Vers un cadre unificateur pour la spécification, formalisation et l'analyse d'exigences de sécurité

Hiba Hnaini

▶ **To cite this version:**

Hiba Hnaini. Vers un cadre unificateur pour la spécification, formalisation et l'analyse d'exigences de sécurité. Architectures Matérielles [cs.AR]. École Nationale Supérieure de Techniques Avancées Bretagne, 2024. Français. ⟨NNT : 2024ENTA0016⟩. ⟨tel-05315328⟩

HAL Id: tel-05315328

https://theses.hal.science/tel-05315328v1

Submitted on 15 Oct 2025

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

ENSTA BRETAGNE

THÈSE DE DOCTORAT DE

L'ÉCOLE NATIONALE SUPÉRIEURE
DE TECHNIQUES AVANCÉES BRETAGNE

ÉCOLE DOCTORALE N° 648
Sciences pour l'Ingénieur et le Numérique
Spécialité : *Sciences et technologies de l'Information et de la Communication*

Par
Hiba HNAINI

Vers un cadre unificateur pour la spécification, formalisation et l'analyse d'exigences de sécurité

Thèse présentée et soutenue à « ENSTA Bretagne (Brest) », le « 10 Decembre 2024 »
Unité de recherche : « Laboratoire Lab-STICC, UMR 6285 »

Rapporteurs avant soutenance :

Regine LALEAU — Professeur, Universite Paris-Est Creteil
Bénédicte LE-GRAND — Professeur, Université Panthéon Sorbonne

Composition du Jury :

Président :	Loïc LAGADEC	Professeur, ENSTA Bretagne *(à préciser après la soutenance)*
Examinateurs :	Christophe FELTUS	Chercheur HDR, LIST, Luxembourg
Dir. de thèse :	Raúl MAZO	Professeur, ENSTA Bretagne
Encadrant :	Paola VALLEJO	Maître de conférences, Universidad Eafit
Encadrant :	Joël CHAMPEAU	Maître de conférences, ENSTA Bretagne

Acknowledgement

I deeply thank my advisors, Raúl Mazo, Paola Vallejo, and Joël Champeau for their invaluable guidance, support, and encouragement throughout this research. Their expertise and insights have significantly shaped this thesis, and I am profoundly grateful for their mentorship.

I thank my committee members for their valuable feedback and suggestions, which enhanced this work.

I extend my heartfelt gratitude to my colleagues and friends in the Labsticc-SL, P4S, and VariaMos team for their intellectual discussions and camaraderie. Your support has made this journey both enjoyable and memorable.

I am profoundly grateful to my family for their unwavering love and encouragement. To my parents, who have consistently believed in me and inspired me to pursue my aspirations.

Table of Contents

Introduction en Français

Introduction Générale

La prévalence croissante des systèmes numériques rend essentiel le traitement des problèmes de sécurité dans les systèmes matériels et logiciels. La sécurité n'est plus considérée comme une question technique à traiter à la fin des processus de conception ou de mise en œuvre, mais plutôt comme une priorité à prendre en compte dès le début du cycle de conception. À mesure que notre dépendance aux systèmes numériques augmente, leurs architectures deviennent de plus en plus complexes, augmentant ainsi le potentiel de vulnérabilités et nécessitant une compréhension approfondie de la relation entre les composants matériels et logiciels. De plus, l'évolution de la technologie entraîne de nouvelles interactions, usages et exigences, rendant ces systèmes plus vulnérables aux nouvelles menaces. Presque chaque semaine, des nouvelles d'une cyberattaque sur les systèmes d'organisations privées ou publiques sont rapportées, démontrant le besoin de renforcer les composants matériels et logiciels contre les intrusions malveillantes. Les services publics stratégiques tels que l'énergie, la santé et les télécommunications, essentiels au fonctionnement de notre société, sont également à risque. Les conséquences de ces failles de sécurité sont vastes et variées. En plus de l'interruption immédiate du service, elles peuvent avoir un impact négatif sur la défense et la stabilité politique, affecter les paysages socio-économiques et entraîner des pertes financières significatives. De plus, les dommages ne sont pas seulement opérationnels; ils peuvent également nuire à l'image et à la réputation de l'organisation et éroder la confiance et la crédibilité du public.

Une manière très efficace et prometteuse de s'attaquer aux problèmes fondamentaux consiste à créer des systèmes avec des mesures de sécurité intégrées dès le départ [1]. Cette approche, connue sous le nom de "Sécurité par Conception" (Secure by Design), devient un modèle populaire pour le développement de systèmes, en se concentrant sur la sécurité et la confidentialité des systèmes d'information. Il est désormais essentiel de reconnaître son efficacité et de l'étendre, de la formaliser et d'en faire une pratique universelle. Dans l'approche Sécurité par Conception, la sécurité n'est pas une réflexion après coup mais une partie intégrante du système, en commençant par la création minutieuse de conceptions

architecturales sécurisées. Assurer la sécurité d'un système ne se limite pas à concevoir une architecture théoriquement sécurisée; il s'agit de maintenir cette architecture sécurisée tout au long du cycle de vie du système. Par conséquent, les ingénieurs et les architectes système doivent créer des systèmes qui couvrent tous les risques potentiels, de l'utilisation normale aux menaces internes et externes. Il est important de comprendre que l'époque où l'on pouvait prétendre à l'ignorance est révolue. Désormais, la sécurité est une exigence fondamentale pour tout système, mettant fin à l'époque où la sécurité était considérée comme une question de bricolage ou même une addition coûteuse et optionnelle ou une nécessité de dernière minute.

Énoncé du Problème

Le projet présenté dans cette thèse se concentre sur la conception et l'évaluation d'une approche permettant de spécifier, formaliser et analyser des systèmes sécurisés. L'objectif principal est de développer un cadre complet qui inclut les aspects de sécurité de ces architectures. À mesure que les systèmes modernes évoluent, ils font face à une gamme croissante de problématiques en matière de sécurité. Ces problématiques nécessitent une approche cohérente qui aborde la complexité des systèmes matériels et logiciels contemporains dès leur fondation.

Problématiques

1. **Complexité de la Conception de Systèmes Sécurisés :** Le processus de création de systèmes sécurisés est complexe et multifacette. Les méthodologies actuelles manquent d'un cadre unifié, ce qui rend difficile la prise en compte des préoccupations de sécurité de manière exhaustive. La multitude de langages disponibles pour spécifier et analyser les aspects de sécurité conduit souvent à des approches fragmentées, entravant le développement d'architectures sécurisées.
2. **Avancées Technologiques Rapides :** Avec l'avancement rapide de la technologie, les systèmes modernes sont constamment exposés à de nouvelles vulnérabilités. L'interaction complexe entre les composants matériels et logiciels, l'évolution des interactions utilisateur et les exigences de déploiement nécessitent des solutions innovantes. Les approches de sécurité existantes peinent à suivre ces avancées, ce qui rend essentiel le développement de méthodes disruptives capables de s'adapter

au paysage technologique en constante évolution.

3. **Applicabilité dans le Monde Réel :** Bien que des solutions de sécurité innovantes soient proposées et testées dans des environnements contrôlés, leur efficacité dans des scénarios de déploiement réel reste un enjeu. L'application pratique dans divers contextes est cruciale pour valider l'efficacité de ces approches. Le fossé entre l'exploration académique et la mise en œuvre pratique entrave l'intégration transparente des mesures de sécurité de pointe dans les systèmes réels.
4. **Manque de Modélisation de Sécurité Multiparadigme :** La modélisation de la sécurité nécessite une approche multiparadigme, prenant en compte divers aspects de l'architecture matérielle et logicielle. Les langages de modélisation actuels manquent souvent d'intégration, résultant en des représentations disjointes. Une vue multiparadigme unifiée est essentielle pour comprendre le paysage global de la sécurité d'un système, comblant le fossé entre les concepts théoriques et la mise en œuvre pratique.

Relever ces problématiques est fondamental pour développer une approche robuste, adaptable et pratique de la conception d'architectures matérielles et logicielles sécurisées. En surmontant ces obstacles, le projet vise à contribuer de manière significative à la cybersécurité, en assurant le développement de systèmes résilients face aux menaces de sécurité en constante évolution.

Questions de Recherche

Cette recherche entreprend une exploration complète des architectures matérielles et logicielles sécurisées. Son objectif central est de concevoir et d'évaluer une approche innovante pour leur spécification, formalisation et analyse.

La principale question de recherche abordée dans cette étude est ***RQ - Comment un cadre unifié peut-il être développé pour spécifier, formaliser et analyser des systèmes sécurisés ?*** Ce qui conduit à poser les sous-questions suivantes :

— **RQ1** - Comment spécifier des exigences de sécurité claires, non complexes et complètes pour les systèmes et les domaines ?
  - **RQ1.1** - Comment guider les ingénieurs en exigences de sécurité dans la spécification de chaque exigence de sécurité ?
  - **RQ1.2** - Comment guider les ingénieurs en exigences de sécurité dans la décision des exigences de sécurité à spécifier afin d'améliorer la couverture de

sécurité ?

— **RQ2** - Quelles sont les méthodes efficaces pour formaliser les exigences pour l'analyse de sécurité ?
— **RQ3** - Comment le modèle de sécurité peut-il être analysé en utilisant une métrique de sécurité quantitative objective et quels facteurs doivent être pris en compte ?
— **RQ4** - Comment mettre en œuvre le cadre pour une utilisation intuitive et conviviale ?

Hypothèses de Recherche

Les hypothèses présentées dans cette section constituent la base du cadre de recherche, décrivant le guide pour spécifier, formaliser et analyser les architectures matérielles et logicielles sécurisées. Ces hypothèses servent de propositions testables qui abordent divers aspects des objectifs de recherche. Le développement et la validation de ces hypothèses sont cruciaux pour évaluer l'efficacité et l'applicabilité du cadre unifié proposé et des méthodologies associées. Chaque hypothèse (Table 2) encapsule une affirmation de recherche spécifique, reflétant les résultats anticipés de cette étude.

Méthode de Recherche

La méthodologie de recherche adoptée pour développer le cadre englobant la spécification des exigences de sécurité, la formalisation à l'aide d'un langage de modélisation, et l'analyse de la sécurité basée sur les modèles est conçue pour assurer une progression systématique et une amélioration continue. Cette méthodologie comprend des étapes itératives et incrémentales, avec des processus rigoureux de test, de validation et d'amélioration. La méthodologie se déroule comme suit :

1. **Conceptualisation de Chaque Étape :** La recherche a commencé par une conceptualisation claire de chaque étape : spécification des exigences de sécurité, formalisation à l'aide d'un langage de modélisation, et analyse de sécurité des modèles. Des cadres détaillés et des méthodologies ont été décrits pour chaque étape, établissant la base pour le développement et les tests ultérieurs.
2. **Développement Itératif et Tests :** Chaque étape du cadre a été développée de manière itérative. Après la conception d'une étape, elle a été mise en œuvre,

Question de Recherche (RQ)	**Hypothèse**
RQ	**H:** L'intégration de méthodes existantes et nouvelles peut développer un cadre unifié pour la spécification, la formalisation et l'analyse des systèmes sécurisés.
RQ1.1	**H1.1 :** L'utilisation d'un langage naturel guidé peut significativement améliorer la clarté, la simplicité et l'exhaustivité de la spécification des exigences de sécurité pour les systèmes et les domaines. Cette approche pourrait potentiellement guider les ingénieurs dans la définition de spécifications de sécurité complètes en fournissant un format structuré et compréhensible pour exprimer des exigences de sécurité complexes.
RQ1.2	**H1.2 :** L'utilisation d'une représentation des connaissances, telle qu'une ontologie, peut fournir une approche structurée permettant aux ingénieurs de déterminer et de spécifier les exigences de sécurité nécessaires. Cela pourrait améliorer la couverture globale de la sécurité en fournissant une vue d'ensemble complète de tous les aspects de sécurité pertinents.
RQ2	**H2 :** La formalisation des exigences pour l'analyse de la sécurité peut être efficacement réalisée par une approche systématique qui incorpore des éléments clés tels que la modélisation des menaces, l'évaluation des risques et les contrôles de sécurité. Cette approche doit également tenir compte du contexte spécifique du système analysé.
RQ3	**H3 :** Le modèle de sécurité peut être analysé en utilisant une métrique de sécurité quantitative objective en considérant des facteurs tels que la gravité des potentielles violations et l'efficacité des mesures de sécurité. Cette analyse pourrait fournir une compréhension plus précise et complète de la sécurité du système.
RQ4	**H4 :** La mise en œuvre des trois parties du cadre à travers une application web pourrait améliorer son intuitivité et sa convivialité. Cela pourrait être réalisé en utilisant une interface utilisateur qui sépare et identifie chaque partie du cadre, en fournissant des instructions et des retours complets.

Table 1 – Questions de Recherche et Hypothèses Correspondantes

testée et affinée. Les retours de la phase de test ont été utilisés pour identifier les domaines à améliorer, assurant que chaque étape soit robuste et fonctionnelle avant de passer à l'étape suivante.

3. **Preuve de Concept (PoC) :** Une approche de preuve de concept (PoC) a été utilisée pour valider la viabilité et la faisabilité de chaque étape. Des expériences PoC ont été menées pour démontrer l'application pratique des composants du cadre. Les résultats des expériences PoC ont orienté les améliorations et ajustements ultérieurs pour renforcer l'efficacité du cadre.
4. **Expériences Contrôlées :** Des expériences contrôlées ont été conçues et réalisées pour évaluer systématiquement la performance et la précision de chaque étape. Une analyse comparative a été effectuée pour mesurer l'efficacité du cadre par rapport aux méthodes et outils existants. Les observations des expériences contrôlées ont été essentielles pour identifier les forces et faiblesses, conduisant à des améliorations ciblées.
5. **Cas Pratique :** Le cadre a été testé dans une étude de cas détaillée en conditions réelles. Cette application a permis une évaluation approfondie de l'utilisabilité, de l'adaptabilité et de l'efficacité du cadre dans une situation pratique. Les retours et observations de l'étude de cas ont été instrumentaux pour apporter des améliorations et ajustements supplémentaires au cadre, garantissant son alignement avec les besoins pratiques.
6. **Amélioration Continue et Versionnage :** Sur la base des retours des tests, des expériences PoC, des expériences contrôlées et de l'étude de cas, des améliorations itératives ont été apportées aux composants du cadre. Chaque cycle de test et d'évaluation a conduit à de nouvelles versions du cadre. Le versionnage et la documentation ont été maintenus de manière rigoureuse, permettant la traçabilité et la comparaison entre les itérations.

En employant cette méthodologie itérative et incrémentale, exprimée dans la Figure 2, la recherche a assuré l'évolution systématique du cadre. Les processus de test, de validation et d'amélioration continue, y compris les expériences PoC, les expériences contrôlées et un cas pratique, ont été cruciaux pour affiner les composants du cadre. La méthodologie a permis la création d'un cadre robuste, adaptable et pratique pour la spécification, la formalisation et l'analyse des exigences de sécurité, répondant aux défis posés par les paysages de sécurité en évolution et les scénarios d'application pratique.

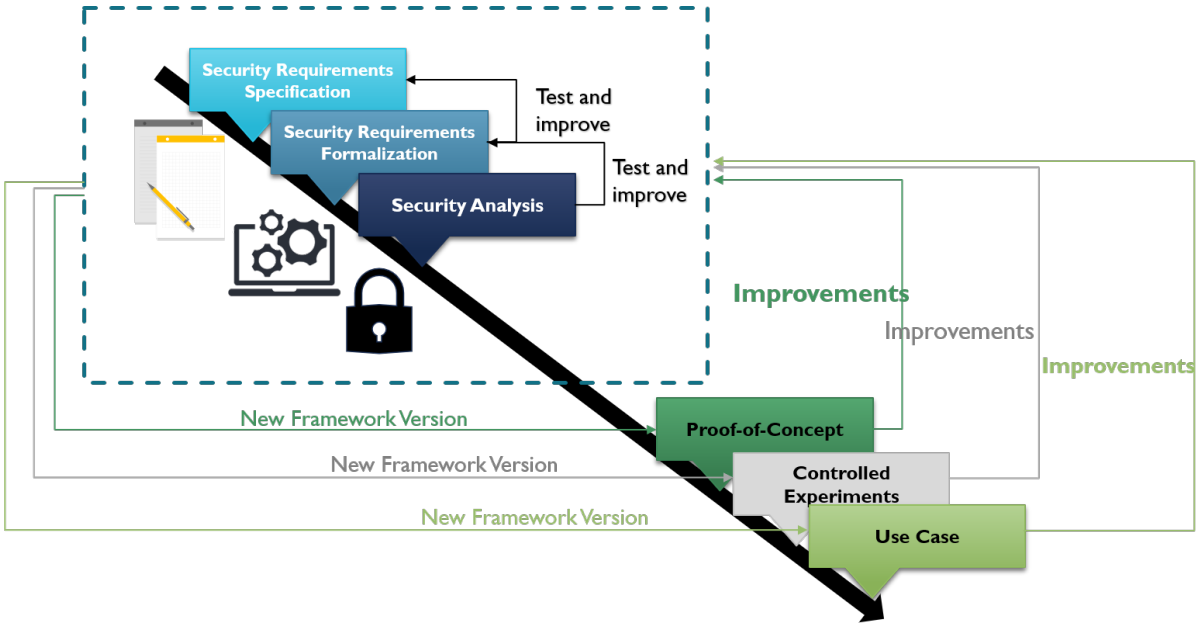


Figure 1 – Méthode de Recherche

Contributions

Cette étude de recherche apporte des contributions significatives à la cybersécurité en abordant les défis clés liés à la spécification, la formalisation et l'analyse des exigences de sécurité. Les contributions objectives suivantes ont été réalisées, améliorant ainsi la précision et l'efficacité des processus liés à la sécurité.

1. **Modèle semi-structuré pour les exigences de sécurité :**Cette recherche introduit un template semi-structuré pour les exigences de sécurité, fournissant aux ingénieurs des lignes directrices claires pour spécifier les exigences de sécurité. Le template décrit les différentes parties des exigences de sécurité, aidant les ingénieurs à obtenir des spécifications complètes et précises.
2. **Ontologie des critères de sécurité :** Le développement d'une ontologie des critères de sécurité est une contribution importante. Cette ontologie aide les ingénieurs à définir des mécanismes de sécurité basés sur des critères spécifiques au domaine et suggère des critères supplémentaires pour améliorer la couverture de sécurité du système. En associant cette ontologie au modèle, la recherche facilite l'identification des exigences de sécurité spécifiques au domaine et propose des exigences supplémentaires pour une couverture de sécurité complète.
3. **Nouveau langage de modélisation :** La recherche introduit un nouveau lan-

gage de modélisation incorporant des informations spécifiques au domaine et au contexte du système. Ce langage permet l'application de l'évaluation des risques et du traitement des risques. Il fournit une métrique qualitative pour l'analyse de la sécurité, en tenant compte du contexte spécifique du système analysé.

4. **Simulation et analyse efficaces :** La recherche présente des méthodologies pour transformer les modèles de sécurité en code de contraintes et utiliser des solveurs de contraintes pour la simulation et l'analyse avec une métrique qualitative de sécurité. De plus, elle définit des critères pour sélectionner des opérationnalisations appropriées en fonction des variables fournies par le contexte. Ces contributions garantissent que le processus d'analyse de la sécurité est rigoureux et adaptable à différents contextes, améliorant ainsi l'applicabilité des résultats de la recherche.
5. **Intégration avec VariaMos :** L'étude intègre avec succès le cadre développé dans VariaMos, une contribution significative à la communauté de modélisation des exigences. Cette intégration renforce les capacités de VariaMos, permettant aux ingénieurs de spécifier, de modéliser et d'analyser les exigences de sécurité dans un contexte de système plus large, améliorant ainsi la précision et la pertinence du processus d'analyse de la sécurité.

Organisation de la thèse

La thèse est organisée comme suit.

Chapitre 1 passe en revue la littérature sur la spécification, la formalisation et l'analyse des exigences de sécurité. Chaque section pose diverses questions de recherche visant à affiner l'étude de la spécification des exigences de sécurité, suivie de la formalisation, et se conclut par l'analyse. Ces questions seront méthodiquement abordées tout au long du chapitre.

Chapitre 2 décrit l'approche et le cadre, offrant des informations essentielles pour comprendre la thèse. Il comprend un exemple continu pour illustrer les étapes de l'approche.

Chapitre 3 décrit la phase initiale du cadre, en se concentrant sur la spécification des exigences de sécurité. Il démontre le modèle et l'ontologie utilisés dans cette

phase. Il clarifie ensuite leur utilisation combinée et développe enfin leur mise en œuvre.

Chapitre 4 introduit le langage formel pour les exigences de sécurité tel que discuté dans cette thèse, ainsi que la méthodologie d'analyse de la sécurité associée. Il élabore sur les modèles utilisés et les transformations ou fédérations appliquées entre eux. De plus, il décrit en détail la méthodologie d'analyse de la sécurité. Enfin, il couvre les aspects de mise en œuvre.

Chapitre 5 décrit les évaluations réalisées pour évaluer la méthode. Il expose les tests d'utilisabilité effectués et leurs résultats. De plus, il fournit une étude de cas approfondie du cadre complet.

Chapitre 6 conclut la thèse, tandis que le **Chapitre 7** présente les travaux futurs.

INTRODUCTION

General Introduction

The growing prevalence of digital systems has made addressing security issues in hardware and software systems essential. Security is no longer seen as a technical issue to be dealt with at the end of the design or implementation processes but rather as a priority that must be considered from the beginning of the design cycle. As our reliance on digital systems increases, their architectures become increasingly complex, increasing the potential for vulnerabilities and necessitating a thorough understanding of the relationship between hardware and software components. In addition, the evolution of technology brings new interactions, uses, and requirements, making these systems more vulnerable to new threats. Almost every week, news of a cyberattack on private or public organizations' systems is reported, demonstrating the need to strengthen hardware and software components against malicious intrusions. Strategic public services such as energy, healthcare, and telecommunications, which are essential for our society's functioning, are also at risk. The consequences of these security breaches are broad-reaching and varied. In addition to immediate service interruption, they can have a negative impact on defense and political stability, affect socioeconomic landscapes, and cause significant financial losses. Moreover, damage is not only operational; it can also damage the image and reputation of the organization and erode public trust and confidence.

A highly effective and promising way to address the core issues is to create systems with built-in security measures from the start [1]. This approach, known as "Secure by Design", is becoming a popular model for system development, focusing on the security and confidentiality of information systems. It is now essential to recognize its effectiveness and expand, formalize, and make it a universal practice. In the Secure by Design approach, security is not an afterthought but an integral part of the system, starting with carefully creating secure architectural designs. Ensuring the security of a system is more than just designing a theoretically secure architecture; it requires maintaining this secure architecture throughout the

system's life cycle. Therefore, engineers and system architects must create systems that cover all potential risks, from normal usage to internal and external threats. It is important to understand that the days of claiming ignorance are over. Security is now a fundamental requirement, no longer an optional or last-minute addition.

Problem Statement

The project reported in this thesis focuses on designing and evaluating an approach to specify, formalize, and analyze secure systems. The primary goal is to develop a comprehensive framework that includes the security aspects of these architectures. As modern systems evolve, they face an increasing range of security challenges. These challenges require a cohesive approach that addresses the complexity of contemporary hardware and software systems from the very foundation.

Challenges

(a) **Complexity of Secure System Design:** The process of creating secure systems is intricate and multifaceted. Current methodologies lack a unified framework, making it challenging to address security concerns in a comprehensive way. The multitude of languages available to specify and analyze security aspects often leads to fragmented approaches, hindering the development of secure architectures.

(b) **Rapid Technological Advancements:** With the rapid advancement of technology, modern systems are constantly exposed to new vulnerabilities. The intricate interplay of hardware and software components, evolving user interactions, and deployment requirements require innovative solutions. Existing security approaches struggle to keep up with these advancements, which makes it essential to develop disruptive methods that can adapt to the changing landscape of technology.

(c) **Real-World Applicability:** While innovative security solutions are proposed and tested in controlled environments, their effectiveness in real-world deployment scenarios remains challenging. Practical application in various settings is crucial to validate the efficacy of these approaches. The gap between academic

exploration and practical implementation hampers the seamless integration of cutting-edge security measures into actual systems.

(d) **Lack of Multiparadigm Security Modeling:** Security modeling requires a multiparadigm approach, considering various hardware and software architecture aspects. Current modeling languages often lack integration, resulting in disjointed representations. A unified multiparadigm view is essential to understanding the holistic security landscape of a system, bridging the gap between theoretical concepts and practical implementation.

Addressing these challenges is fundamental to developing a robust, adaptable, and practical approach to designing secure hardware and software architectures. By overcoming these hurdles, the project aims to contribute significantly to cybersecurity, ensuring the development of resilient systems in the face of evolving security threats.

Research Questions

This research embarks on a comprehensive exploration of secure hardware and software architectures. Its central objective is to design and evaluate an innovative approach to their specification, formalization, and analysis.

The main research question addressed in this study is ***RQ - How can a unified framework be developed to specification, formalize, and analyze secure systems?*** Which leads to posing the following sub-questions:

— **RQ1** - How to specify clear, non-complex, and comprehensive security requirements for systems and domains?
  - **RQ1.1** - How to guide security requirements engineers with the specification of each security requirement?
  - **RQ1.2** - How to guide security requirements engineers in deciding which security requirements to specify in order to improve security coverage?

— **RQ2** What are effective methods to formalize the requirements for security analysis?

— **RQ3** - How can the security model be analyzed using an objective quantitative security metric, and what factors should be considered?

— **RQ4** - How to implement the framework for intuitive and user-friendly use?

Research Hypotheses

The hypotheses presented in this section form the foundation of the research framework, outlining the guide for specifying, formalizing, and analyzing secure hardware and software architectures. These hypotheses serve as testable propositions that address various aspects of the research objectives. The development and validation of these hypotheses are crucial in evaluating the effectiveness and applicability of the proposed unified framework and associated methodologies. Each hypothesis (Table 2) encapsulates a specific research claim, reflecting the anticipated outcomes of this study.

