in the option Tree.

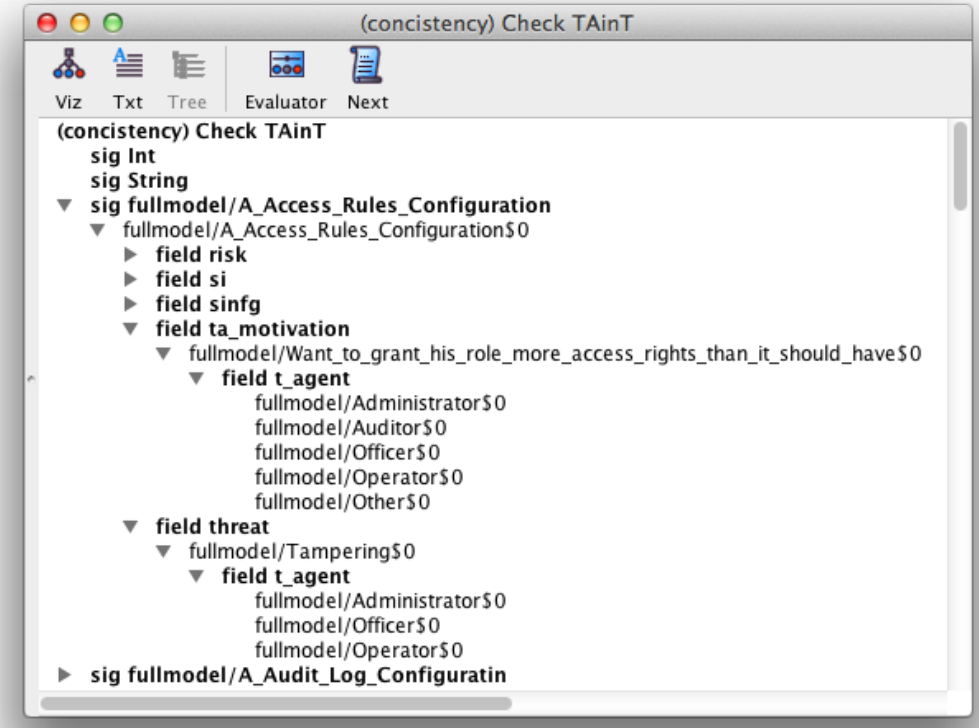

Figure 5.10: Counterexample visualized in the Tree option

That which we can see in the *sig full model*$A\_Access\_Rules\_Configuration$ is an Asset. We have the *field* $ta\_motivation$ and the *field threat* open. It is easy to see that the Threat Agents present in the $ta\_motivation$ are not all present in the *threat*.

5.3.4 SFR Dependencies

In Section 4.4.2 we have presented the application of the Alloy models to validate the SFR dependencies. Here we will present the application on the dependencies present in the CESeCore Security Target. SFR are a long catalogue with dependencies between them. For this case we only will consider the dependencies required and analyze if they are included or not.

Figure 5.11 shows the list of declaration dependencies for two SFRs. We can see that these SFRs have both dependencies required and dependencies inertly required. We now need to include the dependencies imposed in the CC standard. Figure 5.12 shows the declaration of *Components* that will connect the SFRs present in the Full Model, retrieved from the Security Target and the list of dependencies present in the CC standard for each SFR. This also shows the assertion that will verify if all the required dependencies are considered in the Security Target.

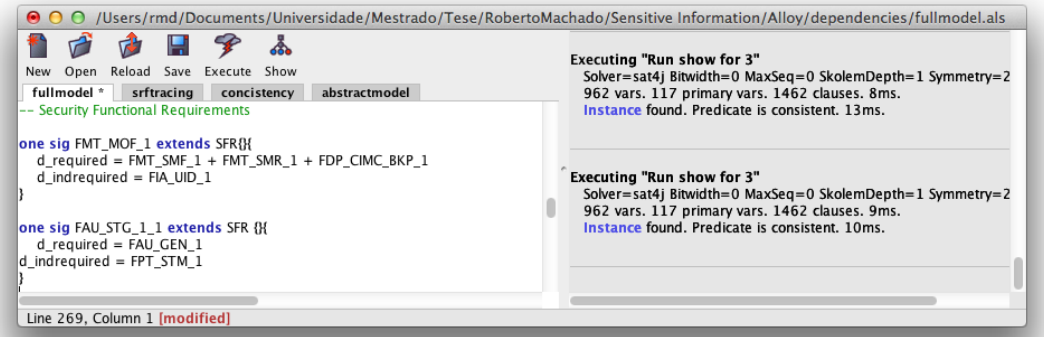

Figure 5.11: Exemples of SFR Dependencies in Alloy

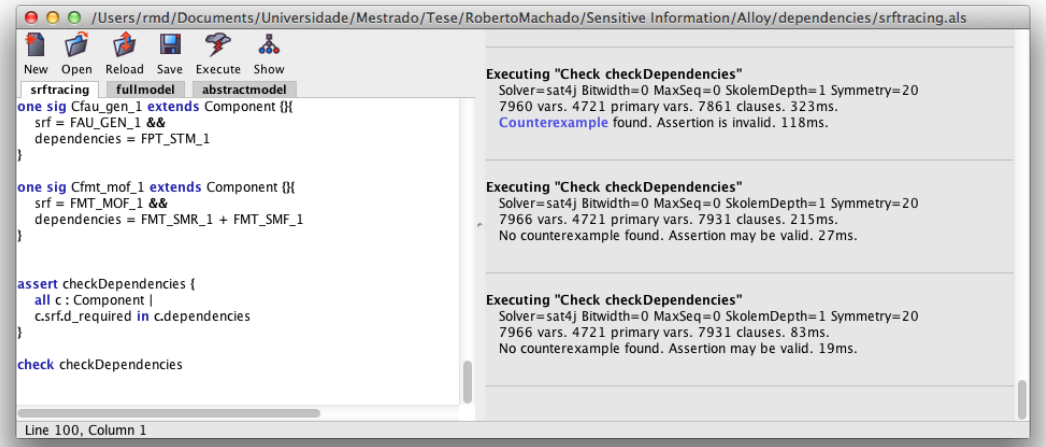

Figure 5.12: Components declaration and assertion for Dependencies

The assertion will check in all the Components if the present SFR has the dependencies required by the CC standard. To execute the assertion we have the *check checkDependencies* command. In this case, no counterexample was found. All SFR dependencies required in the CC standard are considered in CESeCore's Security Target.

5.3.5 SFR Tracing

In Section 4.4.3 we present the last considered application for our Alloy models. The SFR tracing intend to help in the generation of the SFR mapping from the Security Target to the TSFI that are documented in the Functional Specification document.

Figure 5.13 shows the declaration of the new *Interface* concept where we declare the several Interfaces existent in the Functional Specification. For each of them we associate the Asset and the SFRs related to that Asset. In the figure, we can also see the assertion that will analyze if all the Interfaces have the correct correspondent SFRs, for all SFRs that exist in the Security Target.

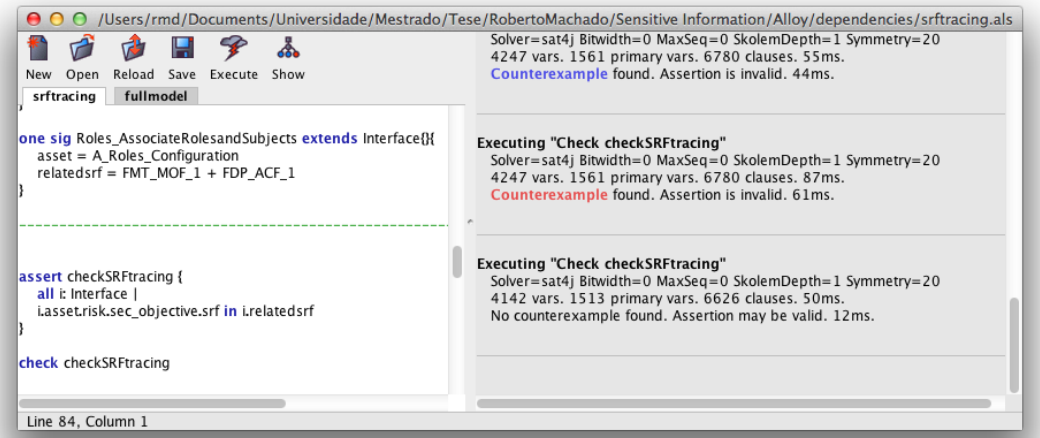

Figure 5.13: SFR Tracing for the Functional Specification

5.4 CC Documentation in XML

In Chapter 4 we have presented an Alloy model that needs to be filled with information from the CC documents. This raises a problem: usually the CC documents do not exist in a machine-readable format. They are usually produced in text editors with a bunch of tables without a standard format. This forced us to consider a better way in which to represent the CC documentation.