Research Method

The research methodology adopted to develop the framework encompassing the specification of the security requirements, the formalization using a modeling language, and the security analysis based on the models is designed to ensure systematic progress and continuous improvement. This methodology comprises iterative and incremental steps, with rigorous testing, validation, and enhancement processes. The methodology unfolds as follows:

(a) **Conceptualization of Each Step:** The research initiated with a clear conceptualization of each step: specification of security requirements, formalization using a modeling language, and model security analysis. Detailed frameworks and methodologies were outlined for each step, establishing the foundation for subsequent development and testing.

(b) **Iterative Development and Testing:** Each framework step was iteratively developed. After the conception of a step, it was implemented, tested, and refined. Feedback from the testing phase was used to identify areas for improvement, ensuring that each step was robust and functional before proceeding to the next stage.

(c) **Proof of Concept (PoC):** A proof of concept (PoC) approach was used to validate the viability and feasibility of each step. PoC experiments were conducted to demonstrate the practical application of the framework components. The outcomes of PoC experiments guided further refinements and adjustments to enhance the framework's effectiveness.

Research Question (RQ)	**Hypothesis**
RQ	**H:** Integrating existing and new methods can develop a unified framework for the specification, formalization, and analysis of secure systems.
RQ1.1	**H1.1** Using guided natural language can significantly improve the clarity, simplicity, and completeness of the security requirements specification for systems and domains. This approach could potentially guide engineers in defining comprehensive security specifications by providing a structured and understandable format for expressing complex security requirements.
RQ1.2	**H1.2:** Using a knowledge representation, such as an ontology, can provide a structured approach for security requirements engineers to determine and specify the necessary security requirements. This could improve overall security coverage by providing a comprehensive overview of all relevant security aspects.
RQ2	**H2:** Formalizing the requirements for security analysis can be effectively achieved through a systematic approach that incorporates key elements such as threat modeling, risk assessment, and security controls. This approach should also consider the specific context of the system under analysis.
RQ3	**H3:** The security model can be analyzed using an objective quantitative security metric by considering factors such as the severity of potential breaches and the efficiency of the security measures. This analysis could provide a more accurate and comprehensive understanding of the system's security.
RQ4	**H4** Implementing the three parts of the framework through a web application could improve its intuitiveness and user-friendliness. This could be achieved using a user interface that separates and identifies each part of the framework, providing comprehensive instructions and feedback.

Table 2 – Research Questions and Corresponding Hypotheses

(d) **Controlled Experiments:** Controlled experiments were designed and carried out to systematically evaluate the performance and precision of each step. Comparative analysis was performed to measure the framework's efficiency against existing methods and tools. Insights from controlled experiments were instrumental in identifying strengths and weaknesses, leading to targeted improvements.

(e) **Practical Use Case:** The framework was tested in a detailed real-world case study. This application allowed for an in-depth evaluation of the framework's usability, adaptability, and effectiveness in a practical situation. The case study's feedback and observations were instrumental in further improvements and adjustments to the framework, guaranteeing its alignment with practical needs.

(f) **Continuous Improvement and Versioning:** Based on feedback from testing, PoC experiments, controlled experiments, and the case study, iterative improvements were made to the framework components. Each round of testing and evaluation led to new versions of the framework. Versioning and documentation were maintained meticulously, allowing traceability and comparison between iterations.

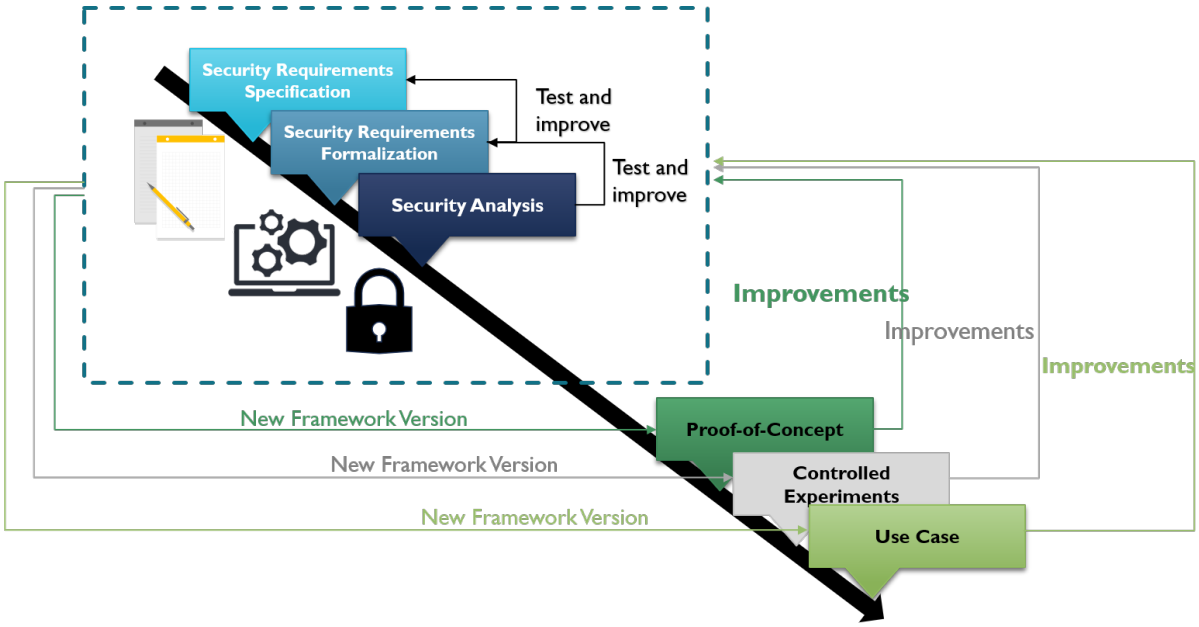


Figure 2 – Research Method

By employing this iterative and incremental methodology expressed in Figure

2, the research ensured the systematic evolution of the framework. Continuous testing, validation, and improvement processes, including PoC experiments, controlled experiments, and a use case, were instrumental in refining the framework components. The methodology allowed for the creation of a robust, adaptable, and practical framework for security requirements specification, formalization, and analysis, addressing the challenges posed by evolving security landscapes and practical application scenarios.

Contributions

This research study makes significant contributions to cybersecurity by addressing key challenges related to the specification, formalization, and analysis of security requirements. The following objective contributions have been achieved, enhancing the precision and effectiveness of security-related processes.

(a) **Semi-structured Template for Security Requirements:** This research contributes a semi-structured template for security requirements, providing engineers with a clear guideline for specifying security requirements. The schema delineates the various components of security requirements, helping engineers obtain comprehensive and precise specifications.

(b) **Ontology of Security Criteria:** Developing an ontology of security criteria is a significant contribution. This ontology assists engineers in defining security mechanisms based on domain-specific criteria and suggests additional criteria to enhance the security coverage of the system. By coupling this ontology with the template, the research facilitates the identification of domain-specific security requirements and suggests additional requirements for comprehensive security coverage.

(c) **New Modeling Language:** The research introduces a new modeling language incorporating domain-specific information and the system context. This language enables the application of risk assessment and treatment assessment. It provides a qualitative metric for security analysis, taking into account the specific context of the system under analysis.

(d) **Effective Simulation and Analysis:** The research presents methodologies for transforming security models into constraint code and employing constraint

solvers for simulation and analysis with a qualitative security metric. Additionally, it defines criteria for selecting suitable operationalizations based on context-provided variables. These contributions ensure that the security analysis process is rigorous and adaptable to different contexts, enhancing the applicability of the research findings.

(e) **Integration with VariaMos:** The study successfully integrates the developed framework into VariaMos, a significant contribution to the requirements modeling community. This integration enhances the capabilities of VariaMos, allowing engineers to specify, model, and analyze security requirements within a broader system context, thus improving the accuracy and relevance of the security analysis process.

Thesis Organization

The thesis is organized as follows.

Chapter 1 reviews the literature on security requirements specification, formalization, and analysis. Each section poses various research questions aimed at refining the study of security requirements specification, followed by formalization, and concluding with analysis. These questions will be methodically addressed throughout the chapter.

Chapter 2 outlines the approach and framework, offering essential information for understanding the thesis. It includes a continuous example to illustrate the steps of the approach.

Chapter 3 describes the initial phase of the framework, focusing on the specification of security requirements. It demonstrates the template and the ontology employed in this phase. It further clarifies their combined usage and ultimately elaborates on their implementation.

Chapter 4 introduces the formal language for security requirements as discussed in this thesis, along with the associated security analysis methodology. It elaborates on the utilized models and the transformations or federations applied among them.

Additionally, it describes the security analysis methodology in detail. Lastly, it covers the implementation aspects.

Chapter 5 describes the assessments performed to evaluate the method. It outlines the usability tests conducted and their outcomes. Additionally, it provides an in-depth use case of the complete framework.

Chapter 6 wraps up the thesis, while **Chapter 7** outlines the future work.

Chapter 1

State of The Art

This section summarizes the state of the art in three areas: existing methodologies for writing security requirements in natural language; existing approaches for designing systems using engineering languages; methodologies for identifying risks, threats, and vulnerabilities; and existing methodologies for performing system security analyses. Each subsection addresses specific research questions to refine our research. We applied automated and manual literature reviews using keywords extracted from the research questions to answer these questions.

1.1 Methodologies for Writing Security Requirements

Research Questions This part of the state-of-the-art aims mainly to answer the following four questions:

— ***Q1: Are there any existing solutions for reducing ambiguity, vagueness, and complexity when specifying requirements in natural language?***

— ***Q2: Do the existing solutions make it possible to define security aspects (security criteria and security mechanisms) in a requirement?***

— ***Q3: Can the existing solutions be used to define requirements for a family of systems?***

— ***Q4: Can the existing solutions be used to define requirements for self-adaptive systems?***

Since we aim to create a security framework by designing as many types of systems as possible, we have included product lines and self-adaptive systems, most often excluded from such studies.

The solutions considered in this section of the state of the art are presented below.

1.1.1 ACE (Attempto Controlled English)

The Attempto Controlled English (ACE) [2] schema was proposed by Fuchs and Schwitter in 1996 . ACE is a controlled natural language, a subset of standard English designed to serve as a knowledge representation language. As a language, ACE must be learned to be used competently. It is a formal language in which texts are computationally processable.

The language consists of vocabulary and grammar. The vocabulary comprises predefined function words, predefined fixed phrases, and content words. The grammar defines the syntax of the ACE language. ACE is accompanied by several tools and resources such as an APE (ACE parser), RACE (ACE reasoner), ACERules, ACEWiki, AceWiki-GF, etc. **The ACE language is not oriented toward security requirements and therefore does not take into account security criteria and mechanisms. It is possible to specify the requirements of a system family using Mass Common Noun Phrases. However, it does not consider self-adaptive systems.**

1.1.2 Gellish

In 2003, van Renssen introduced Gellish [3], another formalized natural language (or controlled natural language). The Gellish semantic modeling methodology is based on an integrated information architecture, which means that it covers the integration of a taxonomy and an ontology containing a definition of the formal Gellish language, dictionaries with language-specific terminology, knowledge about types of things, information about individual things, and information about individual things within their content.

Gellish offers several types of modeling, including product and process modeling, document modeling, requirement modeling, knowledge modeling, and dictionary modeling.

The Gellish language does not allow for the specification of security criteria and mechanisms. It is also not suitable for specifying the requirements of a system family and self-adaptive systems.

1.1.3 EARS (Easy Approach to Requirements Syntax)

In 2009, Mavin et al. introduced six schemas for requirement specification known as EARS [4]. These schemas aim to reduce or avoid ambiguity, imprecision, complexity, omission, duplication, verbosity, inappropriate implementation, and non-testability of requirement documents written in natural language. The six schemas are a generic requirement syntax that is then specialized into five types of requirements (omnipresent, event-driven, undesired behavior, state-driven, and other optional characteristics).

The EARS approach, with its six schemas, did not fully address the specification of security aspects. **By using the optional characteristic schema, it is possible to specify security requirements without concepts to address security aspects such as security criteria and security mechanisms. This approach only considers requirements directed towards a single system, which excludes the possibility of specifying requirements for a system family using its models or schemas. The same applies to the requirements of self-adaptive systems, which were not considered in these schemas.**

We evaluated the models using a case study as a proof of concepts and found them insufficient for correctly writing (i.e., reducing complexity and ambiguity) security requirements. For example, a security requirement specified using this schema would be: "The <mobile phone> $_{SystemName}$ shall <ensure data confidentiality through radio channel encryption algorithms> $_{SystemResponse}$."The schema, in this case, can identify the system name and system response without any further level of detail.

1.1.4 Kamalrudin et al.'s Template

Kamalrudin et al. (2017) proposed a schema to specify security requirements [5]. This approach offers a library of security requirements and a schema to reduce the complexity and ambiguity of defining requirements in natural language and helping requirements engineers specify security requirements.

Unlike the EARS approach, this methodology considers security aspects (security criteria and mechanisms). The security requirements library includes five security criteria: authentication, integrity, confidentiality, availability, and non-repudiation. **However, these security criteria are not directly expressed**

in the security requirements schema proposed in this approach. They are deduced from the security keywords and security mechanisms defined in the schema and library. Similarly to the EARS approach, this approach does not address the requirements specification for a system family or self-adaptive systems.

After evaluating the library and schema using a proof-of-concept case, it was found that the library and schema were insufficient to express all the security requirements required by a system. This limitation arises because the library encompasses only three security criteria: confidentiality, integrity, and authentication. For example, it is not possible to express the following requirement concerning privacy: "The <mobile phone> $_{subject}$ <must ensure> $_{verb}$ <user privacy> $_{object}$ <by deleting user data when the bootloader is unlocked.>" (no way to express this requirement). The schema also does not take into account conditionality.

1.1.5 Mazo et al.'s Template

Mazo *et al.* (2020) introduced a requirement specification template [6] inspired by Rupp's template [7] and the Relax language [8]. This template is structured into nine sections: (1) Conditions under which a behavior occurs, (2) family of systems, systems, or parts of a system, (3) degree of priority or legal obligation, (4) activity, (5) object or objects, (6) additional details, (7) conditionality in the object, (8) verification criterion (adjustment) of the requirement, and (9) Relax requirement statements for self-adaptive systems.

This template does not cover security aspects such as security criteria and mechanisms. In contrast to the approaches previously discussed, this template allows for the specification of requirements for families of systems (or product lines) and self-adaptive systems. This flexibility is provided in the second section of the template (family of systems, systems, or parts of a system) and the ninth section (relax requirement statements).

We evaluated the template using the same case study as a proof-of-concept that we used to assess the two previous templates. Our findings indicate that Mazo et al.'s template significantly reduces complexity and ambiguity compared to prior approaches, as it provides more structured guidance for requirement formulation.

However, security criteria and mechanisms are treated as additional information and are not detailed in the template.

1.1.6 Cube

Pabuccu et al. introduced the Cube requirement writing template [9] specifically tailored for software systems, using the 5W1H questions. The template comprises the following components:

— **Why:** Specifies the purpose of the requirement.
— **Who:** Identifies the actor associated with the requirement.
— **Where:** Indicates the location or context of the requirement.
— **When-trigger:** Specifies the precondition or event that triggers the requirement.
— **When-condition:** Describes the postcondition or state resulting from the requirement.
— **What:** Defines the action or activity required by the requirement.
— **How:** Explains the approach or method for developing the system to meet the requirement.

Cube categorizes requirements into three types and offers distinct templates for each:

— **Business Requirement:** *WHO - WHAT - WHERE - WHEN TRIGGER - WHEN CONDITION - Aim, Reason: WHY.*
— **User Requirement:** - *when WHO - WHEN TRIGGER - WHERE - What - if - WHEN CONDITION - HOW (optional) - WHY (optional).*
— **Functional Non-functional Requirement:** *Who - What - when - When Trigger - and When Condition - Where - WHY (optional) - HOW (optional).*

The Cube requirement templates, while robust in addressing various aspects of software system development, do not include specific provisions for handling security concerns, specifications tailored for a family of systems, or requirements for self-adaptive systems. This omission implies that users of the Cube templates may face challenges in effectively addressing these important aspects.

1.1.7 Esser and Struss's Template

Esser and Struss introduced a template-based approach [10] aimed at facilitating the acquisition of requirements for functional testing of control software in passenger vehicles. Their methodology utilizes natural language templates to capture the required specifications, which can then be translated into propositional logic and temporal relationships to form a model of expected behavior.

The structure of their template follows an "if (start-condition) - then (consequence) - until (end-condition)" sentence format. For example, a requirement might be expressed: "if the system is in mode m1, lamp L3 is off, and button B4 is released, then immediately lamp L3 is lit until button B4 is pressed again or the system exits mode m1."

While Esser and Struss's approach primarily focuses on generating test cases from functional specifications, it offers limited support for addressing security aspects, family of systems, and auto-adaptive systems.

1.1.8 ReSA

Mahmud et al. propose a toolchain for the specification of structured requirements in the ReSA language [11]. ReSA is an ontology-based requirement specification language tailored for developing automotive embedded systems. It encompasses natural language terms (words and phrases), syntax, and an ontology that defines the concepts and syntactic rules of the specification. Additionally, it employs requirements boilerplates to structure the specification, offering six boilerplates or templates:

— **Simple:** A statement that contains a modal verb, such as *shall* (e.g., "system shall be activated").
— **Proposition:** A proposition or assertive statement (e.g., "button is pressed").
— **Complex:** Constructed from a Simple or Proposition boilerplate and an adverbial conjunctive (e.g., "the error shall be reported while the fault is present").
— **Compound:** Comprising two or more Simple or Proposition boilerplates and logical operators, *AND/OR* (e.g., "system shall be activated and the driver shall be notified").
— **Conditional:** A different variant of conditional statements, *i.e., if, if-else, if-elseif*, or *if-elseif-else*, and conditional nesting.

— **Prepositional Phrase:** Used to describe timing properties, the occurrence of events, and other complements to the subject of a main phrase (e.g., “within 5ms, by the driver").

Although ReSA provides structured support for requirements specification, it has limitations in addressing security aspects, family of systems considerations, and auto-adaptive systems.

1.1.9 Firesmith Template

Firesmith explores the importance of reusable parameterized templates to define security requirements [12]. He proposes an asset-based risk-driven analysis approach to determine the relevant parameters for reusing these templates, aiming to enhance the quality of security requirements in the specifications. Firesmith offers guidance on template creation and illustrates with an example specifying integrity:

“The [application/component/data center/business unit] must safeguard the [identifier|type] data it transmits from corruption (e.g., unauthorized addition, modification, deletion, or replay) caused by [unsophisticated/somewhat sophisticated/sophisticated] attacks during the execution of [a set of interactions/use cases] as indicated in [specified table]".

However, accessing detailed information about these templates in databases can be challenging, thus limiting their usability [13] [14] [15].

1.1.10 AMAN-DA

Souag et al. propose a method for eliciting and analyzing security requirements [16]. This approach utilizes a multi-level domain ontology of security concepts. It offers a structured template for expressing security requirements: *<When> <Under what condition> <Agent name> "Shall/Should/Will” <Action> <Assets> <Additional Information>* (e.g., “The web publishing system should lock accounts after reaching the login failure threshold"). **While AMAN-DA offers organized assistance for specifying security requirements, it has shortcomings when it comes to dealing with considerations related to families of systems and auto-adaptive systems.**

1.2 Methodologies for Formalizing Security Requirements

Once the requirements are defined, formalization becomes essential for a security assessment. We have recognized modeling languages that fall into three main categories: Methodologies for Designing Secure Systems, Methodologies for Identifying Threats, and Approaches for Identifying Attack Surfaces. These languages offer organized methods for recording security-related details, allowing for a methodical examination and assessment of system security. They are vital in guaranteeing comprehensive security inclusivity and aiding in well-informed decision-making at every stage of the system development process.

1.2.1 Methodologies for Designing Secure Systems

We have identified two objectives related to modeling languages for secure systems in our project:

(a) Identify modeling languages for secure systems' hardware and software architectures.

(b) Identify the tools that implement these languages and identify their strengths and weaknesses.

Research Questions The approaches found in our literature review are UMLSec, SysML-Sec, SecureUML, Secure Tropos, and Secure i*. Each approach is analyzed in light of the following five questions.

— ***Q1. Are the existing solutions and their tools up-to-date and still usable?***

— ***Q2. Do the existing solutions allow modeling security aspects (security criteria and security mechanisms)?***

— ***Q3. Do the existing solutions allow modeling requirements for a family of systems?***

— ***Q4. Do the existing solutions allow modeling requirements for self-adaptive systems?***

UMLSec UMLSec is an extension of UML, created by Jan Jurjens [17], which allows the expression of security-related information in system specification diagrams. This extension is defined as a UML profile. To use the UMLSec profile, the CARISMA plugin must be installed in an Eclipse modeling environment. The plugin is available but outdated and only usable with UML1. Other tools like ArgoUML exist, but they are no longer available online.

UMLSec considers three security criteria (confidentiality, integrity, and access control) but does not allow representing associated security mechanisms. Additionally, the UMLSec profile does not address modeling security requirements for system families, product lines, or self-adaptive systems.

However, an advantage of the UMLSec profile is that the UML metamodel is available, which facilitates language modification or extension.

SecureUML In the same year, 2002, Lodderstedt et al. proposed SecureUML [18], a modeling language for model-driven development of secure distributed systems based on UML. It is based on role-based access control (RBAC) with additional support to specify authorization constraints. It should be used with the SecureMova tool, which is only available on Linux.

As it is a role-based access control mechanism (RBAC), it allows the representation of two security criteria: access control and authorization. However, it only models the access control security mechanism. Like UMLSec, SecureUML does not enable modeling security requirements for system families or self-adaptive systems.

SI* In 2003, Liu et al. proposed a language for security and privacy analysis using i* [19], an agent-oriented requirements modeling language. No tools were found available online, as all projects have been abandoned. However, the graphical language can be used without a tool.

The language allows the representation of security criteria and security mechanisms without naming them explicitly. Security criteria can be represented as soft goals, and security mechanisms can be represented as tasks. However, like i*, SI* does not allow modeling security requirements for system families or self-adaptive systems. Additionally,

since SI* is a graphical language with meta-models available online, it would be easy to extend this language as well.

Secure Tropos In 2007, Mouratidis et al. introduced the Secure Tropos language as an extension of Tropos [20], an agent-oriented software methodology. Like SI*, Secure Tropos tools are no longer available online, as all projects are abandoned, but the graphical language can be used without the tool. **The Secure Tropos methodology enables the representation of security criteria and security mechanisms through the concepts of Security Feature and Security Mechanism, respectively. In 2014, Mellado et al. proposed SecureTropos-SPL, an extension of Secure Tropos to support security requirements engineering for software product lines based on security goals and risk-driven. However, neither Secure Tropos nor its extension allows for representing security requirements for self-adaptive systems.**

SysML-Sec In 2013, Apvrille and Roudier proposed the SysML-Sec environment to design safe and secure embedded systems with an extended version of the SysML language [21], targeting both the software and the hardware components of these systems. SysML-Sec supports the following methodological steps: (1) requirement capture (specification), (2) functional view, (3) architecture, (4) attack graph, and (5) mechanism design. It is supported by Ttool, which is available and up-to-date.

SysML-Sec allows the specification of eight security criteria (privacy, confidentiality, access control, freshness, nonrepudiation, availability, immunity, integrity) in the requirements capture step. It also allows for the specification of the security mechanism in the functional view and in the mechanism design steps. However, it does not consider product lines or self-adaptive systems.

Other Languages and Methodologies In 2013, MoDELO (Model-Driven Security Policy) was proposed as an extension of UMLSec. Specifically, MoDELO is an MDE-based approach that allows designers to define and evaluate OrBAC (Organization-based access control) security policies [22]. The tool is unavailable online, and it is impossible to use the language without this tool.

In 2017, Hachem et al. proposed SoSSec, a model-driven security methodology

to model and analyze architectures of secure systems of systems (SoSSec) [23]. The language implementation tool is not available online, making it impossible to use the language without this tool.

1.2.2 Methodologies for Identifying Risks, Threats, and Vulnerabilities

We have identified three objectives related to the security analysis aspect of systems.

1. Identify security risks (e.g., Privacy loss, Reputation damage) in the hardware and software architectures of systems. 2. Identify security threats (for example, denial of service, malware) in the hardware and software architectures of the systems. 3. Identify vulnerabilities (e.g., attack surfaces) in systems' hardware and software architectures.

Research Questions

The approaches found in each category (risk identification, threat identification, and vulnerability identification) are analyzed in light of the following five questions:

— ***Q1. Is there a solution to identify (automatically or manually) and possibly represent the risks, threats, and vulnerabilities of systems?***
— ***Q2. Are the existing solutions and their tools up-to-date and still usable?***
— ***Q3. What other aspects of security do the existing solutions allow for identifying and representing?***
— ***Q4. Do the existing solutions allow identifying and representing security aspects related to system families?***
— ***Q5. Do the existing solutions allow for the identification and representation of security aspects related to self-adaptive systems?***

Approaches for Threat Identification

The approaches found in our study of the literature are: STRIDE, OCTAVE, TRIKE, DML, and CORAS.

STRIDE The STRIDE approach was introduced in 1999 at Microsoft by Kohnfelder and Garg to help developers find threats in the company's software products.

In 2017, Khan et al. proposed a threat modeling methodology based on STRIDE for cyber-physical systems [24].

STRIDE is an acronym for the following threat types: "Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of privilege." Each threat type is associated with a security criterion as follows: "Spoofing - Authentication, Tampering - Integrity, Repudiation - Nonrepudiation, Information disclosure - Confidentiality, Denial of service - Availability, and Elevation of privilege - Authorization."

Microsoft's threat modeling tool (Microsoft Threat Modeling Tool) is used to apply the STRIDE methodology. It applies a specific threat modeling approach called STRIDE by interaction.

However, the methodology does not allow for the specification of security mechanisms and does not consider product lines and self-adaptive systems.

CORAS In 2002, Fredriksen et al. proposed CORAS [25], a security risk analysis method that provides a custom language to model threats and risks based on UML. The language consists of five different types of diagrams: resource diagrams, threat diagrams, risk diagrams, processing diagrams, and overview diagrams. The method is accompanied by the CORAS tool, available online.