The CC standard is available in Portable Document Format (PDF) and also in XML [33]. Since the standard is already available in XML we realized that the documentation for a CC certification can also be produced in this format. The first step that we had to take was to create versions in XML of the documents that we were using as case studies. We created a Protection Profile and a Security Target for the CESeCore case study presented in Section 5.3.

After this, which we needed for our work, we realized we could do more and produce something that people could reuse when pursuing a CC certification. As such, we have also created a set of templates for Protection Profiles, Security Target and Sensitive Information in XML. This is also used in the application presented in the next section. Figure 5.14 shows how we utilize XML in this work to provide the CC documents with another use.

We can see in the figure the flow created to use the XML documents. The templates created serve as guidance for the document's development, having all the elements that are recommended in the CC standard. We can then create XML documents containing all the elements of a normal Protection Profile or Security Target. Afterwards, we are able to use the XML documents with the models presented in Chapter 4, or in the toolbox presented in the next section.

The scrips created to parse the XML documents were previously developed in Perl. With these scrips we intended to generate theAlloy models to perform our analysis, creating a script for each type of documents. We also have developed Java scripts to be used standalone to create Alloy documents. The toolbox that we present in the next section

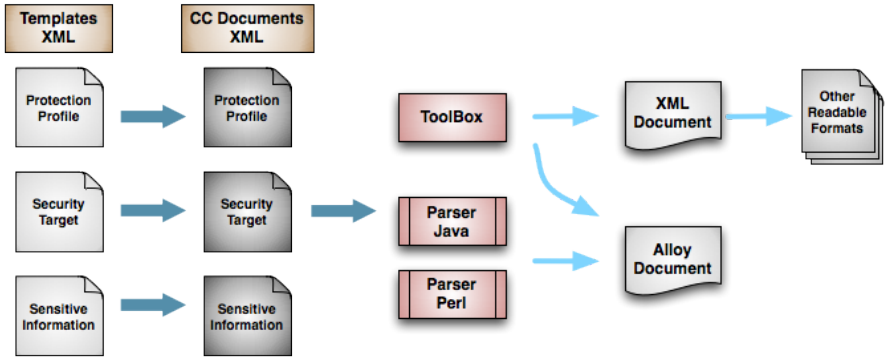

Figure 5.14: CC Documentation in XML

will allow to generate documents in XML and in Alloy.

This work with XML documents for the CC certification process has created new possibilities to work with these same documents. The toolbox that we present in the next section is an example of how useful this format could be in the context of CC certifications. With this exploration we where able to define the structure of what the support application presented in the following section should do.

5.5 ToolBox for CC Documentation Development

This work's fundamental objective is to aid in the development of documents used as evidence in the CC certification process. In this section we will present an application to support our work with Alloy. This is an open source tool, built on top of the Alloy models presented in Chapter 4.

We haven't yet completed this application's implementation. What we will present in this section is the specification of a desktop application development in Java. We present the requirements and the design for the toolbox. To finalize this section we will present the state of the toolbox implementation.

5.5.1 Functional Requirements Specification

This toolbox focuses on the production of Protection Profiles and Security Targets. In this section we will present the requirements for the toolbox. In Figure 5.15 we show the use case for the main features of the application. The features are presented below with a brief description and their requirements.

Create Documents

We can create new documents, Protection Profiles or Security Targets, from an existent document in XML or a completely new document. In the case of a Security Target we must associate a Protection Profile to that Security Target.