CORAS provides a simple graphical representation of security requirements or threats for security analysis. The method allows for representing security mechanisms as treatments but does not consider security criteria. CORAS also does not allow modeling threats for a family of systems or self-adaptive systems.

OCTAVE In 2003, Alberts et al. presented OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation) [26], a threat modeling method based on risk assessment. OCTAVE methodology does not have a dedicated tool. It consists of 8 steps: (1) Establishing risk measurement criteria, (2) Developing an information asset profile, (3) Identifying informational asset containers, (4) Identifying areas of concern, (5) Identifying threat scenarios, (6) Identifying risks, (7) Analyzing risks, and (8) Choosing a mitigation approach.

This methodology covers all the parts necessary for a security requirement (template), including security criteria and security mechanisms. However, it is not suitable for use for self-adaptive systems and product lines.

TRIKE After STRIDE, OCTAVE, and CORAS, the TRIKE [27] threat modeling methodology was introduced in 2005 by Eddington et al. It is a unified conceptual framework for security audit from a risk management perspective by generating threat models. It consists of four models: (1) Requirement Model, (2) Implementation Model, (3) Threat Model, and (4) Risk Model.

The process begins with defining the system's requirements, then how the system is implemented, then threats are generated, and the process is completed with the definition of risk estimation and mitigation.

The tool provided for TRIKE is an Excel sheet. This methodology does not allow one to represent security criteria and security mechanisms. It also does not allow risk analysis for a family of systems or self-adaptive systems.

DML Then, in 2014, Stillions proposed the DML (Detection Maturity Level) [28] threat modeling approach. The DML model is a capability maturity model to reference an individual's maturity in cyberattack detection. DML defines the relationships between intelligence, detection, and response missions. It consists of nine maturity levels (0-8), with the lower levels being the most technical-specific and the higher levels being the most abstract on the technical level. **Using DML does not require a tool. However, it turns out that it does not adequately express the risk analysis required by our framework.**

Approaches for Identifying Attack Surfaces

The approaches found in our literature study are: Manadhata et al.'s Formal Model, SysML Activity Diagram to Detect Attack Surface, and PIMCA.

Manadhata et al.'s Formal Model Manadhata et al. proposed a formal model [29] in 2007 to measure the attack surface of a software system. The system's

attack surface is formalized using a system's input/output automaton model and by defining a quantitative measure of the attack surface in terms of three resources that can be exploited to attack the system: methods, channels, and data.

After identifying the attack surface, the potential damage and attack effort are estimated for each resource based on its attributes (e.g., method's probability and access rights), leading to the measurement of the attack surface.

This methodology does not allow the identification of the security criteria and security mechanisms related to or included in the attack surface. It also does not allow one to identify the attack surface of a family of systems or self-adaptive systems.

SysML Activity Diagram to Detect Attack Surface In 2014, Ouchani and Lenzini proposed a framework to detect the attack surface of a system using a SysML activity diagram [30]. They formalized activity diagrams by creating a library of attack patterns. These patterns include identity spoofing, data leakage, and resource exhaustion. Each attack pattern is also associated with the likelihood or probability of its occurrence.

This methodology does not allow identifying the security mechanism and security criteria related to the attack surface or vulnerabilities. It also does not consider system families and self-adaptive systems.

PIMCA PIMCA, proposed by Drouot and Champeau in 2019 and Sun et al. in 2020 [31] is a language developed in collaboration with the DGA to graphically represent the hardware and software architecture of systems to analyze their attack surfaces. The Pimca workshop has been implemented using the Openflexo tool (openflexo.org).

PIMCA does not include notions of security criteria. It includes security mechanisms in an abstract manner (customs). However, it does not allow for specifying or identifying the attack surfaces of system families or self-adaptive systems.

1.3 Methodologies for Security Analysis

Research Questions

The literature review conducted on the formal specification of security requirements aims to address the following questions:

— ***Q1. Are there existing solutions to analyze the security of a formal system model?***
— ***Q2. Are the existing solutions and tools (if necessary) up-to-date and usable?***
— ***Q3. Do the existing solutions allow for security analysis targeting different security criteria (and mechanisms)?***
— ***Q4. Do the existing solutions allow for security analysis of a family of systems?***
— ***Q5. Do the existing solutions allow for the security analysis of self-adaptive systems?***

Approaches Considered

The approaches considered in this section of the state-of-the-art are the following: Dolev-Yao, Dagger, and MTD.

1.3.1 Dolev-Yao

The Dolev-Yao formal model, proposed by Dolev and Yao in 1981 [32], is a threat model used to evaluate the security of encryption protocols. A system or protocol model must first be created to use the threat model. The threat model is then applied to the encryption protocol model, resulting in protocol analysis.

One of the tools that implements the Dolev-Yao threat model is the ProVerif tool. It is available and up-to-date. SysML-Sec also uses this tool to verify encryption protocols. This analysis method can be used but is not sufficient, as it only considers the confidentiality criterion and the encryption mechanism. It does not apply to family of systems and self-adaptive systems.

1.3.2 MTD (Moving Target Defense)

MTD is another security analysis approach proposed in 2011 by Jajodia et al. [33]. It allows for creating, analyzing, evaluating, and deploying diversified mechanisms and strategies that evolve and change over time to increase complexity and

cost for attackers, limit exposure to vulnerabilities and attack opportunities, and increase system resilience. There are two types of MTD theory: (1) MTD based on the attack surface and (2) MTD based on the attack graph. The first involves moving the system's attack surface to analyze network system security. The second uses a network model, an attack graph, and a dependency graph.

There are no known tools for the MTD approach and no clear documentation or tutorial, so we were unable to use it and, therefore, unable to answer our research questions

1.3.3 NIST SP 800-30 Risk Assessment Method

The initial release of NIST Special Publication 800-30, which serves as a guide to conducting risk assessments, occurred in September 2012. The NIST SP 800-30 Risk Assessment methodology prescribes a systematic procedure to assist organizations in performing detailed risk analysis for their operations [34]. This methodology encompasses several essential steps:

(a) **Preparation for NIST 800-30 Assessment**: Define the objectives and scope of the assessment and determine the locations for creating, transmitting, and storing sensitive data.

(b) **Threat Sources and Events**: Recognize the types of threat sources that confront the organization (e.g., adversarial, accidental) and the potential incidents they might initiate (e.g., phishing, power outage).

(c) **Vulnerabilities and Predisposing Conditions**: Pinpoint vulnerabilities in information systems or environments and consider predisposing conditions during risk assessment.

(d) **Assessing Likelihood of Occurrence**: Using various levels to gauge the probability that threat events occur and result in negative consequences.

(e) **Assessing Impact Magnitude**: Evaluate the consequences of threat events using a tiered approach.

(f) **Risk Evaluation**: Integrate likelihood and impact to determine the organization's risk level.

(g) **Risk Response Communication**: Ensure pertinent organization members are informed about risk-related details to aid decision-making.

(h) **Ongoing Assessment Maintenance**: Persistently observe the risk elements identified in the assessment and review them as threats, vulnerabilities, and risks develop.

While this method does not use a specific tool, it provides a thorough framework and methodology for organizations to evaluate the risks associated with their information systems. It excludes the concepts of security criteria and security mechanisms and is unsuitable for family and auto-adaptive systems.

1.3.4 Dagger

Another security analysis methodology is Dagger, proposed by Peterson in 2016 [35]. Dagger is a semi-automated modeling and visualization framework that addresses the representation of knowledge and information for decision-makers, allowing them to better understand the operational context of network security data.

It uses a system model, dependencies, and a state model to perform security analysis through virtualization. Similarly to MTD, there are no known tools or sufficient documentation to evaluate Dagger against our research questions.

1.3.5 Threat and Hazard Identification and Risk Assessment (THIRA)

Introduced on September 14, 2021 , the Threat and Hazard Identification and Risk Assessment (THIRA) [36] outlines a structured method to aid communities in identifying potential threats and hazards, evaluating their impacts, and determining necessary resources for effective risk management [37][38][39]. The procedure involves three primary steps: pinpointing potential threats and hazards to the community, evaluating the potential impacts of these threats, and securing the resources required for long-term risk management [40]. THIRA's main goal is to enhance community preparedness by offering detailed risk information, aiding in the decision-making process for risk mitigation strategies, and identifying areas where capabilities need enhancement to bolster resilience against various threats and hazards [41]. **No universally mandated specific tool for implementing**

the THIRA process exists. While the approach includes certain security criteria and mechanisms, these are not the main focus. Additionally, the methodology primarily assesses risks at the community level rather than addressing complex or adaptive systems in detail.

1.4 Conclusion

This chapter thoroughly investigates current methods and strategies in security engineering, emphasizing the importance of considering various aspects such as usability, compliance with a range of security standards and mechanisms, support for system families and adaptive systems, and the diverse nature of security modeling paradigms. The tables offered a detailed overview of various methodologies and templates pertinent to security requirements, system design, threat identification, security vulnerability analysis, and security analysis. Each table explores different aspects of these methodologies, covering their usability, integration of security criteria and mechanisms, suitability for system families and adaptive systems, and their support for multiple paradigm approaches. In Table 1.1, various templates are evaluated based on their ability to minimize ambiguity and address security criteria, mechanisms, system families, and auto-adaptive systems. Similarly, Table 1.2 evaluates methodologies such as UMLSec and SecureUML in relation to their usability, the incorporation of security criteria and mechanisms, and the support for family systems and auto-adaptive systems. Table 1.3 sheds light on methodologies such as STRIDE and OCTAVE, focusing on their risk analysis capabilities, usability, integration of security criteria and mechanisms, and support for system families, auto-adaptive systems, and multi-paradigm approaches. Likewise, Table 1.4 evaluates methodologies such as the formal Model of Manadhata et al. and PIMCA based on their capability to pinpoint attack surfaces, usability, inclusion of security criteria, and mechanisms, support for system families, auto-adaptive systems, and multiparadigm approaches. Finally, Table 1.5 examines approaches like Dolev-Yao and MTD in terms of their capabilities for security analysis, ease of use, inclusion of security criteria and mechanisms, and compatibility with system families and self-adaptive systems.

Based on the results from the presented tables, the approaches that align best with our research questions and can be utilized or expanded to develop our frame-

Table 1.1 – Summary of Requirements Templates

Template	**(Q1) Reduces Ambiguity**	**(Q2) Security Criteria**	**(Q3) Security Mechanisms**	**(Q4) System Family**	**(Q5) Auto-Adaptive Systems**
ACE	Yes			Yes	
Gellish	Yes				
EARS	Yes				
Kamalrudin et al. Schema	Yes		Yes		
Mazo et al. Schema	Yes			Yes	Yes
Cube	Yes				
Esser and Struss's Template	Yes				
ReSA	Yes				
Firesmith Template	Yes				

Table 1.2 – Summary of Methodologies for Secure Systems Design

Methodology	**(Q1) Usable Languages**	**(Q2) Security Criteria or Security Mechanisms**	**(Q3) System Family**	**(Q4) Auto-Adaptive Systems**
UMLSec	Yes			
SecureUML	Yes	Yes		
SI*	Yes	Yes		
Secure Tropos	Yes	Yes		
SysML-Sec	Yes	Yes		

Table 1.3 – Summary of Threat Identification Methodologies

Methodology	**(Q1) Risk Analysis**	**(Q2) Usability**	**(Q3) Security Criteria or Security Mechanisms**	**(Q4) System Family**	**(Q5) Auto-adaptive systems**
STRIDE	Yes	Yes	Yes		
OCTAVE	Yes	Yes	Yes		
CORAS	Yes	Yes	Yes		
TRIKE	Yes	Yes			
DML	Yes	Yes			

Table 1.4 – Approaches for Identifying Security Vulnerabilities

Methodology	**(Q1) Attack Surface Identification**	**(Q2) Usability**	**(Q3) Security Criteria or Security Mechanisms**	**(Q4) System Family**	**(Q5) Auto-Adaptive Systems**
Formal Model of Manadhata et al.	Yes				
SysML Activity Diagram	Yes				
PIMCA	Yes		Yes		

Table 1.5 – Summary of Security Analysis Methodologies

Methodology	**(Q1) Security Analysis**	**(Q2) Usability**	**(Q3) Security Criteria & Mechanisms**	**(Q4) System Family**	**(Q5) Self-Adaptive Systems**
Dolev-Yao	Yes	Yes	Yes		
MTD	Yes				
Dagger	Yes				
NIST	Yes				
THIRA	Yes				Yes

work are: the Mazo et al. template for requirements specification, Secure Tropos, SI*, and CORAS for requirements formalization, and NIST and THIRA for security analysis. The Mazo et al. template was deemed appropriate for our approach due to its focus on families of systems and auto-adaptive systems, and it can be extended to include security aspects. For security requirements formalization, SI* and Secure Tropos were identified as the two modeling languages that incorporate the security aspects intended by our approach. Additionally, the various diagrams of the CORAS modeling language were found to be beneficial in simplifying the risk assessment process. CORAS was chosen because it provides structured diagrams that simplify risk assessment and enhance the communication and traceability of results. While robust, STRIDE and OCTAVE do not offer the same level of graphical modeling or the ease of analysis as CORAS. For security analysis, NIST was considered suitable for risk assessment as it provides an objective score. Furthermore, THIRA was found to consider the system's context, which aligns with our goals. This contextual approach is particularly valuable for self-adaptive systems, where operational conditions can change significantly. Conversely, the Dolev-Yao model is primarily designed for analyzing security protocols, focusing on communication and information exchanges in potentially hostile environments. While powerful for modeling attacks on cryptographic protocols, it does not account for broader threat scenarios or contextual risk analysis like THIRA. These methodologies and languages were adapted, extended, or used as inspiration for our framework.

Chapter 2

Approach Definition

This chapter provides an overview of the framework proposed in this thesis. The framework presents a systematic approach to the specification, formalization, and analysis of secure architectures in the context of system engineering. Divided into three key components, the framework guides engineers through the process of integrating security measures into the design of a system from the early stages of development. The framework, as shown in Figure 2.1, consists of specifying requirements for a domain or a product line, specifying the requirements of an application through configuration, modeling the system, and finally applying security analysis to the model. *However, the system configuration step has not been addressed in this thesis.* This approach was achieved by applying the research method explained in the introduction.

Figure 2.1 – Framework Overview

Specification of Security Requirements The first phase focuses on defining security requirements for domains and applications, as in steps 1 and 3 of Figure 2.1, in a clear and detailed manner. Two major contributions, SECRET (SECurity REquirements specification Template) and SCORE (Security Criteria Ontology for Security Requirements Engineering), provide the foundation for this process. *SECRET:* A template of security requirements designed to decompose security requirements into distinct parts, facilitating the specification process. *SCORE:* An ontology of security aimed at assisting engineers in defining security mechanisms specific to domain-specific security criteria. Integrating SECRET and SCORE into

VariaMos, creating a module specifying functional, non-functional, and security requirements tailored to various domains and applications. This integrated module enables engineers to harmoniously define security requirements, leveraging the detailed structure of SECRET and the knowledge richness of SCORE. This improves the quality and relevance of security specifications in system development.

Formalization of Requirements In the formalization phase, the textual security requirements defined in the specification module are converted to the SERENA (SEcurity REquirements aNAlysis) modeling and analysis language, as in step 4 of Figure 2.1. The detailed SECRET model forms the basis for formally expressing textual requirements, while the inclusion of security criteria defined in the SCORE ontology enriches the modeling process with specific details on associated security mechanisms. Formalization ensures consistency between textual specifications and formal models, facilitating verification and validation of security properties. This approach facilitates a smooth transition from textual security specifications to more formal representations, enhancing traceability and system comprehension. It also promotes the seamless integration of security throughout the project lifecycle, fostering the development of robust and resilient systems.

Simulation and Analysis of Models The final phase, step 5 of Figure 2.1, transforms the models into a Constraint Program (CP) for detailed security analysis with solvers. These solvers perform logical calculations on models to validate properties, assess security mechanisms, identify vulnerabilities, and ensure compliance with requirements. Contextual variables guide the selection of optimal operationalizations, adapting the analysis to system conditions, and enhancing result relevance. This approach, which utilizes CP transformation and solvers, not only provides a systematic method for evaluating system robustness but also offers a qualitative security metric. It assists in the proactive identification and mitigation of security flaws, thereby ensuring a secure and resilient system design.

This chapter is structured as follows: Section 2.1 introduces the running example. Subsection 2.1.1 details the use of the SECRET template and SCORE ontology for the specification of requirements. Subsection 2.1.2 covers the formalization of these requirements via SERENA language models. Section 2.1.3 describes the transformation of SERENA models into constraint code for security analysis. The

chapter concludes with a summary of the approach.

2.1 Running Example

The running example used in this thesis is a secure smartphone defined by the German Federal Office for Information Security [1]. The document outlines a comprehensive set of security requirements for smartphone manufacturers, focusing on aspects such as EU law compliance, data actuality, unauthorized access protection, data privacy protection, reduction of attack surface, and advanced requirements.

The requirements were systematically organized and counted according to the specific security criterion they correspond to, as in Table 2.1. The security criteria mentioned in the table are derived from the SCORE ontology. SCORE, which stands for 'Security Criteria, Ontology, Requirements and Evaluation', is a systematic approach to identifying and categorizing security needs.

Table 2.1 – Security Criteria and Number of Requirements

Security Criteria	**Number of Requirements**
Maintainability	2
Access Control	6
Integrity	2
Privacy	5
Authorization	1
Resilience to Attacks	3
Immunity	1
Availability	1
Confidentiality	4
Location Privacy	1

The primary security criteria for smartphones, regardless of the number of requirements, include confidentiality, integrity, and availability, similar to all other domains [42][43][44][45][46]. We have selected three security requirements, detailed in Table 2.2, that align with the three main security criteria. These requirements will serve as the foundation for applying our approach using the running example.

Subsequently, we are able to define the functional requirements presented in Table 2.3.

1. https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/requirements/Requirements-Smartphones.pdf

Table 2.2 – Prose Security Requirements

Requirement	Description
SR1	From the network perspective, the use of the newest Radio Canal Ciphering Algorithms has very high priority. Devices supporting these algorithms are better protected.
SR2	The Hardware Security Element (HSE) must be used to store critical user data.
SR3	All new devices must be provided with the latest operating system (OS) available at release time.

Table 2.3 – Prose Functional Requirements

Requirement	Description
FR1	All smartphones should send the data using the cellular interface.
FR2	All smartphones should receive the data using the cellular interface
FR3	All smartphones should store the data.

2.1.1 Representation of the Running Example using the SECRET Template and the SCORE Ontology

In our effort to guide engineers in the process of writing and specifying security requirements, we have made two major contributions:

SECRET A security requirements template, based on the template proposed by Mazo et al. [6], developed to guide the definition of a security requirement by breaking it down into different distinct parts as shown in Figure 2.2. This template aims to provide a clear and detailed structure, simplifying the process of specifying security requirements.

SCORE A security ontology designed to assist engineers responsible for security requirements in defining security mechanisms related to specific domain-specific security criteria, as in Figure 2.3. SCORE also serves as a guide by suggesting additional security criteria to define, thereby increasing the security coverage of the system.

Definition 2.1.1 (Ontology) *An ontology is a formal and explicit specification of*

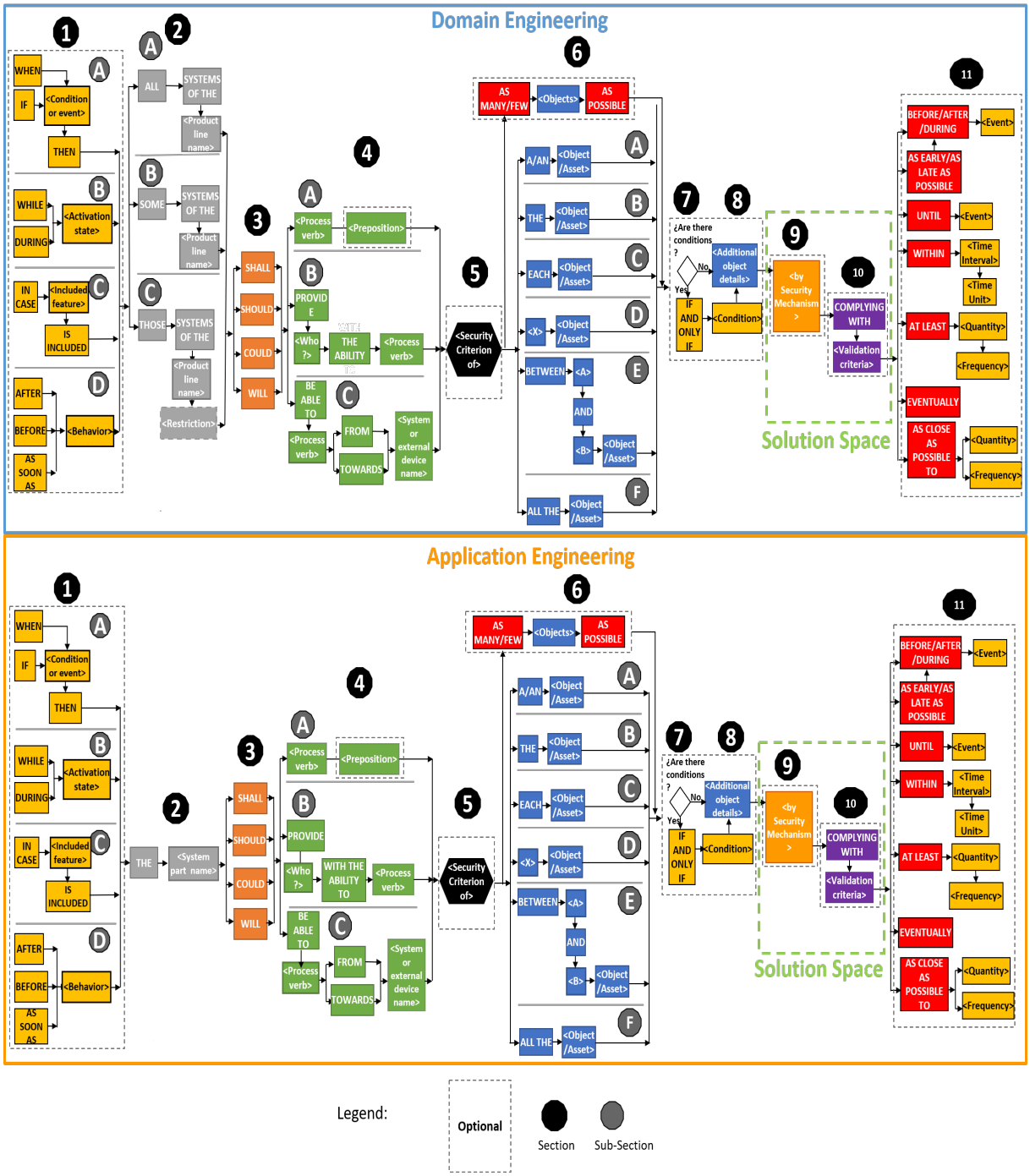


Figure 2.2 – SECRET Template

a shared conceptualization. In simpler terms, it is a structured way of representing knowledge about a particular domain.

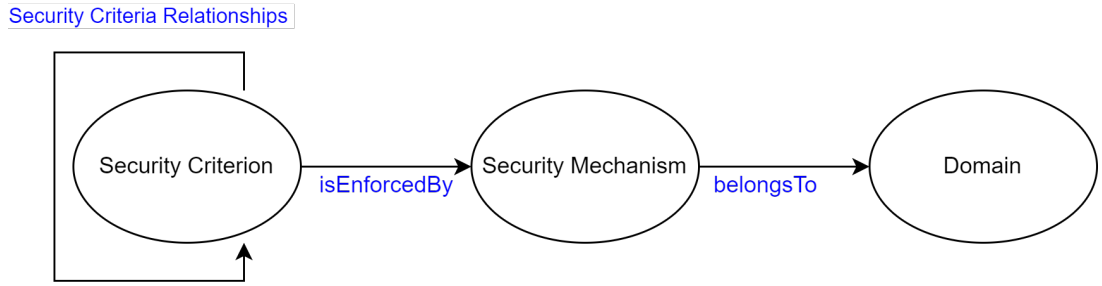


Figure 2.3 – SCORE Core Ontology

We have streamlined the use of the ontology (SCORE) and the template (SECRET) by integrating them into VariaMos [2], a web-based tool. VariaMos uses microservices with a generic and multi-solver approach for multi-language modeling and reasoning on products and product lines. This was a requirement specification module, allowing engineers to harmoniously define functional, non-functional, and security requirements tailored to domains, product lines, and applications.

Specifically, by coupling the ontology with the template, VariaMos can now determine which security criteria and mechanisms are applicable to a requirement based on the specified domain. This coupling also allows for the suggestion of additional security requirements, contributing to a more comprehensive specification tailored to each context.

Domain Requirements Specification

Using the domain engineering space of the SECRET template, we express the requirements as domain or product-line requirements in Tables 2.4 and 2.5. We set the variability to all systems for simplicity.

Using the ontology enhances our security requirements with greater specificity, enabling the definition of a wider range of security criteria and mechanisms, thus improving the system's security posture. We chose to focus on confidentiality to further simplify our running example. Focusing on confidentiality streamlines our example and deepens our understanding of this security aspect, demonstrating how the ontology effectively addresses confidentiality concerns. Figure 2.4 represents the additional security criteria from SCORE related to confidentiality in the

2. https://variamos.com/

Table 2.4 – Domain Functional Requirements using SECRET

Requirement	Description
FR1	<All systems of the smartphones productline>$_{system}$<shall>$_{priority}$<send>$_{activity}$<the data>$_{object}$<using the cellular interface>$_{additionalObjectDetails}$
FR2	<All systems of the smartphones productline>$_{system}$<shall>$_{priority}$<receive>$_{activity}$<the data>$_{object}$<using the cellular interface>$_{additionalObjectDetails}$
FR3	<All systems of the smartphones productline>$_{system}$<shall>$_{priority}$<store>$_{activity}$<the data>$_{object}$

ID	DESCRIPTION
SR1	<All systems of the smartphones productline>$_{SystemOrSystemPart}$<should>$_{priority}$< ensure>$_{activity}$<confidentiality of>$_{securityCriterion}$<the data>$_{object}$<by Radio Canal Ciphering Algorithms>$_{securityMechanism}$
SR2	<All systems of the smartphones product line>$_{SystemOrSystemPart}$<should>$_{priority}$<ensure>$_{activity}$<integrity of>$_{securityCriterion}$<the data>$_{object}$<by storing security-critical data in the HSE>$_{securityMechanism}$
SR3	<All systems of the smartphones product line>$_{SystemOrSystemPart}$<should>$_{priority}$<ensure>$_{activity}$< availability of>$_{securityCriterion}$<the latest operating system (OS)>$_{object}$

Table 2.5 – Domain Security Requirements using SECRET & SCORE

smartphone domain. The security criteria are color-coded by priority: red for high, green for medium, and yellow for low. The sub-requirements include:

— $\xrightarrow[\text{SR1}]{\text{Sub-Req of}}$ **SR1.1:** SR-AC-01 - All systems of the smartphones product line SHALL guarantee access control of data by implementing remote access control.

— $\xrightarrow[\text{SR1}]{\text{Sub-Req of}}$ **SR1.2:** SR-IMM-01 - All systems of the smartphones product line SHOULD support the immunity of the system by implementing an anti-virus.

— $\xrightarrow[\text{SR1}]{\text{Sub-Req of}}$ **SR1.3:** SR-ANO-01 - All systems of the smartphones product line COULD grant anonymity of users by implementing Periodic N-Times Anonymous Authentication.

Anonymity in requirement SR-ANO-01 is suggested due to its low priority with confidentiality, but not enforced.

Similarly, starting from **SR1.1:** SR-AC-01, and the access control criterion, we

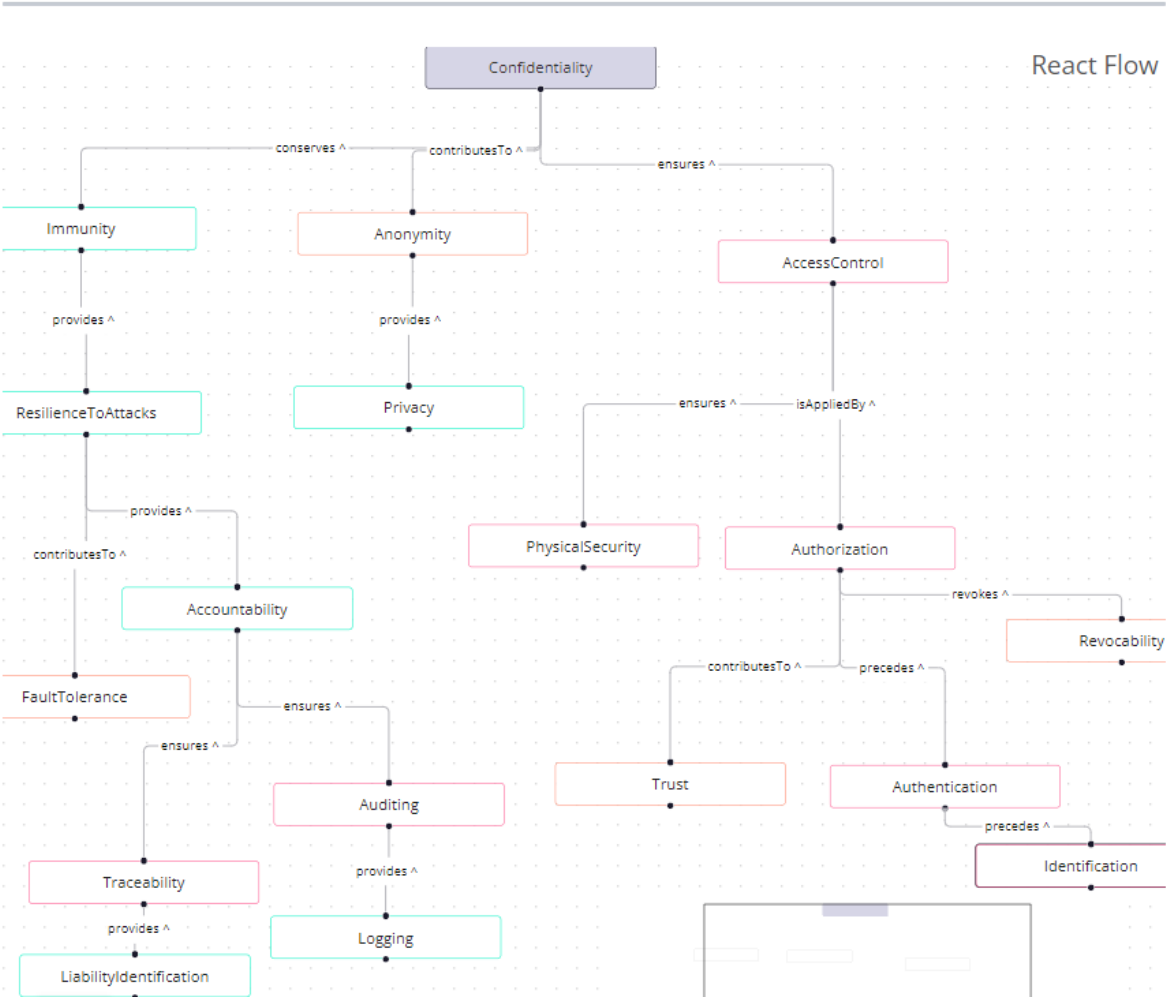


Figure 2.4 – Confidentiality Additional Security Criteria from SCORE

defined the following security requirements using the sub-criteria of access control in the confidentiality sub-ontology :

- $\xrightarrow[\text{SR1.1}]{\text{Sub-Req of}}$ **SR1.1.1:** SR-AUTHO-01 - All systems of the smartphones product line SHALL guarantee the authorization of users by implementing OAUTH.
- $\xrightarrow[\text{SR1.1.1}]{\text{Sub-Req of}}$ **SR1.1.1.1:** SR-AUTHE-01 - All systems of the smartphones product line SHALL guarantee the authentication of the user by implementing asymmetric encryption.
- $\xrightarrow[\text{SR1.1.1.1}]{\text{Sub-Req of}}$ **SR1.1.1.1.1:** SR-ID-01 - All systems of the smartphones product line SHALL guarantee the identification of the user by implementing fingerprints.
- $\xrightarrow[\text{SR1.1.1}]{\text{Sub-Req of}}$ **SR1.1.1.2:** SR-REV-01 - All systems of the smartphones product line COULD grant revocability of authorization by implementing root privilege management.
- $\xrightarrow[\text{SR1.1.1}]{\text{Sub-Req of}}$ **SR1.1.1.3:** SR-TR-01 - All systems of the smartphones product line COULD grant trust of data by implementing a trusted platform module.
- $\xrightarrow[\text{Req-1.1}]{\text{Sub-Req of}}$ **SR1.1.2:** SR-PS-01 - All systems of the smartphones product line SHALL guarantee the physical security of system parts by implementing a trusted execution environment.

From SR1.2. SR-IMM-01, the sub-requirements are:

- $\xrightarrow[\text{SR1.2}]{\text{Sub-Req of}}$ **SR1.2.1:** SR-RES-01 - All systems of the smartphones product line SHOULD support resilience to attack by implementing an intrusion detection system.
- $\xrightarrow[\text{SR1.2.1}]{\text{Sub-Req of}}$ **SR1.2.1.1:** SR-FT-01 - All systems of the smartphones product line COULD grant fault tolerance of the system by implementing a Markov chain based monitoring service.
- $\xrightarrow[\text{SR1.2.1}]{\text{Sub-Req of}}$ **SR1.2.1.2:** SR-ACC-01 - All systems of the smartphones product line SHOULD support the accountability of users by implementing privacy preserving accountability protocol.
- $\xrightarrow[\text{SR1.2.1.2}]{\text{Sub-Req of}}$ **SR1.2.1.2.1:** SR-AUD-01 - All systems of the smartphones product line SHALL guarantee the auditing of data by implementing OWASP.
- $\xrightarrow[\text{SR1.2.1.2.1}]{\text{Sub-Req of}}$ **SR1.2.1.2.1.1:** SR-LOG-01 - All systems of the smartphones product line SHOULD support the logging of events by implementing a trusted computing base.
- $\xrightarrow[\text{SR1.2.1.2}]{\text{Sub-Req of}}$ **SR1.2.1.2.2:** SR-TRA-01 - All systems of the smartphones product line SHALL guarantee the traceability of events by implementing an anonymous

bilateral access control protocol.

— $\xrightarrow[\text{SR1.2.1.2.2}]{\text{Sub-Req of}}$ **SR1.2.1.2.2.1:** SR-LI-01 - All systems of the smartphones product line SHOULD support liability identification of the system by implementing trust enhanced anonymous on-demand routing protocol.

The inter-requirements traceability relationships are then specified between the requirements. **This is further explained in Chapter 3.**

System Configuration

Definition 2.1.2 (Features Model) *A feature model represents software system features and their relationships, managing variability in product lines by defining configurable aspects and dependencies.*

After defining domain requirements, the subsequent step is to detail the requirements for a specific system, a smartphone, by adapting these domain requirements to the system's unique needs and characteristics. This is effectively done through a feature model, a formalism for depicting commonalities and variabilities in a product line or system family. It facilitates the organized management of features—distinct characteristics or functionalities that enhance the system's overall behavior and capabilities. Configuring the domain requirements represented in a feature model allows for customizing them to align with the system's desired features and functionalities, involving selecting and prioritizing pertinent features and adding any necessary system-specific requirements.

All features from the mandatory features model in Figure 2.5 were transformed into requirements for a specific system as in Tables 2.6, 2.7, and 2.8. We ensured comprehensive inclusion by meticulously converting each feature into tailored requirements, aiming for a complete representation of the system's intended functionalities and capabilities. This process established a precise set of requirements, foundational for the system's successful development and implementation.

However, this step has not been emphasized in this thesis and thus has not been implemented.

Application Requirements Specification or Generation

For the application specification, we convert domain requirements into application requirements in Table 2.6, 2.7, and 2.8, using the SECRET template shown

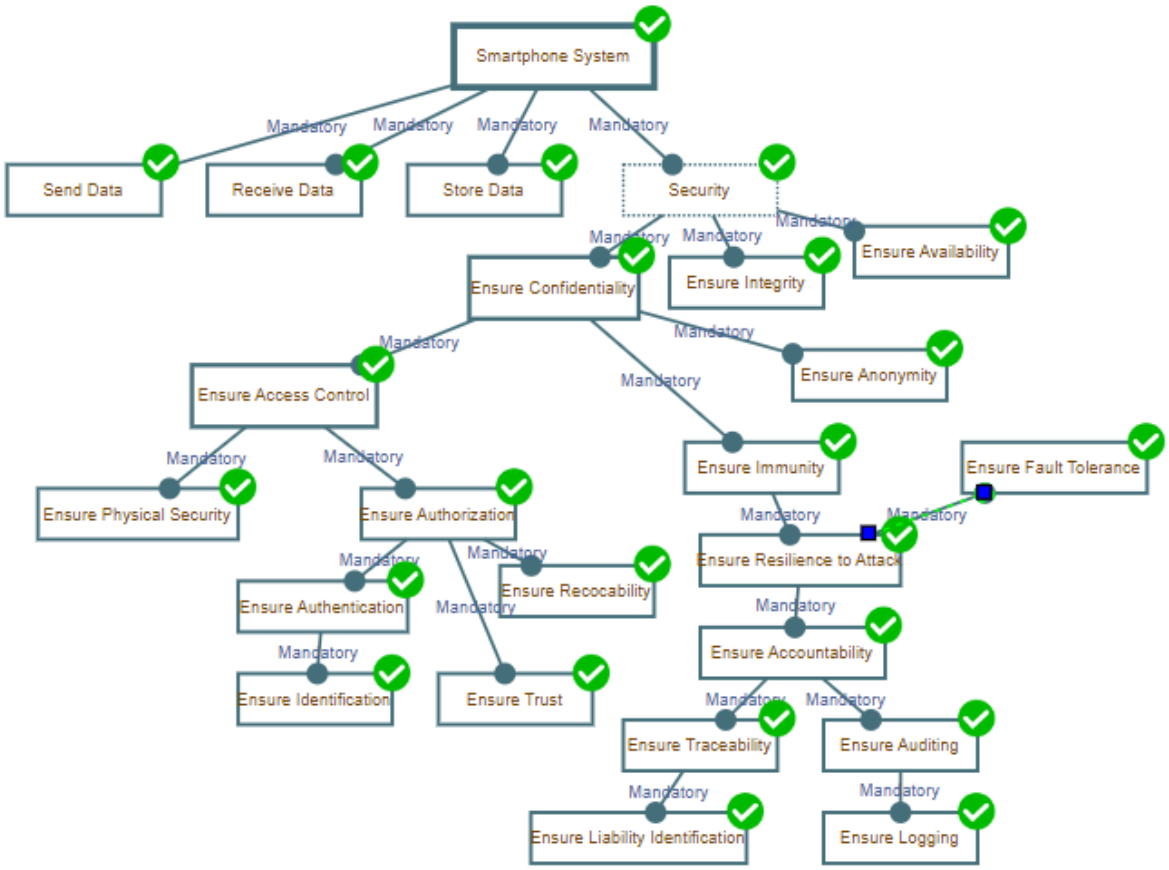


Figure 2.5 – Features Model for System Configuration

in Figure 2.2.

Table 2.6 – Application Functional Requirements using SECRET

Requirement	Description
FR1	<The smartphone>$_{system}$<shall>$_{priority}$<send>$_{activity}$<the data>$_{object}$<using the cellular interface>$_{additionalObjectDetails}$
FR2	<The smartphone>$_{system}$<shall>$_{priority}$<receive>$_{activity}$<the data>$_{object}$<using the cellular interface>$_{additionalObjectDetails}$
FR3	<The smartphone>$_{system}$<shall>$_{priority}$<store>$_{activity}$<the data>$_{object}$

ID	DESCRIPTION
SR1	<The smartphone>$_{SystemOrSystemPart}$<should>$_{priority}$<ensure>$_{activity}$<confidentiality of>$_{securityCriterion}$<the data>$_{object}$<by Radio Canal Ciphering Algorithms>$_{securityMechanism}$
SR2	<The smartphone>$_{SystemOrSystemPart}$<should>$_{priority}$<ensure>$_{activity}$<integrity of>$_{securityCriterion}$<the data>$_{object}$<by storing security-critical data in the HSE>$_{securityMechanism}$
SR3	<The smartphone>$_{SystemOrSystemPart}$<should>$_{priority}$<ensure>$_{activity}$<availability of>$_{securityCriterion}$<the latest operating system (OS)>$_{object}$

Table 2.7 – Application Security Requirements using SECRET & SCORE

2.1.2 Transformation of the Running Example into SERENA Models

The transition from textual to formal representation within the modeling phase is crucial in the security engineering process. This strategic step focuses on transforming the security requirements specified in natural language into a formal representation. Originally defined using the SECRET template and the SCORE ontology, these textual requirements undergo a systematic transformation to be explicitly detailed in the formal modeling language SERENA.

This approach aims to bring clarity and rigor to security specifications by anchoring them in a formal framework, facilitating deep understanding and systematic analysis. The different facets of requirements, initially encapsulated in the SECRET template, are preserved and transposed into a coherent formal language, allowing for a precise representation of the system's security needs.

Parent Requirement	Requirement ID	Description
SR1	SR1.1	SR-AC-01 - The smartphone SHALL guarantee access control of data by implementing remote access control.
SR1	SR1.2	SR-IMM-01 - The smartphone SHOULD support the immunity of the system by implementing an anti-virus.
SR1	SR1.3	SR-ANO-01 - The smartphone COULD grant anonymity of users by implementing Periodic N-Times Anonymous Authentication.
SR1.1	SR1.1.1	SR-AUTHO-01 - The smartphone SHALL guarantee the authorization of users by implementing OAUTH.
SR1.1.1	SR1.1.1.1	SR-AUTHE-01 - The smartphone SHALL guarantee the authentication of the user by implementing asymmetric encryption.
SR1.1.1.1	SR1.1.1.1.1	SR-ID-01 - The smartphone SHALL guarantee the identification of the user by implementing fingerprints.
SR1.1.1	SR1.1.1.2	SR-REV-01 - The smartphone COULD grant revocability of authorization by implementing root privilege management.
SR1.1.1	SR1.1.1.3	SR-TR-01 - The smartphone COULD grant trust of data by implementing a trusted platform module.
SR1.1	SR1.1.2	SR-PS-01 - The smartphone SHALL guarantee the physical security of system parts by implementing a trusted execution environment.
SR1.2	SR1.2.1	SR-RES-01 - The smartphone SHOULD support resilience to attack by implementing an intrusion detection system.
SR1.2.1	SR1.2.1.1	SR-FT-01 - The smartphone COULD grant fault tolerance of the system by implementing a Markov chain based monitoring service.
SR1.2.1	SR1.2.1.2	SR-ACC-01 - The smartphone SHOULD support the accountability of users by implementing privacy preserving accountability protocol.
SR1.2.1.2	SR1.2.1.2.1	SR-AUD-01 - The smartphone SHALL guarantee the auditing of data by implementing OWASP.
SR1.2.1.2.1	SR1.2.1.2.1.1	SR-LOG-01 - The smartphone SHOULD support the logging of events by implementing a trusted computing base.
SR1.2.1.2	SR1.2.1.2.2	SR-TRA-01 - The smartphone SHALL guarantee the traceability of events by implementing an anonymous bilateral access control protocol.
SR1.2.1.2.2	SR1.2.1.2.2.1	SR-LI-01 - The smartphone SHOULD support liability identification of the system by implementing trust enhanced anonymous on demand routing protocol.

Table 2.8 – Additional Security Requirements

The use of SERENA as a formal modeling language provides a structured basis for expressing the complex details of security requirements. This transition ensures the preservation of essential properties of textual specifications, ensuring that key elements, such as security criteria from the SCORE ontology, are integrated accurately and contextualized within formal models.

By following this approach, the modeling phase ensures that security requirements, originally defined informally, acquire a more rigorous form and are ready to undergo formal verification, validation, and analysis processes.

The modeling language and its automatic transformations were implemented in VariaMos. This step is crucial to ensure consistency, accuracy, and fidelity of the specifications throughout the modeling process while adhering to the specific syntax and semantics rules inherent to SERENA.

SERENA: SEcurity REquirements aNAlysis is a goal-oriented modeling language for specifying and analyzing security requirements in software systems. Drawing inspiration from methodologies like Sawyer et al. [47] (KAOS), Secure i* [19], SecureTropos [20], and CORAS [25], SERENA emphasizes formalizing security objectives, semantic analysis, and security by design principles. This approach helps organizations systematically define and address security objectives, improving software security. The language is composed of five different models, each with a specific objective and representation. The meta-model of SERENA is presented in Figure 2.6. **The transformation rules are detailed in Chapter 4.**

Definition 2.1.3 (Goal) *Represent functional goals which indicate the (functional) purpose of the system. Goals are either satisfied (true) or denied (false).*

Definition 2.1.4 (Asset) *The entity or resource that needs to be protected from an attack.*

Definition 2.1.5 (Operationalizations) *Specify how a goal can be achieved. Operationalizations can be either enabled (true) or disabled (false).*

Definition 2.1.6 (Vulnerability) *A flaw or weakness that can be exploited to harm a system or computer activity. (NIST)*

Definition 2.1.7 (Threats) *Circumstances that have the potential to cause losses or problems that may endanger the security characteristics of the system. Examples*

of threats include social engineering attacks, password sniffing, and eavesdropping. Any circumstance or event that may have a negative impact on the operations of the organization, its assets, individuals, other organizations, or the nation through a system via unauthorized access, destruction, disclosure, modification of information, and/or denial of service.

Definition 2.1.8 (Security Mechanisms) *Represent standard security methods to contribute to achieving protection goals. Some of these methods can prevent security attacks, while others can only detect security breaches.*

Definition 2.1.9 (Claims) *Indicate assumptions about the satisfaction of software goals. A claim is considered TRUE until and unless execution control shows that the assumption is not true.*

Definition 2.1.10 (Softgoals) *Specify non-functional requirements of the system. The degree of satisfaction of a softgoal is modeled on an ordinal scale where the set of values is –, -, =, +, ++, ranging from complete denial (–), through neutrality or indeterminacy (=), to complete satisfaction (++).*

Definition 2.1.11 (Soft Influences) *Express the required satisfaction levels of soft goals for a particular value of the context variable. They are soft because it may be impossible to satisfy them for all possible values.*

Definition 2.1.12 (Context Variables) *Abstractions of a part of the system's environment, controlled at runtime by detection. The context is the combination of all context variables, and the set of all contexts represents the variability of the environment.*

Transforming Security Requirements into SERENA Security Criteria Model

It represents the security criteria defined in the SECRET model along with the complex relationships that exist between them. It provides a detailed view of the fundamental elements that structure the system's security requirements. The main objective of the security criteria model is to analyze the security criteria against the SCORE ontology.

The transformation from the security requirements in Tables 2.7 and 2.8 to the security criteria model is represented in Figure 2.7.

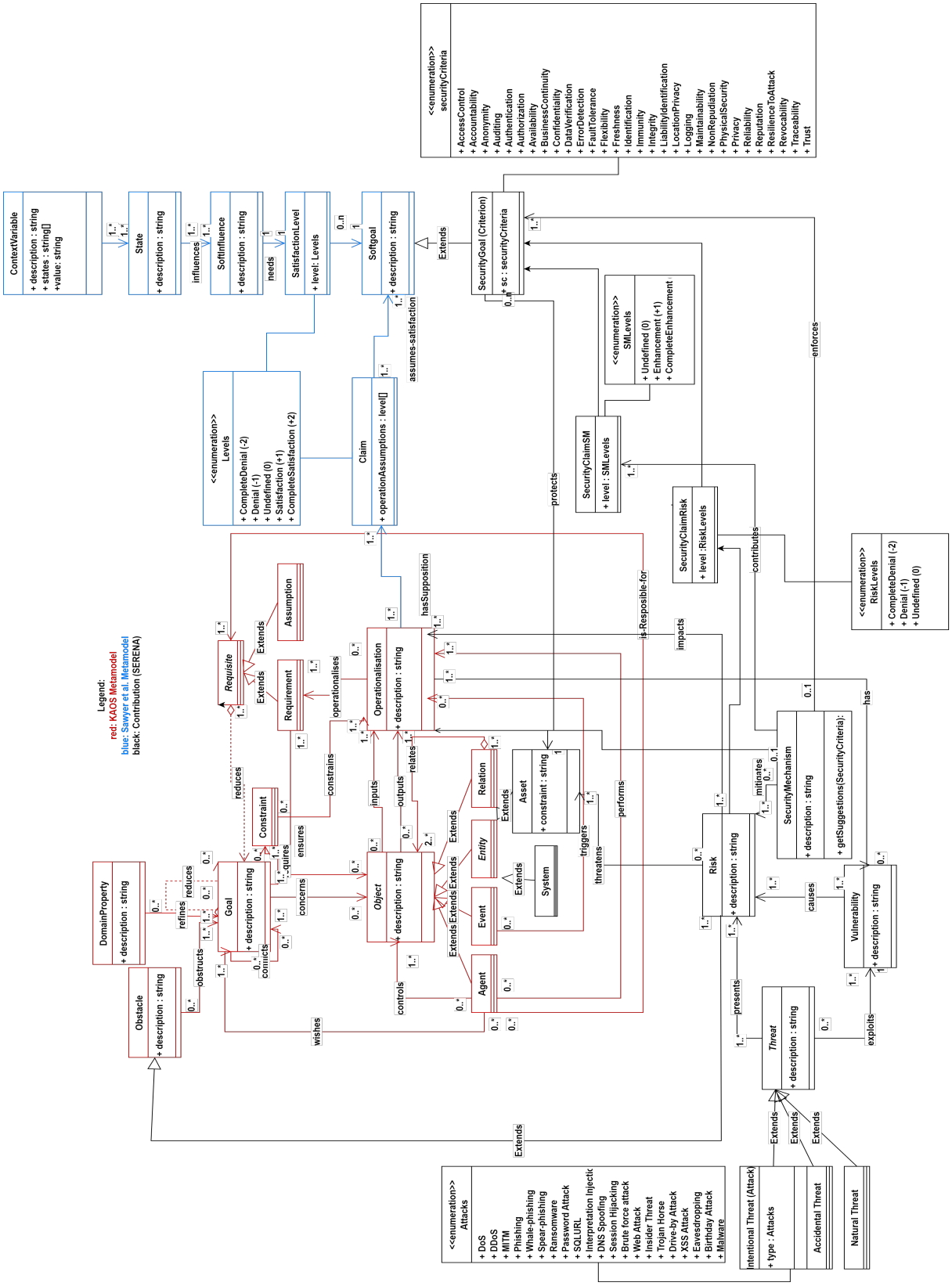


Figure 2.6 – SERENA Meta-model

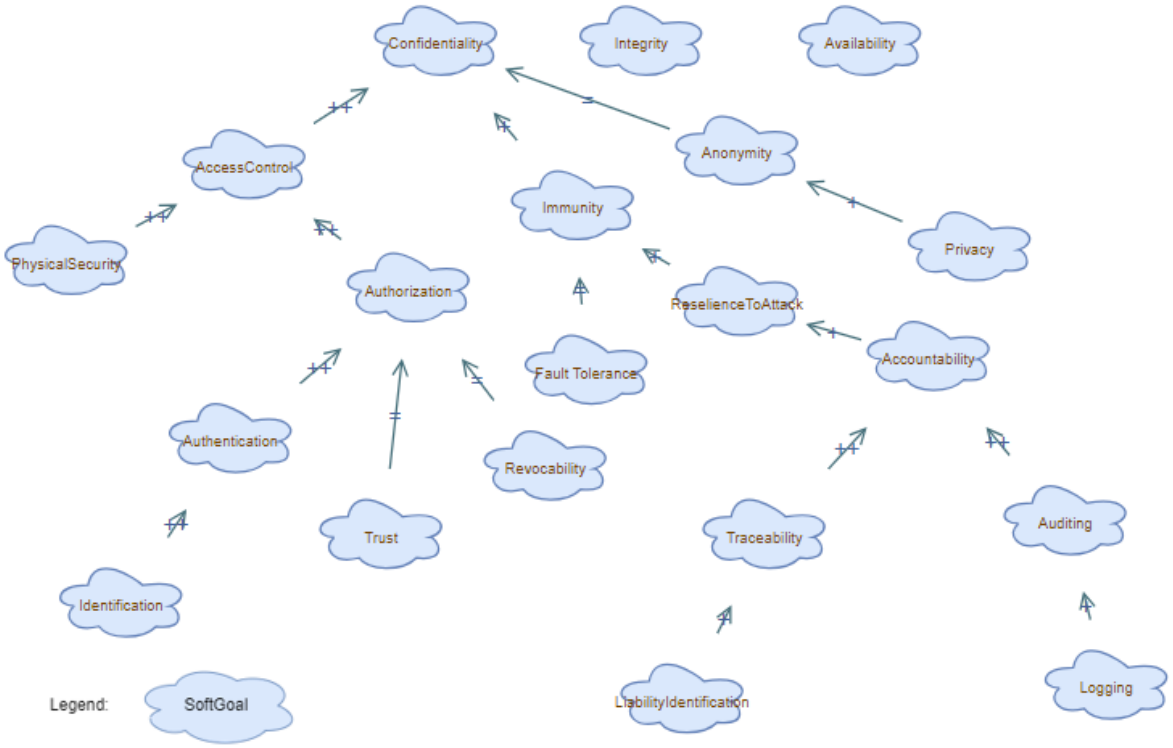


Figure 2.7 – Smartphone SERENA Security Criteria Model

Transforming Functional Requirements into SERENA Goal Model

The second model represents the system's functional goals and operationalizations. It establishes the link between functional goals and the concrete actions that realize them, providing a detailed view of the expected system functionality. Users can specify additional goals (subgoals) and operationalizations if they are not defined in the requirements. The functional requirements specified in Table 2.6 are transformed into the goals in Figure 2.8, and the operationalizations were added.