1. The ToolBox must allow for the creation of a Protection Profile by specifying a name

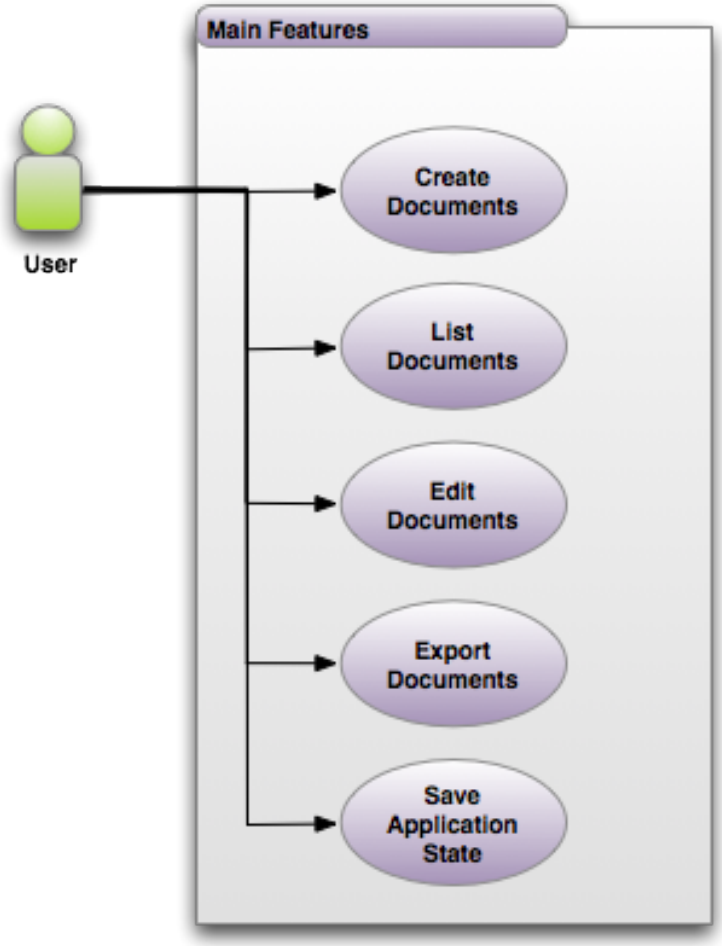

Figure 5.15: Use Case for the Main Features

and CC version.

2. The ToolBox must allow for the creation of a Security Target by specifying a name and CC version.

3. The ToolBox must allow for the creation of a Protection Profile/Security Target from a existent XML document.

4. When a Protection Profile/Security Target is created, it must contain the following predefined main sections: Introduction, Conformance Claim, SPD, Security Objective, SFR, SAR and Rationale.

5. When a Security Target is created, the system must automatically copy the content of all SPD, security objectives, SFRs, SARs and the Rationale from the Protection Profile.

List Documents

We can list all the documents present in the toolbox so as to allow it to select one of them on which to execute other toolbox features.

1. The user must be able to check out any content.
2. The user must be able to add new content.
3. The user must be able to select any and all content to edit.
4. The user must be able to delete any content.
5. The user must be able to list whether text is missing in any part of a Protection Profile/Security Target.

Add Contents to the Documents

After creating a document or selecting one from the list, we can add content to the document.

1. The user must be able to add the following subsections to the Introduction: PP reference (only for PP), TOE overview, ST Reference (only for ST), TOE reference (only for ST) and TOE description (only for ST).
2. The user must be able to add the following subsections to the Conformance Claims: CC conformance claim, PP Claim, Package claim (only for ST), Conformance rationale, Conformance statement (only for PP).
3. The user must be able to add multiple threats OSPs and assumptions to the SPD section.
4. The user must be able to add multiple security objectives for the TOE or Operational Environment to the Security Objective section.
5. The user must be able to add a component fount in the SFR catalogue to the SFR section.
6. The user must be able to add a component fount in the SAR catalogue to the SAR section.
7. The user must be able to create and add extended components.
8. The user must be able to create and add assets and relate them with security objectives and threats.
9. The user must be able to create and add Threat Agents and relate them with Assets.
10. The user must be able to relate Threat Agents with Threats and with Information Risk.
11. The user must be able to add multiple mapping between security objectives and SPDs.
12. The user must be able to add multiple mapping between security objectives for the TOE and SFRs.

Alloy related requirements

The requirements of this section describe how we can use the Alloy models in the toolbox.