Evaluating the Risks on Operationalizations to Create the SERENA Risk Model

In this third model (Figure 2.9), the focus is to evaluate the risks linked to threats against operationalizations of functional goals. It pinpoints specific threat scenarios and evaluates their impact on security criteria for a comprehensive risk assessment. Users are required to identify vulnerabilities, threats, and risk levels.

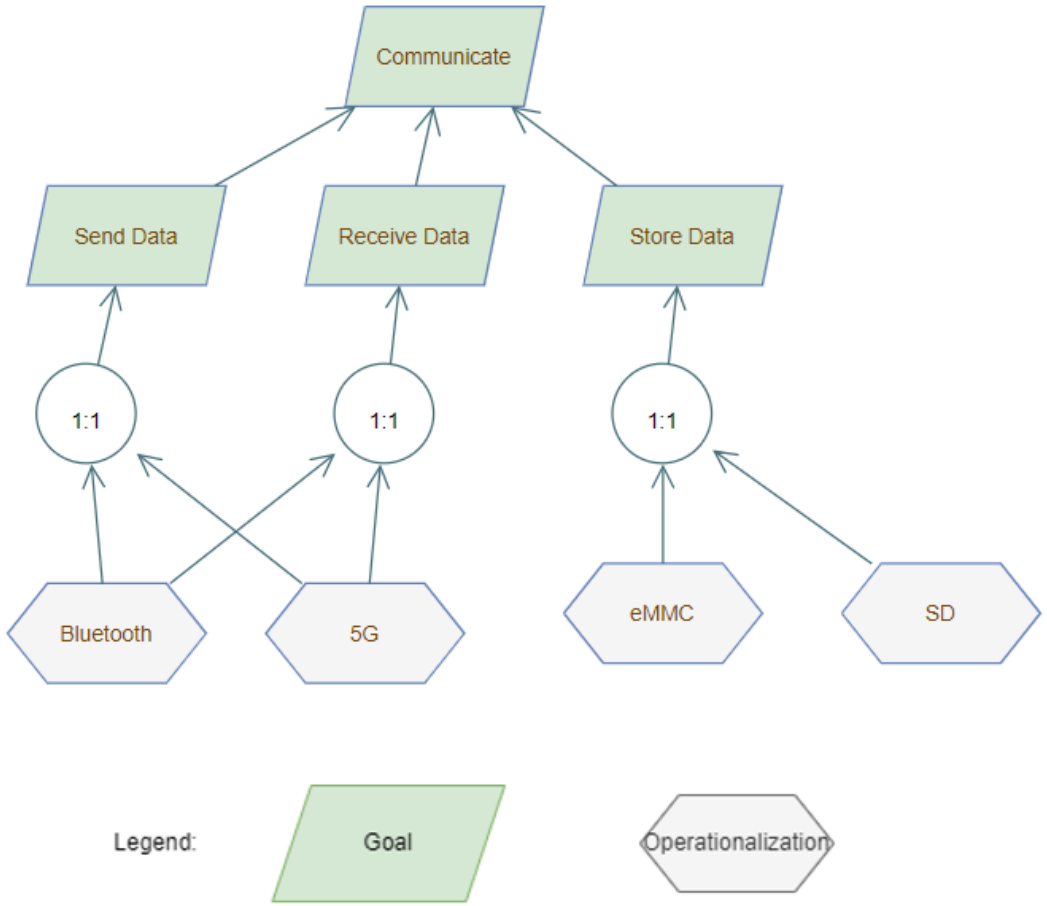


Figure 2.8 – Smartphone Goal Model

The risk value assigned to each threat in the SERENA risk model is not arbitrary but is determined by applying the risk assessment method defined in NIST SP 800-30 [34]. Risk levels are denoted as '=' for very low, '-' for low and medium, and '–' for high and very high in Table 2.9. We chose to represent five security criteria (softgoals) to provide a simpler overview of the security requirements for the system. The approach requires filling in the risk evaluation table with details of the identified threats and vulnerabilities and then calculating the risk level using the 5x5 matrix. This involves assessing each threat in terms of likelihood of occurrence and potential impact and then crossing these two factors to determine the corresponding risk level in the matrix. Subsequently, the risk level assigned to each threat is reported in the SERENA risk model for further security analysis

Table 2.9 – NIST Risk Evaluation Matrix

Likelihood	**Impact**				
	Very Low	**Low**	**Medium**	**High**	**Very High**
Very Low	Low	Low	Medium	High	High
Low	Low	Medium	High	High	Very High
Medium	Medium	High	High	Very High	Very High
High	High	High	Very High	Very High	Very High
Very High	High	Very High	Very High	Very High	Very High

(Table 2.10).

Using the NIST risk evaluation method, the overall risk level of each threat was estimated. However, in the next step, the risk level of each threat in each security criterion needed to be assessed (Table 2.11). By assigning individual risk levels to each threat for each security criterion, a detailed view of the system's vulnerability to different threats was obtained. This process started with the global risk level as the maximum level. The risk levels can be interpreted from the MITRE Common Weakness Enumeration [3] (from the Common Consequences section).

Evaluating the Security Mechanisms on Operationalizations to Create the SERENA Treatment Model

The fourth model integrates security mechanisms designed to mitigate identified threats. It highlights the specific mechanisms added to enhance the system's security and evaluates their respective contributions to protecting the defined security criteria. Here, security mechanisms and their security contributions are specified, with those mentioned in the security requirements automatically included. The security mechanisms are extracted from the five security requirements SR1, SR2, SR3, SR1.1, SR1.2 (Tables 2.7 and 2.8) from which the security criteria represented in 2.9 were selected. To specify which security mechanism acts on which threat, it is possible to use certain attack and defense databases, such as the MITRE Defense Database [4]. These databases provide detailed information on known attack techniques as well as recommended countermeasures to mitigate them. Using these databases as a reference, users can associate appropriate security mechanisms with

3. https://cwe.mitre.org/data/definitions/325.html
4. https://d3fend.mitre.org/offensive-technique/attack/T1557/

Threat Event	Threat Sources	Threat Source Characteristics			Relevance	Likelihood of Attack Initiation	Vulnerabilities and Predisposing Conditions	Severity and Pervasiveness	Likelihood Initiated Attack Succeeds	Overall Likelihood	Level of Impact	Risk
		Capability	**Intent**	**Targeting**								
Modify system behaviour	Hacker	High	Obstruct Functioning	5G, SD	High	Moderate	Data Leakage, Arbitrary Code Execution	High	Low	Low	High	High
Detect confidential data	MITM	Moderate	Data detection	Bluetooth	High	Moderate	Bluetooth Mesh Profile Authvalue Leak	High	Moderate	Moderate	Moderate	High
Stop or modify services	Malware	Moderate	Obstruct Functioning	e-MMC	High	High	Firmware Insecurity	High	Moderate	Moderate	High	High

Table 2.10 – Adversarial Risks Evaluation

	Confidentiality	Integrity	Availability	Access Control	Immunity
Hacker	–	-	-	-	–
MITM	–	–	-	=	–
Malware	-	–	–	-	–

Table 2.11 – Risk Evaluation for Security Criteria

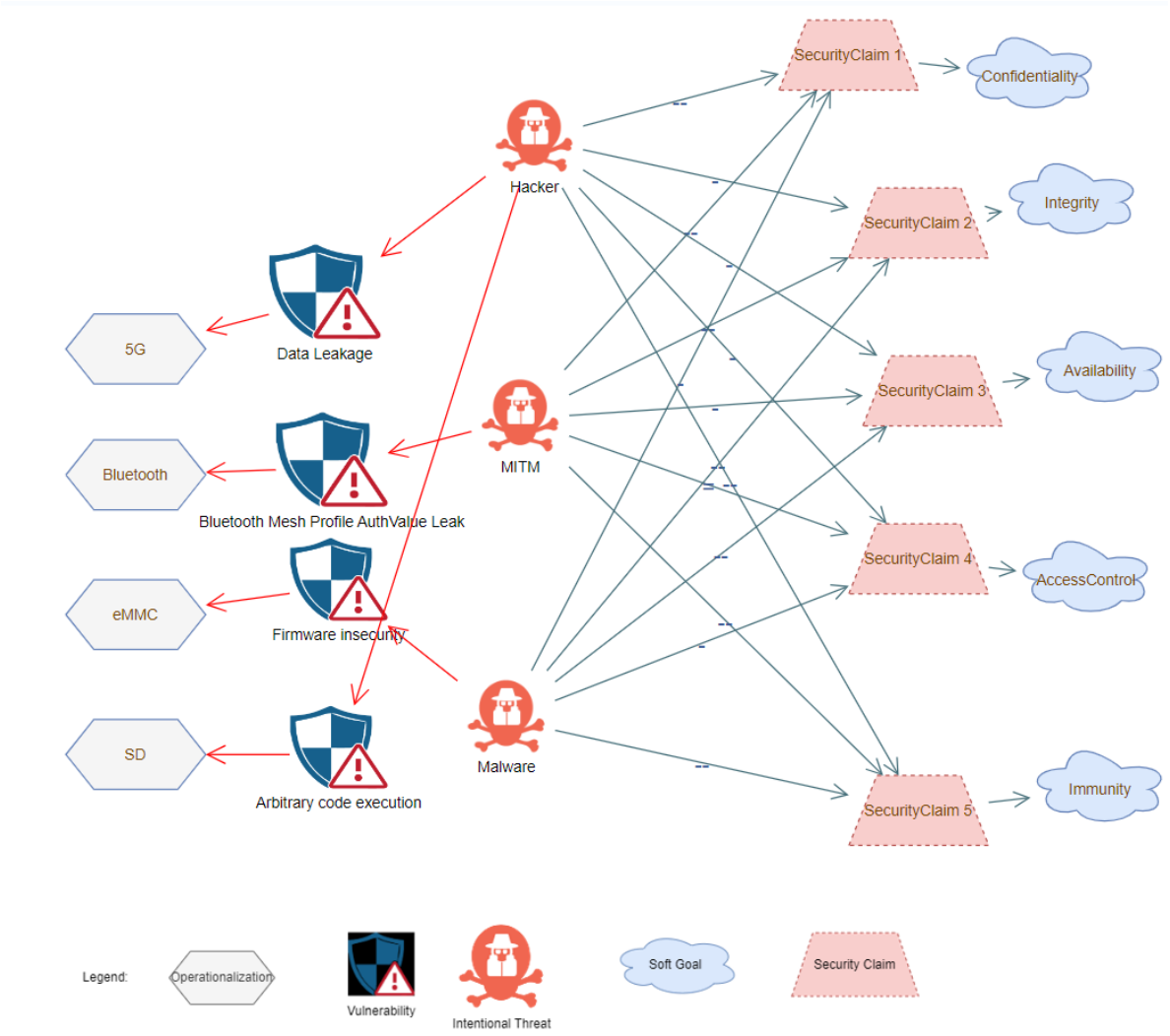


Figure 2.9 – Smartphone SERENA Risk Model

specific threats identified in the risk model, thereby improving the system's ability to protect against cyber threats, as shown in Table 2.12.

The final step involves estimating each mechanism's security level to each cri-

	Radio Canal Ciphering Algorithms	Storing security-critical data in the HSE	Latest operating system (OS)	Remote access control	Anti-virus
Hacker	X	X	X	X	
MITM	X	X		X	
Malware		X	X		X

Table 2.12 – Security Mechanisms mitigate Threats

terion (Table 2.13).For estimation, refer to security standards and best practices from industry guides or standardization bodies [5].

	Radio Canal Ciphering Algorithms	Storing security-critical data in the HSE	Latest operating system (OS)	Remote access control	Anti-virus
Confidentiality	++	++	+	++	+
Integrity	++	++	+	++	+
Availability	=	=	+	+	=
Access Control	=	+	=	++	+
Immunity	=	++	++	=	++

Table 2.13 – Security Mechanism Contribution to Security Criteria

Federation of the Goal, Risk, and Treatment SERENA Models to Create the SERENA Overall Model

Finally, the fifth model (Figure 2.11) provides an overview by integrating goals, operationalizations, the overall security analysis of operationalizations on security criteria, as well as other non-functional objectives (soft goals). It also considers contextual variables that can influence the system's security, thus providing a comprehensive and holistic representation of security requirements. Contextual variables and their "soft influences" must be specified in this model. Softgoal satisfaction is modeled on an ordinal scale –, -, =, +, ++, from complete denial (–) to complete satisfaction (++). The scale ranges from -2 (–) to 2 (++).

5. https://www.enisa.europa.eu/topics/iot-and-smart-infrastructures/smartphone-guidelines-tool

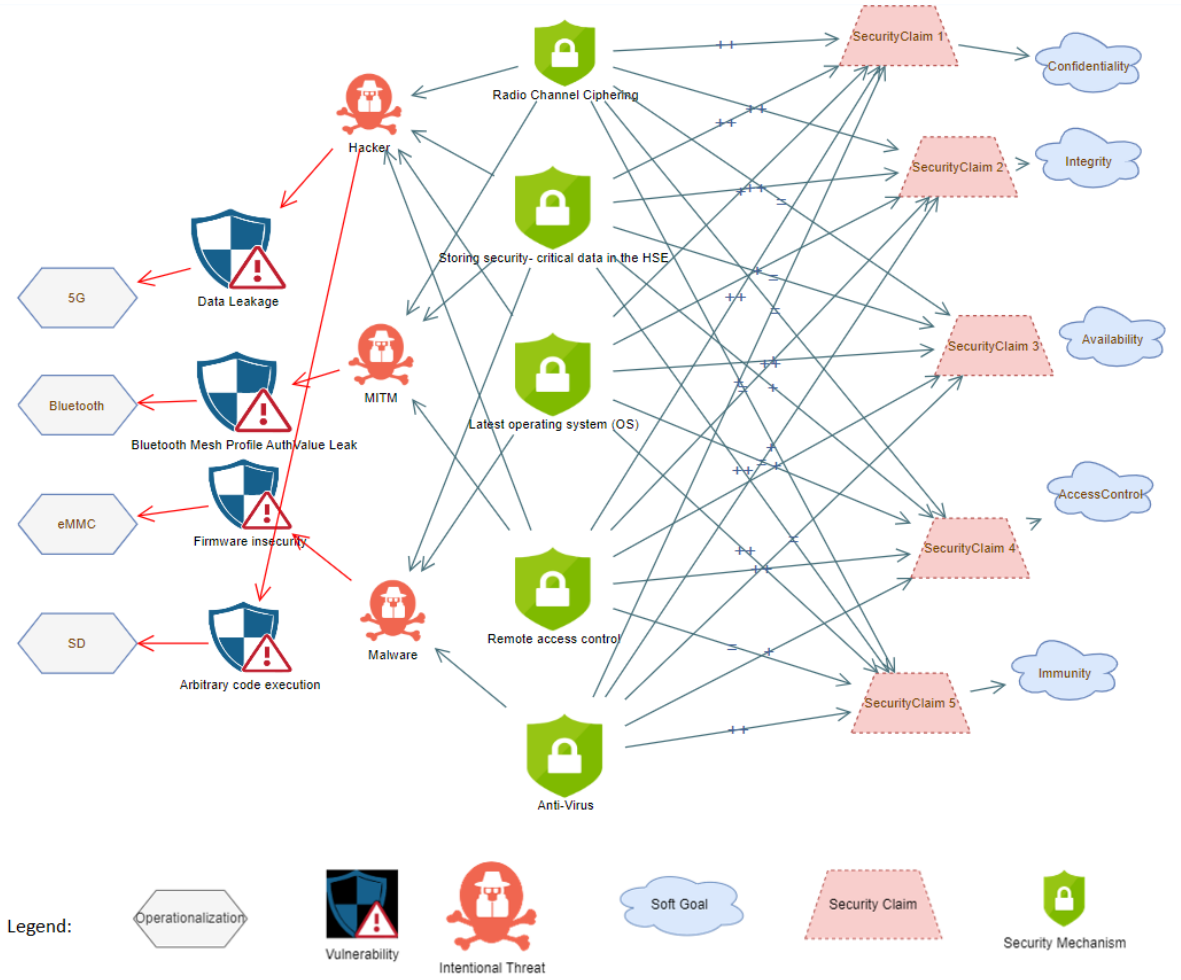


Figure 2.10 – Smartphone SERENA Treatment Model

The overall security value (OSV) for each operationalization (O) relative to a security criterion (SC) is the sum of the risk values (RV) from all threats (T) and the defense values (DV) from all security mechanisms (SM). Mathematically, it is expressed as:

$$OSV_{O,SC} = \sum_{T} RV_T + \sum_{SM} DV_{SM} \tag{2.1}$$

Where:

— $OSV_{O,SC}$ represents the overall security value of operationalization O towards security criterion SC.

— RV_T represents the risk value of threat T.

— DV_{SM} represents the defense value of security mechanism SM.

For example, the overall security value for 5G and Confidentiality is calculated as follows:

$$OSV_{5G,Confidentiality} = RV_{Hacker} + DV_{RadioChannelCiphering} + DV_{Storingsecurity-criticaldatainthe HSE} + DV_{Latestoperatingsystem(OS)} + DV_{Remoteaccesscontrol} = -2 + 2 + 2 + 1 + 2 = 5 \quad (2.2)$$

The numerical value is translated back to its original value, where 5 (greater than 2) indicates complete satisfaction (++). Table 2.14 lists all security values for each operationalization according to security criteria.

	Confidentiality	Integrity	Availability	Access Control	Immunity
5G	++	++	+	++	++
Bluetooth	++	++	=	++	=
eMMC	++	++	-	+	++
SD	++	++	+	++	++

Table 2.14 – Overall Security Value (Claim) by Operationalization

Context Variables The two context variables that could be identified for the system are Location Sensitivity and Network Connectivity.

— Location Sensitivity:

- Low: The smartphone is in an area with minimal surveillance or no surveillance, such as a residential neighborhood or rural area.
- Medium: The smartphone is in an area with periodic surveillance or surveillance conducted by automated monitoring systems, such as a public park or shopping mall.
- High: The smartphone is in an area constantly monitored by security personnel or advanced surveillance systems, such as a government building or airport.

— Network Connectivity:

- Secure: The smartphone is connected to a trusted Wi-Fi network or cellular network with strong encryption and security protocols.

- Public: The smartphone is connected to a public Wi-Fi network or open cellular network without encryption or security measures.
- Offline: The smartphone is not connected to any network and operates offline.

The final involves defining the soft influence required by the system at a certain level of context variable (Table 2.15). It evaluates how different context conditions can influence the system's security requirements and determines the appropriate level of security needed to address these specific conditions. This ensures that the system is configured to operate securely in different environments or usage scenarios.

Context Variable		Security Criterion				
		Confidentiality	Integrity	Availability	AccessControl	Immunity
Location Sensitivity	Low	+	+	=	=	=
	Medium	++	+	+	+	++
	High	++	++	++	+	++
Network Connectivity	Secure	+	=	+	+	+
	Public	++	++	++	++	++
	Offline	=	=	=	=	+

Table 2.15 – Context Variable Soft Influence

2.1.3 Transformation of the SERENA Model to Constraint Code for Security Analysis

Constraint Programming Constraint programming (CP) solves combinatorial problems using methods from artificial intelligence, computer science, and operations research [48]. In constraint programming, users declaratively state the constraints on the feasible solutions for a set of decision variables. Constraints differ from traditional programming constructs in that they specify the properties of a solution to be found rather than a sequence of steps to execute [48]. The roots of CP can be traced back to constraint logic programming, which embeds constraints into a logic program. This was first introduced by Jaffar and Lassez in 1987, extending a specific class of constraints in Prolog II. The first implementations were Prolog III, CLP(R), and CHIP [48]. The key aspects of CP include modeling a problem as a set of decision variables and constraints, propagating the constraints

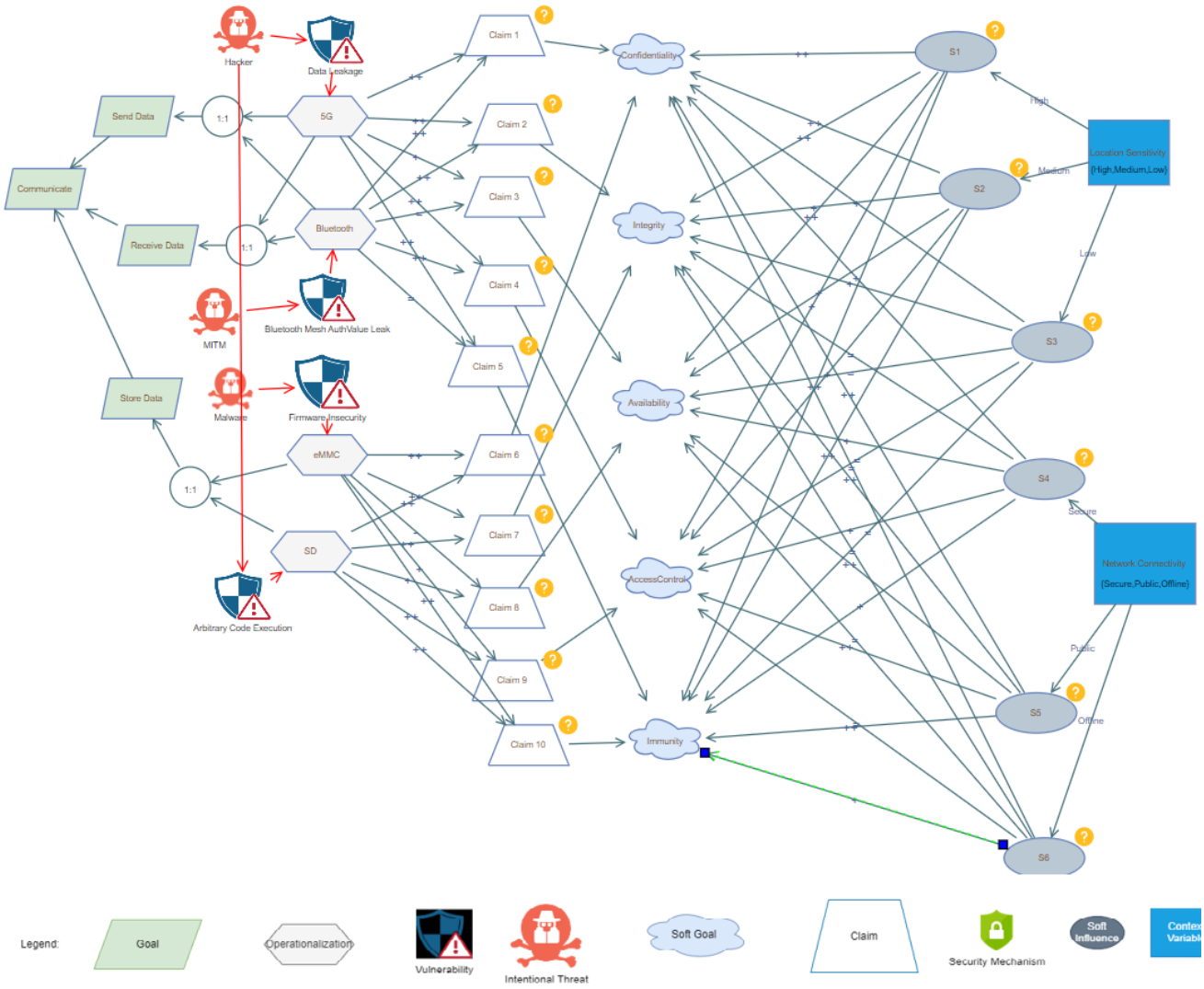


Figure 2.11 – Smartphone Overall SERENA Model

to prune the search space, and employing systematic or local search methods to find solutions that satisfy all the constraints [49].

Constraint Modeling

After the SERENA modeling and risk assessments for the smartphone, the next step is to translate the SERENA model into constraint code to objectively quantify the system's security level. The modeling approach, following Sawyer et al. [47], uses mathematical expressions or logical rules to represent the security requirements, constraints, and objectives of the system, describing the relationships

among the system elements such as goals, operationalizations, soft goals, and context variables. We have used the Common Logic Interchange Format (CLIF) and solvers module provided by VariaMos [50] for automatic constraint code generation.

A constraint program, represented as a triple (X, D, C), consists of variables (X), domains (D), and constraints (C) that define feasible variable values. Typically, these programs use finite domains. In a constraint program, the impact on a softgoal by an operationalization is denoted using integers from 0 (- -) to 4 (+ +), while Boolean elements are represented by 0 and 1.