1. The user must be able to validate all the restrictions imposed by the General Model.
2. The user must be able to validate if all threats are satisfied by a security objective.
3. The user must be able to validate if all security objectives are satisfying a threat.
4. The user must be able to validate if all security objectives are satisfied by an SFR.
5. The user must be able to validate if the all SFRs are satisfying a security objective.
6. The user must be able to validate if the SFR Dependencies are correct in the Protection Profile/Security Target.
7. The user must be able to validate the generation of an SFR Tracing from a Security Target for a Functional Specification.

Export Documents to XML and Alloy

At any time that we are working with a document we can export it to XML or to Alloy. This is useful, for instance, at any moment we are performing analyses with the models presented in Chapter 4.

1. The user must be able to export a Protection Profile/Security Target to Alloy.
2. The user must be able to export a Protection Profile/Security Target to XML.

Additional features

1. The system must be able to build a CC component catalogue based on an CC XML-file.
2. The system must provide a list of EALs to relate with any document.
3. The system could offer a search-functionality.

5.5.2 Non-Functional Requirements

To clarify requirements not related to the actual features of the toolbox, requirements to the project are identified here.

1. The system must be developed as a desktop application.
2. The system must be available in a friendly Graphical User Interface (GUI).
3. The data must be saved in a persistent manner.

5.5.3 Design

The last section we have presented the requirements for our toolbox. This section will present an overview of how the toolbox will be designed in order to fulfill the requirements.

Three-layer Architecture

The overall architecture used for the toolbox is based on a three-layer architecture: presentation, business and data. This is based on the client-server pattern:

- The presentation layer creates the GUI with which the user interacts. This layer works as a client to the business layer.
- The business layer contains the business logic which means that it controls the system's functionality. This layer works as client for the data layer and as server to the presentation layer.
- The data layer contains all the data. It works as a client for the the business layer.

This design defines how our implementation will be structured.

Logic Model for the ToolBox

To represent all the information present in the CC documents, the toolbox will have a set of classes for each of the document's components.

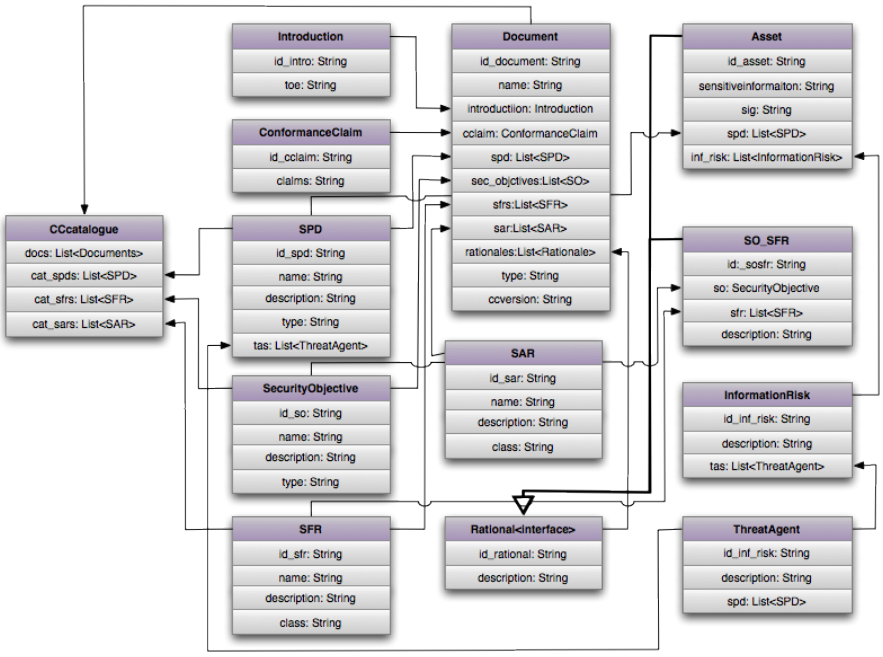

Figure 5.16: Logic Model for the ToolBox

Figure 5.16 shows the classes diagram for the application's main components.

5.5.4 State of Implementation

Here we make a quick overview of the toolbox implementation state at this moment in time. We present the technologies used to develop the application, how the code is structured and what we considered to storage the data present in the application.

Technologies

The application was developed with Java in the NetBeans Integrated Development Environment (IDE). The GUI was developed with Swing. We have also used Java and Perl to develop the parsers for the application.

Code Structuring

The code is structured as a NetBeans project solution containing three packages:

- A package with the data layer with all the CC concepts and their respective contents.
- A package with the business layer for all CC concepts.
- A package with all the classes that give origin to the application's GUI.

The code can be found in a free repository at GitHub[2].

Storage

As this application was only a proof of concept we have decided against the creation of a database. We have used a Java persistence mechanism instead, which allows us to save the state of the objects at a given point and reload back to the same state. This mechanisms are called ObjectStreams.

At this moment, lacking conclusion, is the Alloy part of the business layer and almost everything in the presentation part. We hope to conclude this dissertation is presented for evaluation.

[2]https://github.com/bertoluchi07/CCtoolbox

Chapter 6

Conclusions and Future Work

6.1 Conclusion

This work presents an overview of CC certification, based mostly on evidence production and includes a new approach using Alloy to validate the main CC documents. CC is a well recognized standard to achieve security certification of software products. These certifications are mostly based on the production of documents as evidence that the product really does what the vendors claim. CC documentation is something complex to produce and it can cost the organizations trying to certify their products with it time and money. The certification process could take months to years depending on the deepness of the EAL chosen for the certification. The number of documents and their rigor are also based on the EAL.

In this work we have presented the structure of the main documents used in CC. We have stated that these documents are properly interrelated between them. Their development is mostly a systematic process, which makes this a tedious work, leading to mistakes in the documentation that could delay their acceptance by the evaluators. We have seen that the evidence submission process is cyclic and that only ends when the evaluators approve the evidence. This implies a correct development, to cut the cycles necessary to approve some type of evidence. To assure that the documents used as evidence in a CC certification are correctly implemented is the main objective for this work.

Certifications with CC give origin to a significant set of documents. However, in the context of this work we have identified the most important documents to analyze them and provide tools to help in their development. We have identified the Protection Profile, the Security Target and the Functional Specification as the main documents for the CC certification. Another consideration for this work was the the CC's General Model which is the model that relates the concepts present in the CC. We have remarked that these documents have different abstraction levels, given them a hierarchical relation, from the General Model (in the CC), the Protection Profile, which is an instantiation of the General Model for a type of software product, to the Security Target, which is a subset oriented to a specific product of the Protection Profile. Finally, we have also taken into consideration the Functional Specification where the security specified in the Security Target is oriented towards the implementation of the product.

For each of the documents selected for analysis we have stated the main consistency con-

cerns to validate and also found elements in the documentation, such as the SFR Tracing and the SFR Dependencies, which are particularly difficult to elaborate. We chose the Alloy language to elaborate models for the CC documents that we use to validate the properties and also to help the evaluation of the elements presented before. Alloy is a powerful yet simple tool, allowing for small models to be built quickly and analyzed with the AA SAT solver.

The model developed in this work with Alloy follows the same abstraction philosophy of the CC documents. We have created an Abstract Model to represent the most abstract level of the documents, the General Model of the CC standard. On top of this we have a Full Model that we use to include the information from the documents. Using these two parts of the model we can analyze the document's consistency and generate information useful for the certification process, such as the SFR Tracing that could be used to fulfill that requirement in the Functional Specification or to validate an already existent SFR Tracing.

Another way to aid in the production of CC evidence documents that was also considered in this work was a desktop application in Java to help in the production of Protection Profiles and Security Targets. At first we considered the problem of loading information from existent documents. As such, we created XML templates for Protection Profiles, Security Targets and Sensitive Information. These templates can now be used to develop documents in a format easily readable by a machine. We have also developed scripts in Perl and Java to easily parse documents in XML to Alloy documents. All of this work has contributed to the development of the support application. The toolbox allows us to load components from existing documents in XML, to edit these documents and to produce a new version. The main objective for this toolbox is the inclusion of the Alloy models, making it easy to work with them even if one doesn't have experience with Alloy.