Transformation Rules

— "MainGoal" is guaranteed to be fulfilled (set to a value of 1). It also confirms that the aggregate of the subgoals aids in the attainment of the main goal. The formula determines the overall input of the subgoals in reaching the main goal by multiplying the value of the main goal by the count of subgoals and then comparing this product to the total of all subgoal values. The equation stipulates that the main goal is accomplished only if the sum of the subgoals together supports its realization. Each subgoal plays a part in the total success of the main goal, and their collective contributions should satisfy the main goal's criteria.

$$\text{MainGoal} = 1 \tag{2.3}$$

$$\text{MainGoal} \times n = \sum_{i=1}^{n} \text{SubGoal}_i \tag{2.4}$$

— The group cardinality dependency between the goal *Send Data* and the operationalizations *5G* and *Bluetooth* is translated into the following constraint:

$$SendData = 5G + Bluetooth \tag{2.5}$$

This implies that if *Send Data* is satisfied (equals 1), exactly one (XOR) of *Bluetooth* or *WiFi* must also be satisfied, and vice versa.

$$\begin{aligned} \text{SendData} \in \{0,1\} \wedge 5\text{G} \in \{0,1\} \wedge \text{Bluetooth} \in \{0,1\} \\ \wedge (\text{SendData} = 1 \Leftrightarrow 5\text{G} + \text{Bluetooth} = 1) \end{aligned} \tag{2.6}$$

A more general representation would be:

$$\text{Goal} = \text{Op}_1 + \text{Op}_2 + \ldots + \text{Op}_n \tag{2.7}$$

This implies that if Goal is satisfied (equals 1), exactly one of the operationalizations Op_i must also be satisfied, and vice versa. We can generalize this constraint as follows:

$$\text{Goal} \in \{0,1\} \land \text{Op}_i \in \{0,1\}\ \forall i \in [1,n] \land (\text{Goal} = 1 \Leftrightarrow \sum_{i=1}^{n} \text{Op}_i = 1) \tag{2.8}$$

For a cardinality constraint with bounds m and n, ensure that the count of satisfied operationalizations $\text{Op}_1, \text{Op}_2, \ldots, \text{Op}_n$ lies between these bounds. Representing this as a goal Goal:

$$m \cdot \text{Goal} \leq \sum_{i=1}^{n} \text{Op}_i \leq n \cdot \text{Goal} \tag{2.9}$$

This implies that if Goal is satisfied (equals 1), the number of satisfied operationalizations must fall within the range $[m, n]$, and vice versa. The generalized equation would be:

$$\text{Goal} \in \{0,1\} \land \text{Op}_i \in \{0,1\}\ \forall i \in [1,n] \land m \cdot \text{Goal} \leq \sum_{i=1}^{n} \text{Op}_i \leq n \cdot \text{Goal} \tag{2.10}$$

— The claims are represented as variables that can be satisfied or not by the user as input values. For example, in claim $C5$, a value of 4 indicates that 5G fully satisfies (++) Immunity, while Bluetooth is indifferent to (=) Immunity, so it has a value of 2. This relationship is represented by:

$$C5 \iff (Bluetooth \Rightarrow Immunity = 2) \land (5G \Rightarrow Immunity = 4) \tag{2.11}$$

When multiple operationalizations affect a soft goal, the equation can be expanded to include all. Using concise mathematical notation, let C denote the claim, O_i the ith operationalization, SG the soft goal, and v_i the impact value

of each operationalization. The equation is then given by:

$$C \iff \bigwedge_{i=1}^{n} (O_i \Rightarrow SG = v_i) \tag{2.12}$$

— Soft influences are modeled as Boolean variables. For instance, S3 indicates that with low location sensitivity, Confidentiality, and Integrity must be met (+), whereas Availability, Access Control, and Immunity remain undefined(=):

$$S3 \iff (LocationSensitivity = Low \Rightarrow (Confidentiality = 3 \wedge Integrity = 3 \wedge Availability = 2 \wedge AccessControl = 2 \wedge Immunity = 2)) \tag{2.13}$$

Generalizing the equation, denote soft influence by S, context variable by C, its specific value by c, and soft goals with values as (SG_i, v_i). The optimized equation is:

$$S \iff (C = c \Rightarrow \bigwedge_{i=1}^{n} (SG_i = v_i)) \tag{2.14}$$

Once transformed into a constraint program, the Minizinc model can be augmented to represent properties more easily than in its original form.

TotalClaims represents the total number of claims, summarized by the following equation:

$$\text{TotalClaims} = \sum_{i=1}^{n} C_i \tag{2.15}$$

Let SC_i represent the security criteria variables for $i = 1$ to the total number of criteria.

$$\text{TotalSecurityCriteria} = \sum_{i=1}^{n} SC_i \tag{2.16}$$

The equation for the total number of soft influences is:

$$\text{Total Soft Influences} = \sum_{i=1}^{n} SI_i \tag{2.17}$$

We aim to maximize a weighted combination of TotalClaims, TotalSoftInfluences, and TotalSecurityCriteria, with weights w_1, w_2, and w_3 respectively, indi-

cating the importance of each variable in the optimization.

$$\text{maximize} \quad w_1 \times \text{TotalClaims} + w_2 \times \text{TotalSoftInfluences} + w_3 \times \text{TotalSecurityCriteria} \quad (2.18)$$

By applying the transformation rules to the SERENA overall smartphone model modeled in Figure 2.11, the following MiniZinc[6] constraint model is achieved. By running the code and entering the values for the context variables, the best operationalizations are determined (TRUE) and the security score is given by the variable $TotS$.

```
var 0..1:Communicate; var 0..1:SendData; var 0..1:ReceiveData;
    var 0..1:StoreData;var bool:BlueTooth;var bool:FiveG;var
    bool:eMMC;var bool:SD;
var 0..4:Confidentiality;var 0..4:Integrity;var 0..4:
    Availability;var 0..4:AccessControl;var 0..4:Immunity;
var bool:C1;var bool:C2;var bool:C3;var bool:C4;var bool:C5;var
     bool:C6;var bool:C7;var bool:C8;var bool:C9;var bool:C10;
enum Sensitivity={High,Medium,Low};
 Sensitivity:LocationSensitivity;
enum State={Secure,Public,Offline};
 State:NetworkConnectivity;
var bool:SI1;var bool:SI2;var bool:SI3;var bool:SI4;var bool:
    SI5;var bool:SI6;
var int:TotS;var int:TotC;var int:TotSI;var int:goal;

constraint Communicate=1;
constraint Communicate*3=SendData+ReceiveData+StoreData;
constraint SendData= FiveG+BlueTooth;
constraint ReceiveData= FiveG+BlueTooth;
constraint StoreData= eMMC+SD;
constraint C1 <-> ((FiveG -> Confidentiality = 4)/\(BlueTooth
    -> Confidentiality =4));
constraint C2 <-> ((FiveG -> Integrity = 4)/\(BlueTooth ->
    Integrity =4));
```

6. **https://www.minizinc.org/**.

```
constraint C3 <-> ((FiveG -> Availability = 3)/\(BlueTooth ->
    Availability =2 ));
constraint C4 <-> ((FiveG -> AccessControl = 4)/\ (BlueTooth ->
     (AccessControl=4)));
constraint C5 <-> ((FiveG -> Immunity=4)/\ (BlueTooth ->
    Immunity=2));
constraint C6 <-> ((eMMC -> Confidentiality = 4)/\(SD ->
    Confidentiality =4));
constraint C7 <-> ((eMMC -> Integrity = 4)/\(SD -> Integrity
    =4));
constraint C8 <-> ((eMMC -> Availability = 1)/\(SD ->
    Availability =3 ));
constraint C9 <-> ((eMMC -> AccessControl = 3)/\ (SD -> (
    AccessControl=4)));
constraint C10 <-> ((eMMC -> Immunity=4)/\ (SD -> Immunity=4));

TotC=C1+C2+C3+C4+C5+C6+C7+C8+C9+C10;
TotS=Confidentiality+Integrity+Availability+AccessControl+
    Immunity;

constraint SI1 <-> ((LocationSensitivity=Low)-> (
    Confidentiality>=3 /\ Integrity>=3 /\ Availability>=2 /\
    AccessControl>=2 /\ Immunity>=2));
constraint SI2 <-> ((LocationSensitivity=Medium)-> (
    Confidentiality>=4 /\ Integrity>=3 /\ Availability>=3 /\
    AccessControl>=3 /\ Immunity>=4));
constraint SI3 <-> ((LocationSensitivity=High)-> (
    Confidentiality>=4 /\ Integrity>=4 /\ Availability>=4 /\
    AccessControl>=3 /\ Immunity>=4));
constraint SI4 <-> ((NetworkConnectivity=Secure)-> (
    Confidentiality>=3 /\ Integrity>=2 /\ Availability>=3 /\
    AccessControl>=3 /\ Immunity>=3));
constraint SI5 <-> ((NetworkConnectivity=Public)-> (
    Confidentiality>=4 /\ Integrity>=4 /\ Availability>=4 /\
    AccessControl>=4 /\ Immunity>=4));
constraint SI6 <-> ((NetworkConnectivity=Offline)-> (
```

```
    Confidentiality>=2 /\ Integrity>=2 /\ Availability>=2 /\
    AccessControl>=2 /\ Immunity>=3));
TotSI=SI1+SI2+SI3+SI4+SI5+SI6;
goal=(1000*TotC)+(100*TotSI)+(100*TotS);
solve maximize goal;
```

Security Analysis We evaluate different values of Network Connectivity and Location Sensitivity to identify the most secure system state, security criterion levels, unattainable security levels, and operationalizations for each case. Top-tier availability (4) was not reached for public networks featuring high location sensitivity. Table 2.16 demonstrates that 5G and SD implementations are the most secure across all scenarios, making other options redundant.

Location Sensitivity	**Network Connectivity**	**Security Score**	**Confidentiality**	**Integrity**	**Availability**	**Access Control**	**Immunity**	**Operationalizations**
High	Secure	19	4	4	3	4	4	5G, SD
High	Public	19	4	4	3	4	4	5G, SD
High	Offline	18	4	4	3	4	3	5G, SD
Medium	Secure	19	4	4	3	4	4	5G, SD
Medium	Public	19	4	4	3	4	4	5G, SD
Medium	Offline	18	4	4	3	4	3	5G, SD
Low	Secure	19	4	4	3	4	4	5G, SD
Low	Public	19	4	4	3	4	4	5G, SD
Low	Offline	19	4	4	3	4	4	5G, SD

Table 2.16 – Combined values of Location Sensitivity and Network Connectivity with Security Score

2.2 Conclusion

This chapter offers a detailed examination of our suggested method, aimed at systematically improving the security of system architectures. By dividing the process into three clear phases—specification, formalization, and analysis— we have presented a thorough framework that assists engineers in incorporating security features into system design from the initial phases of development.

In the Security Requirements Specification phase, engineers define security requirements clearly and precisely using components like SECRET and SCORE. These components help to decompose the requirements into distinct parts, ensuring thoroughness. The integration of these components into VariaMos has resulted in a versatile module that accommodates various domains and applications, enhancing the flexibility and adaptability of the specification process.

Moving on to the Formalization of Requirements phase, we ensure that the textual security requirements defined in the specification module are translated into formal models using the SERENA language. This formalization process not only ensures consistency between textual specifications and formal models but also facilitates verification and validation of security properties. By incorporating security criteria from the SCORE ontology, we enrich the modeling process with specific details on associated security mechanisms, further enhancing the robustness of our approach.

Finally, in the Simulation and Analysis of Models phase, we transform formal models into Constraint Programs (CP) for detailed security analysis using solvers. These solvers enable engineers to validate properties, assess security mechanisms, identify vulnerabilities, and ensure compliance with requirements. Contextual variables guide the selection of optimal operationalizations, allowing the analysis to adapt to system conditions and produce relevant results. This approach not only provides a systematic method for evaluating system robustness but also offers a qualitative security metric, enabling engineers to proactively identify and mitigate security flaws.

In conclusion, our proposed approach equips engineers with the tools, methodologies, and frameworks needed to build resilient and secure systems in today's dynamic technological landscape. By providing a structured methodology for integrating security measures into system design, we empower engineers to address security concerns from the outset, thereby fostering the development of robust and secure systems.

Chapter 3

Security Requirements Specification

Security requirements specification is crucial in system development, focusing on defining the security needs and objectives to safeguard assets, data, and operations against various threats and vulnerabilities. This task is complex due to the detailed security issues and the wide range of potential threats. To address these challenges, researchers have proposed various approaches, including using templates [5] and ontologies [16]. A security requirement template offers a structured framework for systematically organizing and documenting security requirements. By providing predefined sections and categories for different security aspects, templates help ensure consistency and completeness in the specification process. In addition, templates serve as a valuable guide for requirements engineers, helping them capture and document security requirements effectively. Security criteria ontologies offer a formal representation of security concepts, mechanisms, and relationships, standardized criteria, and their connections. This enables requirement engineers to apply domain-specific knowledge and best practices in defining security requirements. Furthermore, these ontologies support the reuse and sharing of security knowledge, enhancing consistency and coherence across various projects and domains.

Both templates and ontologies have limitations: templates lack flexibility and adaptability, and ontologies require extensive domain expertise and maintenance effort. Templates may not fully capture the dynamic nature of security requirements, and ontologies might not handle the complexity and ambiguity of real-world scenarios effectively. We propose an integrated framework to address these issues, combining a security requirements template (SECRET) with a security criteria ontology (SCORE). This framework links both approaches to provide a structured, knowledge-rich environment for specifying security requirements. The SECRET template organizes and documents security requirements, while the SCORE ontology adds domain-specific criteria and relationships.

Together, these two components guide users through the security requirements spec-

ification process, offering both structure and knowledge to support informed decision-making and comprehensive coverage of security concerns. By integrating templates and ontologies, our approach seeks to address the limitations of each approach individually while leveraging their respective strengths to enhance the effectiveness and efficiency of security requirements specification in system development projects.

This chapter is structured as follows: Section 2.1.1 discusses the research method and template. Section 3.2.2 covers ontology construction, concepts, and relationships. Section 3.3 elaborates on the SECRET-SCORE approach, integrating the template and ontology. The chapter concludes with a summary.

3.1 SECRET: SECurity REquirements specifica-Tion template

Based on the template proposed by Mazo et al. [6], the SECRET template [51] specifies security requirements for systems and domains, covering security criteria and mechanisms as shown in Figure 2.2. It facilitates a guided process for specifying requirements, distinguishing between problem and solution spaces, and differentiating domain from application requirements.

3.1.1 Research Method

To develop the SECRET template, we used an action research methodology (Figure 3.1), focusing on social intervention to improve situations and derive insights [52][53][54]. This method includes five phases: diagnosis, action planning, action, evaluation, and learning specification [55]. We selected this approach for its ability to tackle current problems, offer practical solutions, and study phenomena in real contexts [56]. Its suitability for our industrial study is based on its support for empirical experimentation and reduction of validity threats. Each cycle involves identifying and defining problems, planning actions, implementing solutions, evaluating outcomes, and learning to begin a new cycle until consensus is reached from the stakeholders. We conducted two action research cycles, each specifying the security requirements for a system.

The first cycle reviewed the security requirements of the Remote Patient Monitoring System [57] and an Electric Vehicles (EV) Charging System [1]. The initial SECRET template for security requirements in semi-structured Natural Language was introduced.

1. Electric Vehicles Charging System

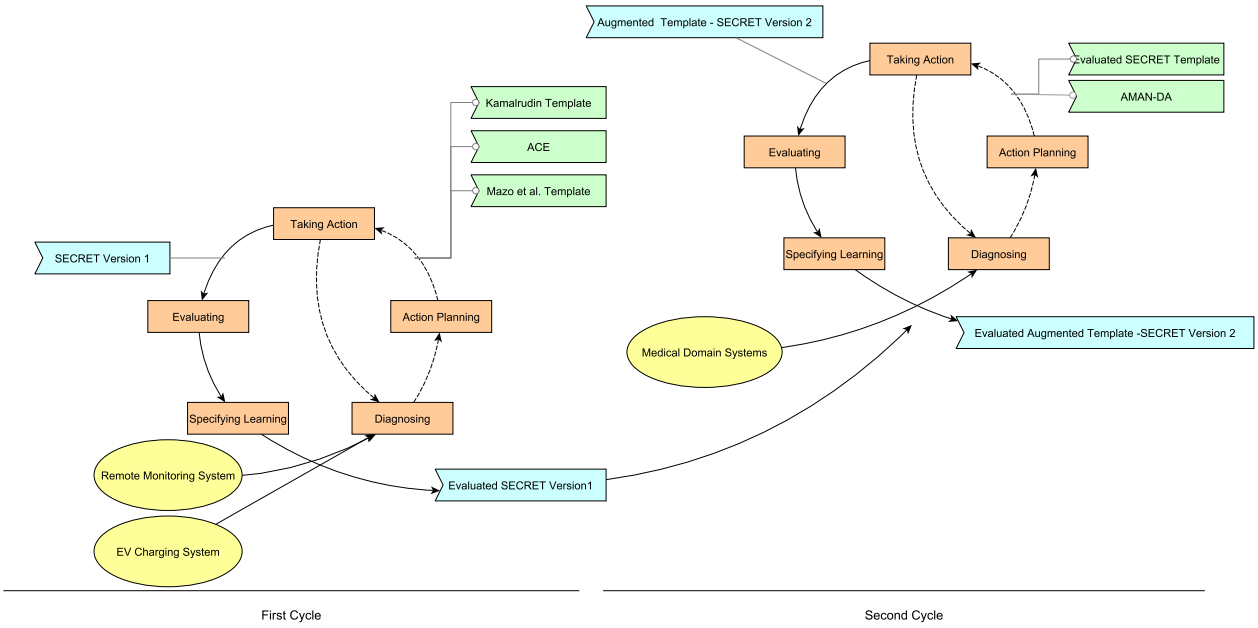


Figure 3.1 – Action Research - SECRET

Three solutions were considered: prose specifications [57] using Mazo *et al.* templates, ACE, and Kamalrudin *et al.* templates [2] [3].

In the second cycle, they examined the requirements specification of Medical Domain Systems or Devices proposed by the ANSM (Agence nationale de sécurité du médicament et des produits de santé) [4]. Three potential solutions were explored: the prose-style security requirements specification, the security requirements specification using the first version of the SECRET template, and the security requirements specification using the improved SECRET template (version 2) [5].

3.1.2 Proposed Template - SECRET

Through two action research cycles, we refined Mazo *et al.*'s template [6], producing two versions of the SECRET template. The first version [6], consisted of 11 parts. Parts 1-4 matched Mazo *et al.*'s sections 1-4; part 5 covered security criteria; part 6, the object part; parts 7 and 8, object conditionality and details; part 9, the security mechanism;

2. Remote Patient Monitoring System Requirements
3. Electric Vehicles (EV) Charging System Requirements
4. ANSM'S GUIDELINE Cybersecurity of medical devices
5. Remote Medical Device Domain Requirements
6. SECRET V1 : https://shorturl.at/bAVX5

and parts 10 and 11 aligned with Mazo *et al.*'s parts 8 and 9. The second version, shown in Figure 2.2 [7], addressed the lack of separation between domain and application security requirements, problem and solution spaces, and the representation of standards and norms. It was organized into three parts: domain and application specifications in both problem and solution spaces, with part 8 detailing relevant norms and standards.

1-Conditions under which a behavior occurs. This component (1 - yellow) is identical in domain and application requirements. The options are:

— **(a) Logical Conditions:** Behaviors occur when a specific condition or event is met.
IF *<condition or event>* ***THEN*** .

— **(b) State-Guided Requirements:** Behaviors required while in a certain state.
WHILE | ***DURING*** *<activation state>* .

— **(c) Optional Elements:** Behaviors depend on the inclusion of a specific feature.
IN CASE *<included feature>* ***IS INCLUDED*** .

— **(d) Temporary Conditions:** Behaviors that follow another behavior.
AFTER | ***BEFORE*** | ***AS SOON AS*** *<behavior>* .

— **(e) Complex Conditions:** Use keywords like When, While, or Where to specify complex behaviors.

2-Family of systems, systems or parts of a system. This component varies by domain and application requirements.

— **System or part of system:** This is represented by part (2 - gray) in Figure 2.2, indicating a specific application requirement. The notation for the system or its part is:
THE *<system or part name>* .

— **Family of systems:** It is represented by part (2 - gray) in Figure 2.2, specifying domain requirements. Thus, the following notations are used:

- (a) Name of all or some systems of a product line.
ALL SYSTEMS OF THE | ***SOME SYSTEMS OF THE*** *<product line name>* .
- (b) Name of the product line and condition or restriction of some product line systems.
THOSE ***SYSTEMS OF THE*** *<product line name> <restriction>* .

7. SECRET V2: https://shorturl.at/antSV

3-The degree of priority. It is the component (3 - orange) of Figure 2.2.

— **(a) Essential requirements:** necessary for product success. **SHALL** .

— **(b) Recommended requirements:** optional for product success. **SHOULD** .

— **(c) Desirable requirements:** used to enhance user experience and satisfaction. **COULD** .

— **(d)** We added a fourth priority, **will**, for optional requirements at the discretion of the technical team. **WILL** .

4-The activity It is component 4 (green) in Figure 2.2, comprising three activity types:

— **(a) Autonomous activity:** no user involvement, with the system(s) operating autonomously from start to finish.

<process verb> .

— **(b) User interaction:** the system(s) enable a user to initiate or complete specific actions.

PROVIDE *<who?>* ***WITH THE ABILITY TO*** *<process verb>* .

— **(c) Interface requirement:** the system performs a behavior dependent on another entity.

BE ABLE TO *<process verb>*

- **(i) If the external system executes the behavior:**

  FROM *<system or external device name>*

- **(ii) If the behavior is executed by the system and interacts with another system or external device:**

  TOWARDS *<system or external device name>* .

5-Security Criteria This component (5 - black) in Figure 2.2 denotes the security goals required. *<Security Criteria of>* . An example of a security criterion is Authentication. <The Remote Monitoring System>$_{system}$<SHALL>$_{priority}$<guarantee >$_{processVerb}$< **authentication of**>$_{securityCriteria}$< person or device>$_{object}$< with whom it is interacting>$_{additionalObjectDetails}$<by identity verification >$_{securityMechanism}$.

6-Object or Objects The object(s) or asset(s) are represented by the available ranges:

— **(a) Single object.** ***ONE*** *<object>* .

— **(b) A specific object.** ***THE*** *<object>* .

— **(c) Each object of a set.** ***EACH*** *<Object>* .
— **(d) Multiple objects.** *<X> <objects>* , where X is the number of objects.
— **(e) Range of objects.** ***BETWEEN*** *<A>*
— ***AND*** *<B> <objects>* . **(f) All objects in a set.** ***ALL THE*** *<objects>*

7-Conditionality in the object Represented by part (9 - yellow) in Figure 2.2 with the notation: ***IFF*** *<condition>* .

8-The complementary details Represented by component (10 - blue) in Figure 2.2 with notation: *<additional object details>* .

9-Security Mechanism The security mechanism, integral to the solution, specifies the methods for meeting security criteria. Represented by part (7 - orange) in Figure 2.2, it uses the notation *<Security Mechanism>* . For instance, identity verification is a security mechanism for authentication. <The Remote Monitoring System>$_{system}$<SHALL>$_{priority}$< guarantee
>$_{processVerb}$<the authentication>$_{securityCriteria}$<of person or device>$_{object}$<with whom it is
interacting>$_{additionalObjectDetails}$<**by identity verification**>$_{securityMechanism}$.

10-Validation Criterion or Standard Security requirements must comply with specific norms or standards, represented as part (8 - purple) in Figure 2.2. The notation used is: ***COMPLYING WITH*** *<Validation Criteria>* .

11-Relax requirements statements for self-adaptive systems Represented by component (11 - red) in Figure 2.2.

— (a) Should adjust the occurrence of objects to maximize or minimize their number.
AS MANY | ***AS FEW*** *<object>* ***AS POSSIBLE*** .
— (b) Achievable before, during, or after a specific event.
BEFORE | ***AFTER*** | ***DURING*** *<event>* .
— (c) Describes timing priorities.
AS EARLY AS POSSIBLE | ***AS LATE AS POSSIBLE*** .
— (d) Sustain until the event.
UNTIL *<event>* .
— (e) Must be sustained for a specific duration.
WITHIN *<time interval> <time unit>* .

— (f) Should reach a minimum frequency or time indefinitely.

AT LEAST <*quantity*> <*frequency*> .

— (g) Behavior that eventually occurs.

EVENTUALLY .

— (h) Indicates flexible, recurring events or quantities.

AS CLOSE AS POSSIBLE TO <*quantity*> <*frequency*> .

Traceability

Meta-data The metadata of security requirements was sourced from the ISO/IEC/IEEE 29148:2011 standard. This standard offers guidance on requirements engineering and management for activities under ISO/IEC 12207 and ISO/IEC 15288, detailing applicable information elements and their content.

— **Identification:** A unique identifier for the requirement
— **Version:** The current version of the requirement (to trace evolution)
— **Creation Date:** Date when the requirement was first created (to trace the requirement over time).
— **Update Date:** Date of the last update of the requirement (to trace the requirement over time)
— **Delivery Date:** The date or deadline for delivering the requirement (functionality).
— **Difficulty:** Difficulty in achieving the requirement
— **Stakeholder - Priority:** Priority of the requirement for the stakeholder.
— **Stakeholder Source:** The stakeholder who is the source of the requirement (traceability to the stakeholder).
— **Reference Document:** The document in which the source or explanation of the requirement is found.
— **Risk:** Consequences or risk of avoidance.
— **Constraints:** Constraints affecting the system's lifecycle of interest, such as cost, schedule, political and operational aspects.
— **Rationale:** The justification to establish each requirement must be captured. The rationale provides the reason why the requirement is necessary and refers to any analysis, business study, modeling, simulation, or other substantial objective evidence, simulation, or any other substantial objective evidence.
— **Applicability:** If the requirement is applicable in reality or not.
— **Component Name:** Name of the component affected by the requirement.

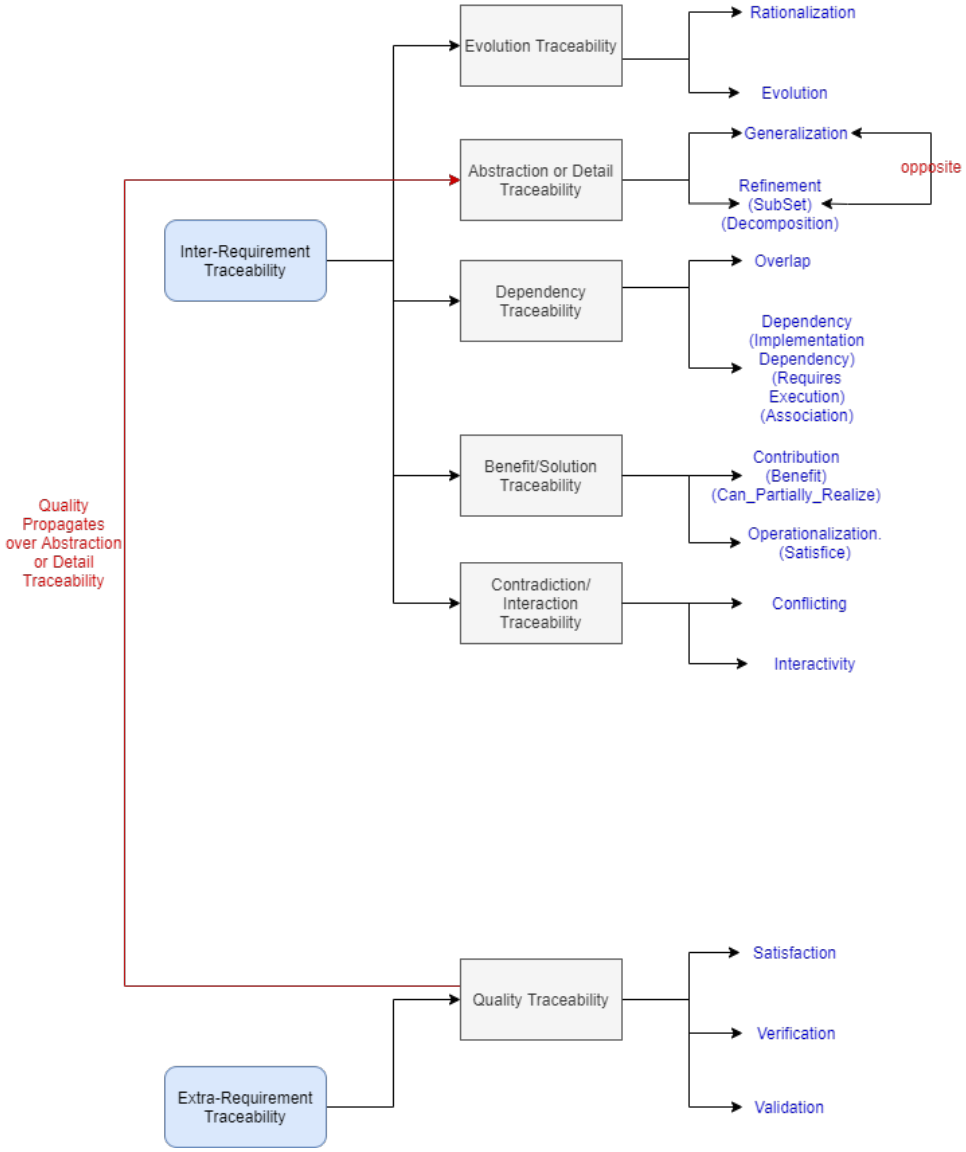


Figure 3.2 – Traceability Relationships

Requirements Traceability Relationships Based on studies of Großer et al. [58], Kassab et al. [59], and Spanoudakis et al. [60], we identified two traceability types (Figure 3.2): inter-requirements and extra-requirements.

— **Inter-requirements Traceability:** Refers to relationships between requirements.

- **Evolution Traceability:**
  — Rationalization: Represent the rationalization of objects or document the reasons for evolutionary steps. $R1 \rightarrow R2$
  — Evolution: Document the input-output relationships of actions leading from

one or more existing objects to one or more new or modified objects. $R1 > R2$

- **Abstractions or Details Traceability:**
  - Refinement: Becomes more specific in details and is indeed more demanding than the original requirement. $R1 \triangleright R2$
  - Generalization: Becomes more general or more abstract in detail, then is indeed more demanding than the original requirement might have been. $R1 \triangleleft R2$
- **Dependencies Traceability:**
  - Overlap: Represents a superposition relationship between terms in a requirement statement or use case and elements like a class, attribute, or association in the analysis model. This overlap indicates shared features of the system or domain, suggesting potential dependencies. $R1 \Leftrightarrow R2$
  - Dependency: Manages dependencies between objects at the same development stage, often due to constraints like resources, skills, or compatibility. $R1 \odot R2$
- **Benefits/Solutions Traceability:**
  - Benefit: A requirement's benefit may depend on another's prior implementation. For example, requirement A's benefits may only be realized after implementing requirement C. This concerns recognizing A's value, not its feasibility. $R1 \propto R2$
  - Operationalization: This relationship converts non-functional requirements into solutions for the target system. $R2 = f(R1)$
- **Conflicts/Interactions Traceability:**
  - Conflicts: The development and modification of requirements and concepts often generate conflicts due to varying interpretations, assumptions, interests, and objectives of stakeholders. $R1 \oplus R2$
  - Interactivity: Non-functional requirements, being static goals, do not interact directly. However, their links with features can cause interactions where fulfilling one non-functional requirement might hinder or help others, such as security and performance in the command feature. $R2 = R2 + I(R1)$

— **Extra-requirements Traceability:** Refers to relationships between requirements and other artifacts.

- **Quality Traceability:**
  - Satisfaction: An element e1 satisfies e2 if it meets e2's expectations, needs, desires, or fulfills a condition e2 represents. Requirements are satisfied by artifacts if they meet relevant conditions. Depending on their refinement relationship, a requirement may also help satisfy higher-level requirements or goals.

— Verification: Test plans, specifications, and reports documenting requirement results.
— Validation: Validation ensures specifications meet user needs and the system satisfies requirements. It provides evidence that all requirements are fully implemented and traceable to system requirements.

3.2 SCORE: Security Criteria Ontology for security Requirements Engineering

SCORE is a comprehensive ontology focused on security criteria and their mechanisms. The ontology helps specify complementary security requirements by linking criteria and improving security coverage. In addition, it assists in selecting suitable security mechanisms for specific requirements in a given domain. Figure 2.3 illustrates the core ontology.

3.2.1 Research Method

Drawing inspiration from Souag et al. [16], the SCORE methodology for ontology development encompasses six phases: objective, scope, knowledge acquisition, conceptualization, implementation, and validation. The objective phase details the ontology's intended uses, scenarios, and target audience. The scope phase specifies the domain. Data are collected from diverse sources in the knowledge acquisition phase, using a literature review to pinpoint pertinent security criteria. The conceptualization phase employs standards and norms to establish security criteria and their interrelations, supported by Constructive Grounded Theory, and arranges this information into a conceptual model. The implementation phase involves using tools such as Protégé [8] to encode the ontology in RDF or OWL/XML formats. The validation phase confirms the ontology's alignment with its intended representations, including expert assessments to verify completeness and accuracy. The ontology is deployed in a web interface [9] and its usability is tested to evaluate user feedback as detailed in Chapter 5.

8. Protégé: https://protege.stanford.edu/
9. http://193.52.45.42:8787/

3.2.2 Proposed Ontology - SCORE

The SCORE ontology (Figure 3.3) encompasses a range of cybersecurity concepts and their interconnections, demonstrating how security criteria are linked. Included concepts are Reputation, Trust, Reliability, Accountability, Authorization, Access control, Authentication, Confidentiality, Integrity, Availability, Freshness, Resilience, Privacy, Physical Security, Logging, Identification, Non-repudiation, Anonymity, Flexibility, Maintainability, Audit, Fault tolerance, Data verification, Location privacy, Traceability, Revocability, Error detection, Liability identification, Immunity, and Business continuity. Relationships among these concepts were analyzed based on connection levels, leading to the identification of actions like Ensures, TestsFor, Provides, Conserves, ContributesTo, Revokes, IsAppliedBy, and Precedes, which were prioritized into high (Ensures, IsAppliedBy, Precedes), medium (Provides, Conserves, TestsFor), and low (ContributesTo, Revokes) categories. For example, a requirement that belongs to the *healthcare* domain and has an *Authorization* security criterion can have *role-based access control* as a security mechanism and *Authentication* as a supplementary security criterion.

Concepts of the Security Criteria Ontology

The ontology is created using cybersecurity concepts and their relationships. It conveys how security criteria are related in a context-neutral method. The various concepts recognized for this ontology and their corresponding explanations are detailed in the following.

— **Reputation:** "The opinion that people in general have about someone or something, or how much respect or admiration someone or something receives, based on past behaviour or character." (Cambridge Dictionary).
— **Trust:** "The confidence one element has in another that the second element will behave as expected." (NIST SP 800-95 under Trust from Open Grid Services Architecture Glossary of Terms).
— **Reliability:** "The ability of a system or component to function under stated conditions for a specified period of time." (NIST SP 800-160 Vol. 2 Rev. 1 from IEEE Standard Computer Dictionary).
— **Accountability:** "The principle that an individual is entrusted to safeguard and control equipment, keying material, and information and is answerable to proper authority for the loss or misuse of that equipment or information." (CNSSI 4009-2015 from NSA/CSS Manual Number 3-16 (COMSEC)).
— **Authorization:** "The process of verifying that a requested action or service is ap-

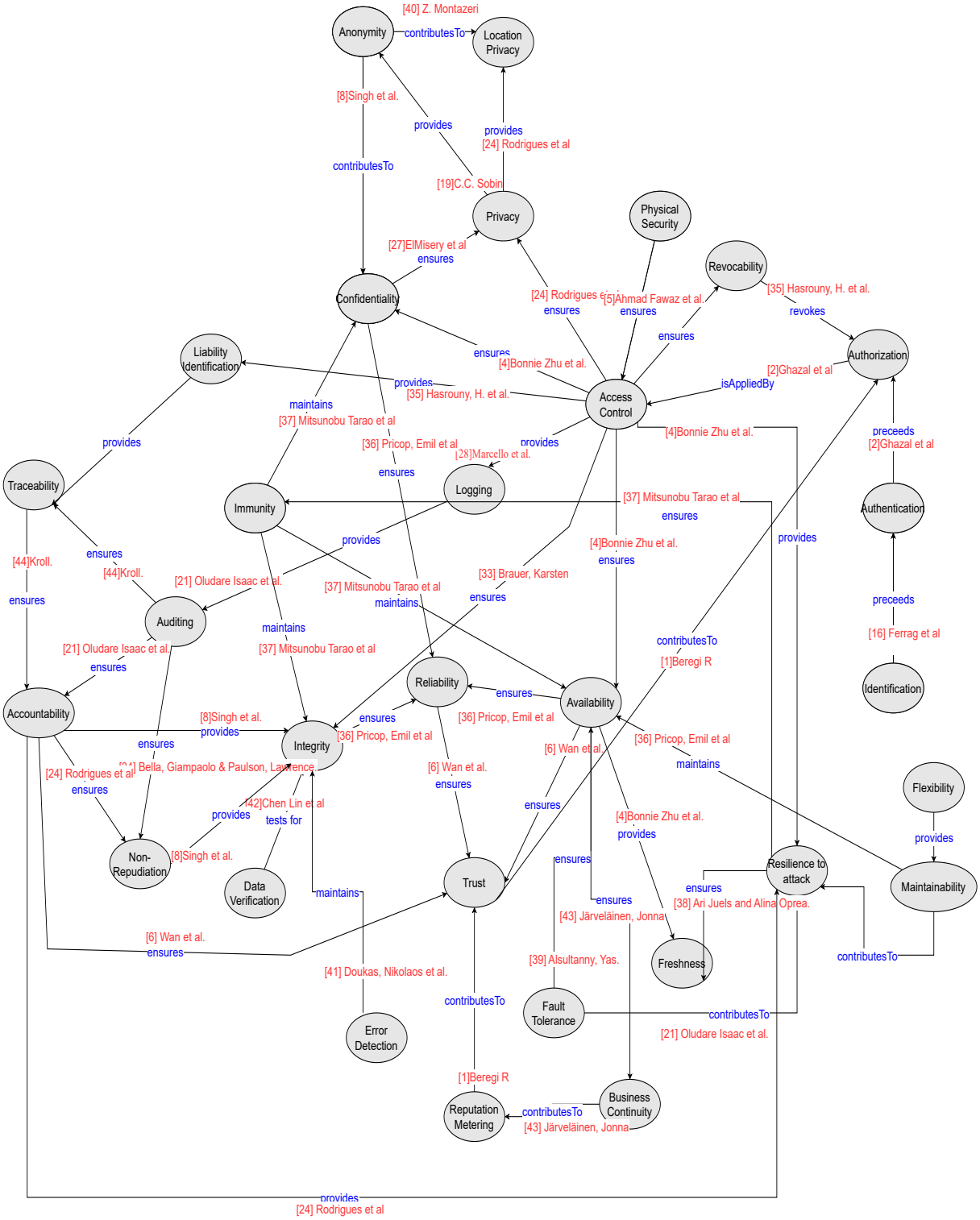


Figure 3.3 – SCORE

proved for a specific entity." (NIST SP 800-152 under Authorization).

— **Access control**: "The process of granting or denying specific requests to 1) obtain and use information and related information processing services and 2) enter specific physical facilities (e.g., federal buildings, military establishments, border crossing entrances)." (FIPS 201-3 under Access Control).

— **Authentication:** "Verifying the identity of a user, process, or device, often as a prerequisite to allowing access to resources in an information system." (NIST SP 800-82 Rev. 2 under Authentication).

— **Confidentiality:** "The property that data or information is not made available or disclosed to unauthorized persons or processes." (NIST SP 800-66 Rev. 1 under Confidentiality from 45 C.F.R., Sec. 164.304).

— **Integrity:** "Guarding against improper information modification or destruction, and includes ensuring information non-repudiation and authenticity." (NISTIR 7621 Rev. 1 under Integrity).

— **Availability:** "Timely, reliable access to information by authorized entities." (NIST SP 800-57 Part 1 Rev. 5 under Availability)

— **Freshness:** "The quality of being new or original, and usually therefore interesting." (Cambridge Dictionary).

— **Resilience:** "The ability to prepare for and adapt to changing conditions and withstand and recover rapidly from disruption. Resilience includes the ability to withstand and recover from deliberate attacks, accidents, or naturally occurring threats or incidents." (NIST SP 800-160 Vol. 2 Rev. 1 from OMB Circular A-130 (2016)).

— **Privacy:** "Assurance that the confidentiality of, and access to, certain information about an entity is protected." (NIST SP 1800-10B under Privacy from NIST SP 800-130).

— **Physical Security:** "The security discipline concerned with physical measures designed to safeguard personnel; to prevent unauthorized access to equipment, installations, material, and documents; and to safeguard them against espionage, sabotage, damage, and theft." (Center for Development of Security Excellence).

— **Logging:** "Logging in information security refers to the process of collecting and storing information about system and network activity, in order to track and analyze the actions of users, systems, and other entities. Logs can include information such as user logins and logouts, file access, network connections, and system events." (ISO 27001).

— **Identification:** "The process of discovering the identity (i.e., origin or initial history) of a person or item from the entire collection of similar persons or items." (FIPS 201-3

under Identification).

— **Non-repudiation:** "In a general information security context, assurance that the sender of information is provided with proof of delivery, and the recipient is provided with proof of the sender's identity, so neither can later deny having process the information." (NIST SP 800-57 Part 2 Rev.1 under Non-repudiation).

— **Anonymity:** "Condition in identification whereby an entity can be recognized as distinct, without sufficient identity information to establish a link to a known identity." (NISTIR 8053 from ISO/IEC 24760-1:2011).

— **Flexibility:** "The ability to change or be changed easily according to the situation." (Cambridge Dictionary).

— **Maintainability:** "The ease with which a software product can be modified to correct defects, modified to meet new requirements, modified to make future maintenance easier, or adapted to a changed environment." (ISTQB Glossary).

— **Audit:** "Independent review and examination of records and activities to assess the adequacy of system controls, to ensure compliance with established policies and operational procedures." (NIST SP 800-53 Rev. 5 from CNSSI 4009-2015).

— **Fault tolerance:** "A property of a system that allows proper operation even if components fail." (NISTIR 8202).

— **Data verification:** "Data verification includes completeness, self-consistency, but includes protection against alteration and corruption." (ISO/IEC JTC 1/SC 22/OWGV N 0116).

— **Location privacy:** "The concept of location privacy can be defined as the right of individuals to decide how, when, and for which purposes their location information could be released to other parties." (Computer and Information Security Handbook (Second Edition), 2013).

— **Traceability:** "The ability to know the actions an entity has performed by means of recorded information" ([61] in Encyclopedia of Information Science and Technology, Third Edition Chapter 416).

— **Revocability:** "Capable of being revoked; able to be cancelled." (Revocable - Collins Dictionary).

— **Error detection:** "Error detection refers to the detection of errors caused by noise or other impairments during transmission from the transmitter to the receiver. Error detection typically enables reliable delivery of digital data over unreliable communication channels." (Patent US11323548B2).

— **Liability Identification:** "Accountability or user identification during communication. Messages can be used to identify users" [62].

— **Immunity:** "The state of being protected from something" (Oxford Learner's Dictionaries).
— **Business continuity:** "Business continuity is a corporate capability. This capability exists whenever organizations can continue to deliver their products and services at an acceptable predefined capacity within an acceptable time frame whenever business disruptions occur." (ISO 22301 2019).

Relationships of the Security Criteria Ontology

— High Priority
- **ensures:** Make sure to provide something. Both criteria need to be present simultaneously, both are obligatory (if we have criterion CA, we are sure to have criterion CB without any additional information about the existence of criterion CB)

$$\text{ensures}\,(CA, CB) \Leftrightarrow (CA \in S) \wedge (TA = TB) \rightarrow (CB \in S) \tag{3.1}$$

- **isAppliedBy:** Use something for a practical purpose. Both criteria need to be present simultaneously, and both are mandatory (if criterion CA *isAppliedBy* criterion CB, it means that criterion CB implements criterion CA).

$$\text{isAppliedBy}\,(CA, CB) \Leftrightarrow (CA \in S) \wedge (CB \in S) \wedge (TA = TB) \rightarrow \text{implements}\,(CA, CB) \tag{3.2}$$

- **preceeds:** To be or go before something or someone in time or space. Criterion CA is present before criterion CB in time, but both are obligatory. It means criterion CA needs to be applied or implemented before criterion CB.

$$precedes\,(CA, CB) \Leftrightarrow (CA \in S) \wedge (CB \in S) \wedge (TA < TB) \wedge (TB - TA + DTA > 0) \tag{3.3}$$

— Medium Priority
- **provides:** Make something available for use. Both criteria may be present simultaneously. The CB criterion can be achieved by the CA if accompanied by some additional information, but the CA criterion does not directly cause the CB criterion (CA + additional information implies CB).

$$provides(CA, CB) \Leftrightarrow (CA \in S) \wedge (TA = TB) \wedge (CA + additional\ information \rightarrow (CB \in S)) \tag{3.4}$$

- **testFor:** Use it to determine whether something is working correctly or how effective it is. Both criteria may be present at the same time. It is highly recommended to have the CA criterion for the sake of the CB criterion. If criterion CA *testFor* criterion CB means that criterion CA tests or verifies that criterion CB is achieved.

$$testFor(CA, CB) \Leftrightarrow (CA \in S) \wedge (TA = TB) \rightarrow verify(CA, CB) \quad (3.5)$$

- **conserves:** To keep and protect something from damage, change, or waste. Both criteria shall be present simultaneously. It is highly recommended to have the CA criterion to protect the CB criterion in a threatening event. Criterion CA conserves criterion CB while another event happens (not all the time).

$$conserves(CA, CB) \Leftrightarrow (CA \in S) \wedge (CB \in S) \wedge (TA = TB) \wedge EventHappensAt(T) \rightarrow protects(CA, CB, DT) \quad (3.6)$$

— Low Priority

- **contributesTo:** It is one of the reasons why something happens. There is no obligation for both criteria to be present simultaneously. The CB criterion requires that the CA and other criteria be met. Criterion CA is not enough on its own to achieve criterion CB.

$$contributesTo(CA, CB) \Leftrightarrow (CA \in S) \wedge hasOtherCriteria(S) \rightarrow (CB \in S) \quad (3.7)$$

- **revokes:** To say officially that an agreement, permission, a law, etc. is no longer in effect. The CA and CB criteria are present at the time of the cancellation decision. If criterion CA *revokes* criterion CB, it implies that criterion CA can cancel criterion CB.

$$revokes(CA, CB) \Leftrightarrow (CA \in S) \wedge (CB \in S) \wedge cancels(CA, CB, T) \rightarrow (CB \notin S) \quad (3.8)$$

3.3 SECRET-SCORE Approach

The SECRET-SCORE methodology, outlined in the third step of Figure 3.4, presents a sophisticated approach for defining security requirements using the SCORE ontology. This method effectively merges the SCORE ontology with the existing SECRET frame-

Symbol	Description
C	security criteria
S	system for which the security criteria are defined
CA	security criterion A
CB	security criterion B
TA	execution start time of CA
TB	execution start time of CB
DTA	execution duration of CA
T	the moment (starting time) an event happens
DT	the duration(time) an event happens
$implements$	puts the other criterion into effect
$additionalinformation$	information that can suggest how or where a criterion is applied
$verify$	to confirm the existence or application of the other criteria
$EventHappensAt$	an external event, such as an attack, that happens to the system at a specific moment
$protects$	to cover or shield the other criterion during an event and make sure it is not lost
$hasOtherCriteria$	the system has additional criteria that help others (CB) exist
$cancels$	neutralizes or negates the effect of the other criterion

Table 3.1 – Symbols Legend

work, particularly connecting it to the parts focusing on security criteria and mechanisms as shown in Figure 3.5. This integration allows security engineers to apply SCORE insights, improving the Security Criteria and Mechanisms within the SECRET framework, ensuring these elements are well-informed by the ontology.

The SCORE ontology acts as a reference point and introduces additional security criteria, each with specific priority levels. These extra criteria are crucial in broadening the scope of security requirements, serving as important additions to the basic security criteria specified in the SECRET template. They tackle potential security issues that might not have been previously considered, thereby enhancing the overall security strength of the system. Integrating the SCORE ontology with the SECRET template creates a uniform and detailed framework for articulating security requirements. The contributions of the ontology ensure that the security criteria are thorough, non-redundant, and clearly defined. Moreover, the ontology provides a transparent and organized depiction of the connections among different security criteria, effectively reducing confusion and

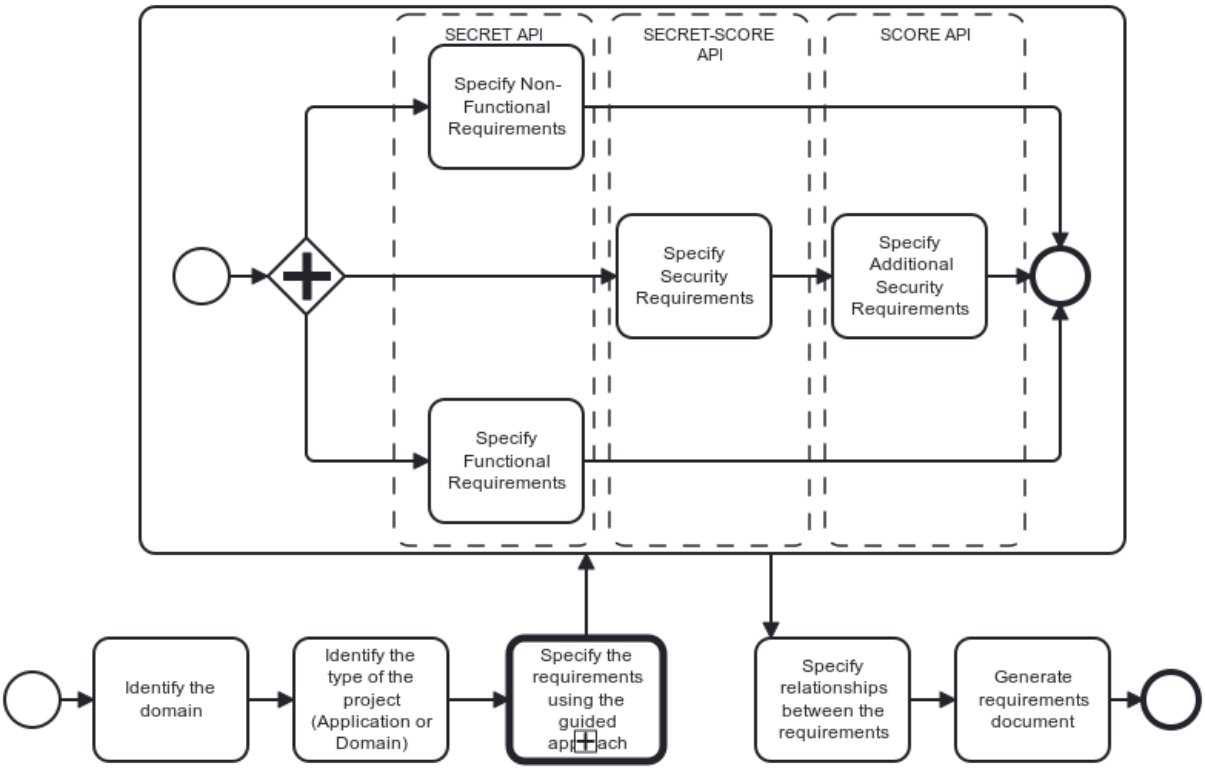


Figure 3.4 – Requirements Specification Process Using SECRET-SCORE in BPMN

inconsistencies during the specification phase.

This unified method enhances the formulation of security requirements, thereby enhancing interactions among requirements engineers and additional stakeholders. It also promotes a more detailed and subtle comprehension of security requirements, crucial in their successful realization during the system design and implementation stages. The SECRET-SCORE approach represents an advanced and sturdy technique that guarantees the accuracy and thoroughness of security requirements.