To apply this work we have studied a lot of CC certifications documents. However, we have focused on two case studies: the Keon CA and the CESeCore certifications. Both of these projects have achieved a successful certification. In the case of CESeCore, we have participated in the project and have also created a document to describe the whole certification process followed by the CESeCore project. We have chosen CESeCore to show the model's application and also where we had the possibility to help in the document's development and to contribute with corrections retrieved from these models.

In short, with this work we have developed some solutions for the always tedious work that is writing CC documents. We also afford the possibility to generate information that will serve as a solution for parts of the systematic process that exist when we are producing documentation for CC. Although these last points are quite useful, the main goal of this work is the introduction of guarantees for Alloy use. We have developed a toolbox that supports the main CC documents' whole development. This is absolutely necessary to save money and to gain time in the certification process.

6.2 Future Work

We think we were able to document the CC certification process and the evidence production process to achieve a certificate, mostly in that which concerns the Protection Profile and the Security Target. The next step would be to study other documents used as evidence for CC certifications and try to understand how we could use Alloy to assure their

correct development. In the Protection Profile and Security Target analysis there are also properties that could be further developed so as to cover more parts of the documents, for example, a complete analysis of all the rationales present in the documents.

Besides complete completing the development of the application to aid in the production of documents, we left some improvements that could be made to provide a complete development of the documents in the application as future work. Features to implement in the future are: the possibility to export the documents in several formats, to make a document deliverable for evaluation directly from the application possible, a database to support the storage of the several documents' components, and also the possibility of developing more types of CC documents. The application also needs its interface improved into a friendlier look and feel.

References

[1] J. Almeida, M. Barbosa, J. Sousa Pinto, and B. Vieira. Verifying cryptographic software correctness with respect to reference implementations. *Formal Methods for Industrial Critical Systems*, pages 37–52, 2009.

[2] J.B. Almeida, M. Barbosa, J.S. Pinto, and B. Vieira. Deductive verification of cryptographic software. *Innovations in Systems and Software Engineering*, pages 1–16, 2010.

[3] S.R. Band. Comparing insider it sabotage and espionage: A model-based analysis. Technical report, DTIC Document, 2006.

[4] B. Bérard, M. Bidoit, A. Finkel, F. Laroussinie, A. Petit, L. Petrucci, and P. Schnoebelen. *Systems and software verification: model-checking techniques and tools*. Springer Publishing Company, Incorporated, 2010.

[5] J.P. Bowen and M.G. Hinchey. Ten commandments of formal methods. *Computer*, 28(4):56–63, 1995.

[6] K. Caplan and J.L. Sanders. Building an international security standard. *IT professional*, 1(2):29–34, 2002.

[7] D.M. Cappelli. Management and education of the risk of insider threat (merit): Mitigating the risk of sabotage to employers' information, systems, or networks. Technical report, DTIC Document, 2007.

[8] CCRA. Common Methodology for Information Technology Security Evaluation Evaluation methodology July 2009 Revision 3 Final Foreword. (July), 2009.

[9] E.M. Clarke and J.M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys (CSUR)*, 28(4):626–643, 1996.

[10] E. Damiani, C.A. Ardagna, and N. El Ioini. *Open source systems security certification*. Springer-Verlag New York Inc, 2008.

[11] P.T. Devanbu and S. Stubblebine. Software engineering for security: a roadmap. In *Proceedings of the conference on The future of Software engineering*, pages 227–239. ACM, 2000.

[12] A.A. El Ghazi and M. Taghdiri. Analyzing alloy constraints using an smt solver: A case study. *AFM10 (Automated Formal Methods)*, 2010.

[13] S. Gerhart, D. Craigen, and T. Ralston. Experience with formal methods in critical systems. *Software, IEEE*, 11(1):21–28, 1994.

[14] Shaun Gilmore. Common Criteria Evaluation and Validation Scheme Validation Report RSA Certificate Manager Ron Bottomly. *Engineering*, 2006.

[15] E. Goldberg and Y. Novikov. Berkmin: A fast and robust sat-solver. *Discrete Applied Mathematics*, 155(12):1549–1561, 2007.

[16] A. Hall. Seven myths of formal methods. *Software, IEEE*, 7(5):11–19, 1990.

[17] B. Hashii. Lessons learned using alloy to formally specify mls-pca trusted security architecture. In *Proceedings of the 2004 ACM workshop on Formal methods in security engineering*, pages 86–95. ACM, 2004.

[18] C. Heitmeyer, J. Kirby, and B. Labaw. Tools for formal specification, verification, and validation of requirements. In *Computer Assurance, 1997. COMPASS'97.'Are We Making Progress Towards Computer Assurance?'. Proceedings of the 12th Annual Conference on*, pages 35–47. IEEE, 2002.

[19] Constance Heitmeyer, Myla Archer, Elizabeth Leonard, and John McLean. Applying Formal Methods to a Certifiably Secure Software System. *IEEE Transactions on Software Engineering*, 34(1):82–98, 2008.

[20] Constance L. Heitmeyer. On the Role of Formal Methods in Software Certification: An Experience Report. *Electronic Notes in Theoretical Computer Science*, 238(4):3–9, September 2009.

[21] W. Hisao Higaki. *Successful Common Criteria Evaluations: A Practical Guide For Vendors*. CreateSpace, 2010.

[22] ISO/IEC15408-1. Information technology security techniques evaluation criteria for it security part 1: Introduction and general model. 2009.

[23] ISO/IEC15408-2. Information technology security techniques evaluation criteria for it security part 2: Security functional requirements. 2009.

[24] ISO/IEC15408-3. Information technology security techniques evaluation criteria for it security part 3: Security assurance requirements. 2009.

[25] D. Jackson. Automating first-order relational logic. *ACM SIGSOFT Software Engineering Notes*, 25(6):130–139, 2000.

[26] D. Jackson. Alloy: a lightweight object modelling notation. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 11(2):256–290, 2002.

[27] D. Jackson. *Software Abstractions: logic, language and analysis*. The MIT Press, 2006.

[28] D. Jackson, I. Shlyakhter, and M. Sridharan. A micromodularity mechanism. *ACM SIGSOFT Software Engineering Notes*, 26(5):62–73, 2001.

[29] C.B. Jones. *Systematic software development using VDM*, volume 103. Citeseer, 1990.

[30] RSA Keon. RSA Keon CA system Security Target. *Security*, 2006.

[31] B. Kitchenham and SL Pfleeger. Software quality: the elusive target . *Software, IEEE*, 13(1):12–21, 2002.

[32] A.H. Lin. *Automated analysis of security APIs*. PhD thesis, Citeseer, 2005.

[33] J. Loughry. Use of xml in the design and specification of a new high assurance controlled interface. 2004.

[34] D Mellado, E Fernandezmedina, and M Piattini. A common criteria based security requirements engineering process for the development of secure information systems. *Computer Standards & Interfaces*, 29(2):244–253, February 2007.

[35] NIST. Certificate Issuing and Management Components Family of Protection Profiles. *Management*, 2001.

[36] J. Park and J.Y. Choi. Security policy modeling using z notation for common criteria version 3.1. In *Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on*, volume 1, pages 137–142. IEEE, 2009.

[37] CESeCore Project. Security Target for CESeCore. *Security*, 2010.

[38] R. Richards, D. Greve, M. Wilding, and W.M. Vanfleet. The common criteria, formal methods and ACL2. In *ACL2 Workshop*. Citeseer, 2004.

[39] T. Rottke, D. Hatebur, M. Heisel, and M. Heiner. A problem-oriented approach to common criteria certification. *Computer Safety, Reliability and Security*, pages 213–229, 2002.

[40] John Rushby. Formal Methods and their Role in the Certification of Critical Systems. *Aviation*, (March), 1995.

[41] M. Singh and M.S. Patterh. Formal specification of common criteria based access control policy model. *International Journal of Network Security*, 10(3):232–241, 2010.

[42] Richard E. Smith. Cost profile of a highly assured, secure operating system. *ACM Transactions on Information and System Security*, 4:2001, 2001.

[43] J.M. Spivey. *Understanding Z: a specification language and its formal semantics*, volume 3. Cambridge Univ Pr, 1988.

[44] M. Taghdiri and D. Jackson. A lightweight formal analysis of a multicast key management scheme. *Formal Techniques for Networked and Distributed Systems-FORTE 2003*, pages 240–256, 2003.

[45] S.M.C. Téri. Using b method to formalize the java card runtime security policy for a common criteria evaluation.