3.3.1 Tool: VariaMos Requirements Specification Module

The VariaMos [10] tool implements the approach using two languages: Domain Requirements-AC and Application Requirements-AC (AC denotes Auto-Complete). Each language aligns with the application and domain components of the SECRET template, respectively.

10. Variamos

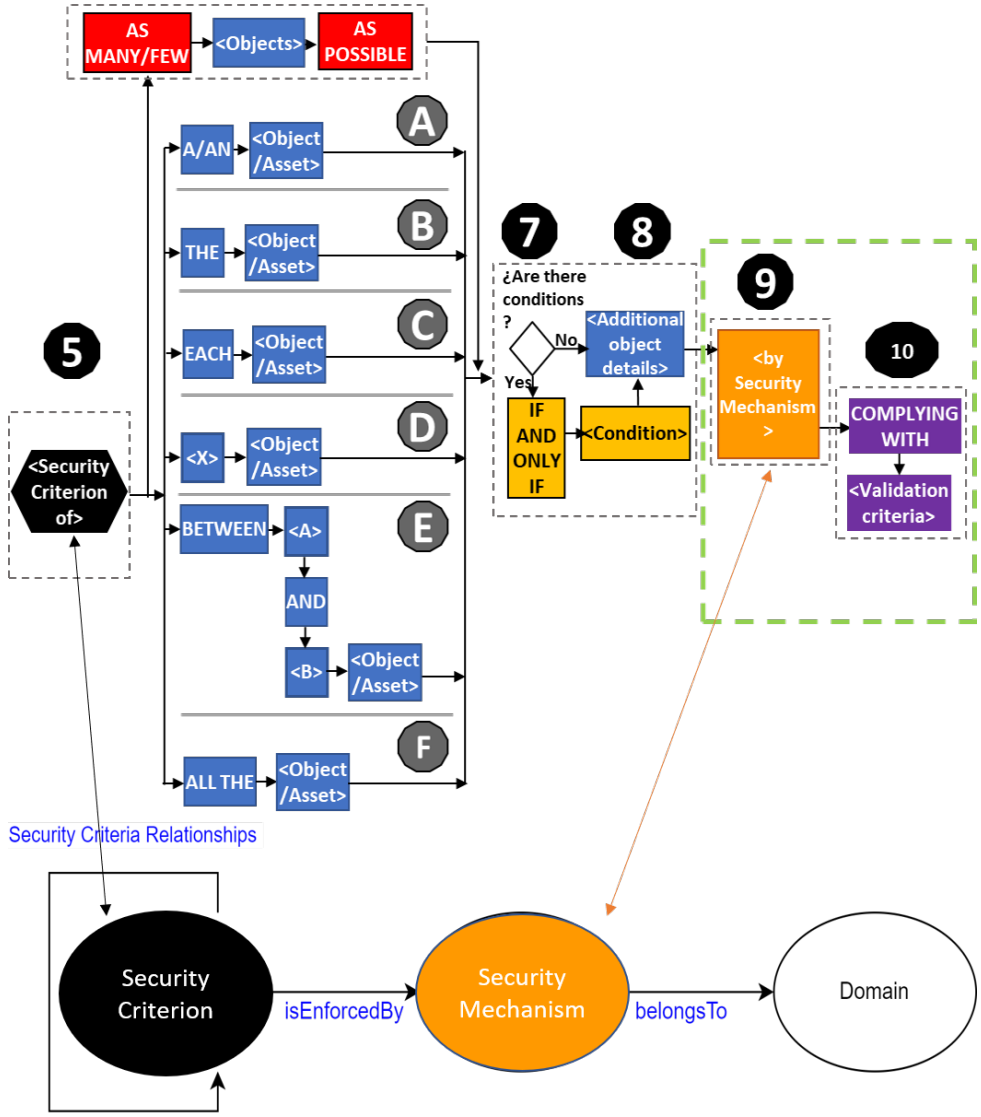


Figure 3.5 – SECRET-SCORE

VariaMos

VariaMos is an extensible web-based software tool designed for engineering and simulating dynamic product lines and variability-intensive systems. It is developed by a collaborative work group dedicated to advancing the field of product line engineering (PLE). VariaMos provides a comprehensive suite of features, combining multiple aspects of PLE into a single, accessible platform.

The key features of VariaMos include:

— **Variability Modeling**: VariaMos lets users create custom variability modeling lan-

guages and detailed models for static and dynamic product lines, efficiently managing complex feature variations.

— **Systems Engineering**: The tool supports product line-driven systems engineering, allowing for model integration, verification, analysis, and the derivation of requirements specifications to ensure robust system design.

— **Simulation**: VariaMos enables users to simulate and test dynamic product lines, aiding in the analysis and verification of complex systems.

— **Extensibility**: VariaMos, being a web-based application, is crafted for easy extensibility, enabling users to add new features or tailor the tool to suit particular project needs.

The key technical details of VariaMos are:

— **Web-Based Platform**:Implemented entirely as a web-based application, VariaMos offers enhanced accessibility and user convenience, eliminating the necessity for local installations.

— **Deployment Evolution**: Originally starting as an Eclipse plug-in integrated with GNU Prolog, the tool has transformed into a cutting-edge web application, enhancing both deployment efficiency and user experience.

— **Collaborative Development**: VariaMos is being developed by a collaborative team that integrates continuous research insights and industry feedback to consistently improve its features and capabilities.

Key use cases and applications include:

— **Product Line Engineering (PLE)**:VariaMos tackles essential PLE activities, managing intricate tasks associated with variability management, configuration, and feature modeling.

— **Dynamic Product Lines**: The tool effectively manages dynamic product lines, crucial for adaptive and evolving modern engineering systems.

— **Research and Industry Applications**:VariaMos is used in academia, enhanced by ongoing research in product line engineering.

Functionalities

The languages incorporate three services, each addressing distinct functionalities:

— **Guided requirements specification:** Domain Requirements-AC and Application Requirements-AC languages enable the specification of ***function, non-functional, and security requirements***. They feature an auto-complete service that suggests the next word based on the template (domain or application) and provides recommendations for security criteria and mechanisms tailored to the selected domain.

This functionality is supported by an API that queries the SCORE ontology in OWL (Web Ontology Language).

— **Additional security requirements suggestion:** Both Domain Requirements-AC and Application Requirements-AC languages can suggest additional security requirements based on the security criteria of a selected requirement.

— **Generation of SRS or SyRS:** Domain Requirements-AC and Application Requirements-AC languages enable the creation of SRS and SyRS documents compliant with ISO/IEC/IEEE29148 [63], detailing functional, non-functional, and security requirements.

Architecture

The Variamos architecture requirements specification module uses four key microservices to improve functionality and modularity, as depicted in Figure 3.6:

— **SCORE Microservice:** Developed in Java Enterprise Edition, this microservice functions as an API for the SCORE ontology. It uses the Apache Jena Library to connect to the SCORE ontology, stored in OWL format, retrieving security criteria and mechanisms.

— **SECRET Microservice:** A NodeJS microservice providing autocomplete for the SECRET template, interacting with the SCORE microservice to access security criteria and mechanisms.

— **Additional Security Requirements Microservice:** Developed in NodeJS, this microservice recommends additional security requirements by connecting to the SCORE microservice to retrieve suggestions based on a specific security criterion.

— **Requirements Document Generation Microservice:** This NodeJS microservice uses the Docx library to generate comprehensive requirements documents.

3.4 Conclusion

This chapter introduces a novel approach to improving the specification of security requirements in technology systems. The proposed approach integrates the SECRET template with the SCORE ontology to provide a structured and comprehensive framework to define security requirements and effectively address potential vulnerabilities. By combining these two elements, the approach aims to improve the security posture of technology systems and enhance their resilience against cyber threats.

The integration of the SECRET template and the SCORE ontology offers several

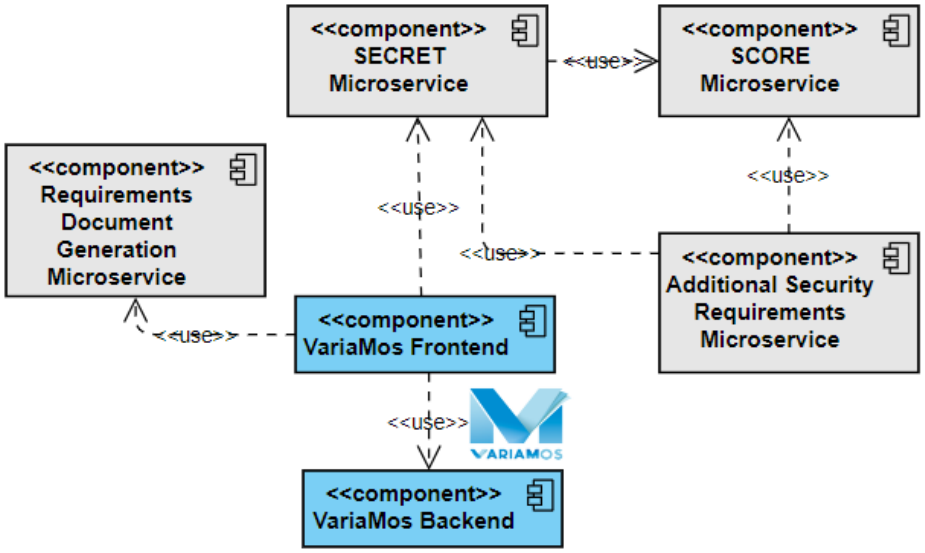


Figure 3.6 – Requirements Module Architecture in UML Component Diagram

advantages in the specification of security requirements. The SECRET template provides a structured format for capturing security requirements, while the SCORE ontology offers a rich vocabulary of security concepts and mechanisms. By linking these two components together, the approach enables requirements engineers to specify security requirements in a more systematic and detailed manner, thereby improving the overall security of technology systems.

A new security requirements language has been developed in the VariaMos online tool to implement this approach. This language provides users with an intuitive and user-friendly platform for specifying security requirements, allowing them to take advantage of the benefits of the SECRET template and the SCORE ontology in a practical and accessible manner.

Chapter 4

Security Requirements Formalization and Analysis

This chapter delves into methodologies and techniques for the modeling and analysis of security within our project. It commences with the rigorous transformation of textual security requirements, as delineated in the preceding requirements module, into structured formal models. We use the SECRET template and the SCORE ontology to formalize textual specifications, thus enabling the systematic design and implementation of secure systems.

The SECRET template provides a structured method for expressing textual requirements. Integrating security criteria from the SCORE ontology adds detailed security criteria and mechanisms to our models. We then convert these models into the SERENA modeling language, strictly following its syntax and semantics. Our methodology ensures a smooth transition from textual to formal security specifications, improving traceability and system understanding. It also integrates security throughout the project lifecycle, leading to more resilient systems.

In our security analysis transition, we shift from using CLIF to Constraint Programming, which allows for rigorous semantic analysis. Within VariaMos, transforming models into the CLIF format is essential for employing solvers in formal analysis, ensuring uniform representation of security properties. The core of our analysis uses Constraint Programming to perform logical computations, verify properties, assess security mechanisms, identify vulnerabilities, and ensure compliance. Our analysis adapts to system conditions, guided by data from context variables, enhancing the relevance and effectiveness of our findings.

This methodology combines formal modeling and Constraint Programming to systematically evaluate and enhance system robustness, ensuring the development of secure systems.

This chapter is organized as follows: Section 4.1 discusses the proposed security requirements formalization approach, SERENA, while detailing the transformation rules

applied. It also discusses the implementation of the modeling language and the transformations in VariaMos. Similarly, the security analysis approach is explained in Section 4.2 with its implementation.

4.1 SERENA: SEcurity REquirements aNAlysis

SERENA is a goal-oriented modeling language designed to specify and analyze security requirements in software systems. Drawing inspiration from methodologies such as Sawyer et al. [47], Secure i* [19], SecureTropos [20], and CORAS [25], SERENA emphasizes formalizing security objectives, semantic analysis, and security by design principles. This approach enables organizations to systematically define and address security objectives, thereby enhancing software security. SERENA comprises five distinct models, each serving a specific objective and representation. With this foundation in place, we proceed to explore SERENA in detail. SERENA's meta-model, depicted in Figure 2.6, provides a comprehensive overview of its components and their relationships. Each model within SERENA serves a specific purpose, ranging from capturing security objectives to specifying security mechanisms and constraints. The meta-model of SERENA encapsulates key concepts. These concepts form the building blocks for expressing security requirements in a structured and formalized manner.

The fundamental elements of the SERENA language, as defined in Chapter 2 Section 2.1.2, include: Goals, assets, operation of goals, vulnerabilities, threats, security mechanisms, claims, soft goals, soft influences, and context variables. The fundamental elements are categorized into four concerns:

— Functional Requirements: Goals, Operationalizations of Goals
— Non-functional Requirements: Softgoals (Security Criteria)
— Security: Asset, Vulnerability, Threats, Security Mechanisms
— Context: Context Variables

4.1.1 SERENA Model Transformations

The SERENA language comprises five models: Criteria, Goal, Risk, Treatment, and Overall. The Security Criteria Model specifies key security criteria; the Goal Model sets broad security objectives; the Risk Model pinpoints potential threats; the Treatment Model suggests risk mitigation strategies; and the Overall Model synthesizes these elements for an overall security perspective. As these models are detailed in Chapter 2 Section 2.1.2, we will focus on the transformation rules between the SECRET template

and the five models.

Key Points of the Transformation The transformation strictly adheres to SERENA's syntactic and semantic conventions. Each SECRET model element is meticulously converted to comply with these rules, ensuring that the formal model aligns with SERENA. The main goal is to accurately present security requirements and integrate them into the SERENA language. The transformation preserves the original meaning of every aspect of the SECRET model. Maintaining a clear conceptual correspondence between the SECRET model elements and their SERENA counterparts is crucial to ensure that the terminology in SERENA reflects the defined concepts in the SECRET model, thus preventing confusion.

General Steps of the Transformation The transformation process adopts a systematic approach, starting with identifying elements like goals and security criteria in the SECRET model. Each element is assessed for accurate representation in SERENA, ensuring a smooth concept transition. Defined transformation rules are applied to ensure correct terminology translation, relationship establishment, and adherence to SERENA's syntax. These rules ensure consistency and accuracy. Rigorous verification follows, validating the SERENA model against security specifications through syntax checks, conceptual alignment, and contradiction resolution, thus maintaining the model's integrity and reliability.

Security Criteria Model

The transformation from the SECRET template to the Security Criteria model in SERENA adheres to a specific rule. Each security criterion (5) of the SECRET template is accurately translated to its SERENA equivalent (soft goal), maintaining the hierarchy to ensure consistency. The transformation also considers relationships and dependencies among criteria to accurately reflect security requirements. This rule links the SECRET template's text to SERENA's formal model, enhancing security modeling and analysis.

$$(SC_{SECRET}, R_{SECRET})_{\text{Security Requirements}} \rightarrow (SC_{SERENA}, R_{SERENA})$$

SC_{SECRET} and R_{SECRET} respectively represent the security criteria and their interrelations in the SECRET template. Similarly, SC_{SERENA} and R_{SERENA} correspond to the soft goals and their interrelations in SERENA's Security Criteria model.

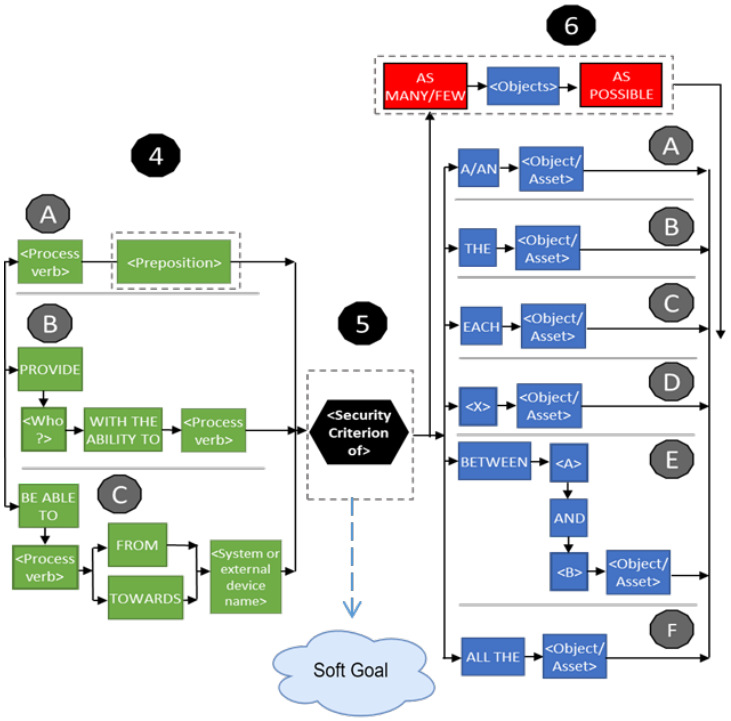


Figure 4.1 – Security Criteria Transformation

SERENA Goal Model

Each functional objective, derived from the activity (4) and object (6) of a SECRET model requirement, is converted into a goal in the SERENA model. Transformation rules connect functional objectives and specific actions, ensuring alignment with the SECRET model's functional specifications.

$$(A_{SECRET}, O_{SECRET}, R_{SECRET})_{\text{Functional Requirements}} \rightarrow G_{SERENA}$$

A_{SECRET} represents the activity, O_{SECRET} the object, and R_{SECRET} the relationships between functional requirements. G_{SERENA} denotes the goals in the SERENA model.

Operationalizations detailing functional requirements are also considered in this model. When a functional requirement from the SECRET model operationalizes a security or other non-functional requirement, these details are incorporated into the SERENA model.

$$\forall FR_i \in FR_{\text{SECRET}} : (\text{operationalizes}(FR_i, NFR_j) \vee \text{operationalizes}(FR_i, SR_k))$$

$$\implies (OD_i \rightarrow Op_{\text{SERENA}})$$

If a functional requirement FR_i in the SECRET model operationalizes either a non-functional requirement NFR_j or a security requirement SR_k, then the associated additional object details OD_i are added as an operationalization in Op_{SERENA}.

SERENA Risk Model

Security criteria from the criteria model are integrated into this model to assess threat impacts on system security.

The operationalizations of functional objectives from the objectives model are used in this transformation to define the context for potential system threats. In this context, the user must specify the vulnerabilities of the operationalizations, the threats, and the level of risk.

$$(SC_{criteria} \in Model_{criteria} \Rightarrow SC_{criteria} \in Model_{risk}) \wedge (Op_{goal} \in Model_{goal}$$

$$\Rightarrow Op_{goal} \in Model_{risk})$$

A security criterion $SC_{criteria}$ from $Model_{criteria}$ is included in $Model_{risk}$, similarly, an operational goal Op_{goal} from $Model_{goal}$ is also included in $Model_{risk}$.

SERENA Treatment Model

For each threat identified in the risk model, it is necessary to determine the corresponding operationalization that operationalizes a specific security requirement.

Relevant security mechanisms (derived from the SECRET model) are identified based on the security requirement associated with the operationalization. Each security mechanism is linked to threats, indicating how it can mitigate the associated threat.

Depending on the priority the security requirement defines, a claim value is assigned to each security mechanism associated with the threat. This reflects the relative importance of each mechanism in protecting against a specific threat. All of these elements are then added to the treatment model.

$$\forall T_i \in T_{RiskModel} : \Big(\exists SR_j \in SR_{SECRET}, Op_{SR_j} \in OR_{SERENA} : \text{operationalizes}(Op_{SR_j}, SR_j)$$

$$\wedge \text{linked}(Op_{SR_j}, T_i) \implies (\exists SM_k \in SM_{SECRET} : \text{mitigates}(SM_k, T_i))$$
$$\implies CV_{SM_k} = f(\text{priority}(SR_j))$$

In this equation, $T_i \in T_{RiskModel}$ denotes a threat, $SR_j \in SR_{SECRET}$ a security requirement, and $Op_{SR_j} \in Op_{SERENA}$ the corresponding operationalization. operationalizes(Op_{SR_j}, SR_j) and linked(Op_{SR_j}, T_i) show that Op_{SR_j} operationalizes SR_j and links to T_i, respectively. $SM_k \in SM_{SECRET}$ is a security mechanism mitigating T_i as indicated by mitigates(SM_k, T_i). CV_{SM_k} is the claim value for SM_k, priority(SR_j) the priority of SR_j, and f(priority(SR_j)) a function assigning claim values based on SR_j's priority.

SERENA Overall Model

The overall model is achieved by federating the goal model with the treatment and risk models and then calculating the security level for the operationalizations. It is also possible to add other softgoals that are not security-related. The functional goals G_i identified in the goal model are integrated into the overall model, thereby preserving the overall vision of the expected system functionalities.

$$\forall G_i \in G_{GoalModel} : G_i \in G_{OverallModel}$$

The operationalizations associated with the functional goals Op_{G_i} are also included in the overall model, providing a detailed representation of how these objectives are achieved.

$$\forall G_i \in G_{GoalModel} : (\exists Op_{G_i} \in Op_{OverallModel} : \text{operationalizes}(Op_{G_i}, G_i))$$

Other "soft" goals from other non-functional requirements are included in the overall model. The specific nature of these "soft" objectives is determined by the type of associated non-functional requirement.

$$\forall NFR_i \in NFR_{SECRET} : \text{type}(NFR_i) \in SG_{OverallModel}$$

Using information about security mechanisms SM_{Op_i} and risk levels RL_{Op_i}, the overall model calculates an overall security value SV_{Op_i} for each operationalization Op_i. This value reflects the overall contribution of each operationalization to achieving the system's security objectives.

$$\forall Op_i \in Op_{OverallModel} : SV_{Op_i} = f(SM_{Op_i}, RL_{Op_i})$$

After the transformation, the user should define the context variables and the softgoals as explained in Chapter 2.

4.1.2 Tool: Implementation in VariaMos

The SERENA modeling language and its transformations were implemented in VariaMos. The detailed implementation is explained in the appendix.

The modeling language is implemented as five models: (i) SERENA Criteria Model (ii) SERENA Goal Model (iii) SERENA Risk Model (iv) SERENA Treatment Model (v) SERENA Overall Model.

Functionalities

— **Transformation from SECRET to SERENA Criteria Model:** Implemented using NodeJS, receives the textual requirements and extracts the security criteria, and add them as softgoals in the criteria model.
— **Transformation from SECRET to SERENA Goal Model:** Extracts the functional goals and their operationalizations and adds them to a new goal model.
— **Transformation from SERENA Goal Model to SERENA Risk Model:** Extracts the operationalizations and softgoals (security criteria) from the goal and criteria models and creates a new risk model.
— **Transformation from SERENA Risk Model to SERENA Treatment Model:** Extracts all the defined elements in the Risk Model and adds the security mechanisms and their values from the textual security requirements.
— **Federation between SERENA Goal, SERENA Treatment, and SERENA Risk Models:** Federates all information from the goals, risk, and treatment models to automatically calculate the security value for each operationalization to each security criterion.

We also made sure to provide backward traceability by saving the ID of the source requirement in each generated element in the transformation.

Architecture

The transformation tool consists of a single microservice utilized by the VariaMos front end, as shown in Figure 4.2.

4.2 Security Analysis

Concerning the security analysis approaches applied in our framework, we adopted the NIST 800-30 Risk Assessment [34] approach in the SERENA Risk Diagram to cal-

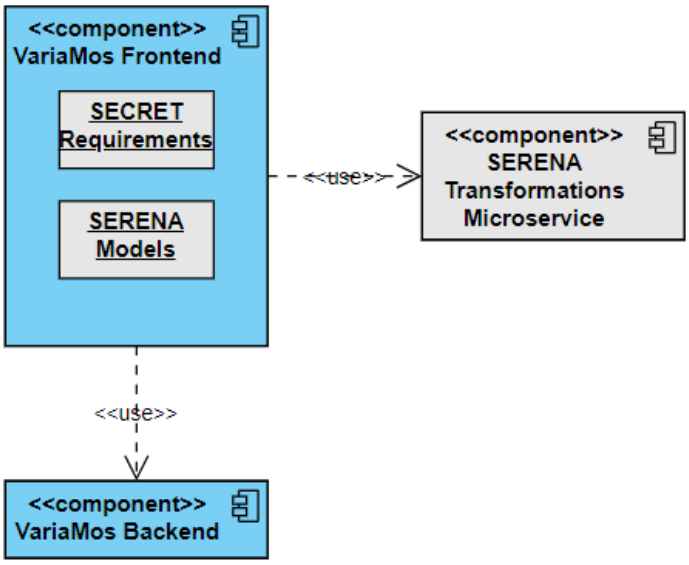


Figure 4.2 – SERENA Transformations toool

culate the risk level of each threat towards the security criteria. The values are automatically translated according to the requirement priority for the treatment evaluation in the SERENA Treatment Model. Otherwise, they are deduced from the MITRE Defense Database [1].

To obtain the best operationalization and an objective security value, we applied the approach of Sawyer et al. [47] to translate the SERENA Overall Model into a constraint program that analyzes and solves the model through solvers.

The entire analysis approach was inspired by the Threat and Hazard Identification and Risk Assessment (THIRA) [39]. THIRA includes identifying threats, evaluating threats, and securing resources. The Cybersecurity and Infrastructure Security Agency explained in their threat and hazard exposure methods [2] the use of context variables for threat analysis.

After transforming the SERENA Overall model into a constraint program, several operations can be applied to solve the model, such as checking the satisfiability of the model. However, our main interest in this thesis is security analysis. We thus focused on enhancing the system's security while choosing the best operationalizations according to the context variables of the system. The operation or request can be represented as following: maximize $w_1 \times \text{TotalClaims} + w_2 \times \text{TotalSoftInfluences} + w_3 \times \text{TotalSecurityCriteria}$

1. https://d3fend.mitre.org/offensive-technique/attack/T1557/
2. https://www.cisa.gov/sites/default/files/publications/Risk%2520Assessment%2520Methodologies.pdf

The details of the constraint program, its operations, and all approaches are given in Chapter 2, Section 2.1.2.

4.2.1 Tool: Using the VariaMos CLIF Module

We used the VariaMos CLIF module proposed by Correa et al. [50] for automatic translation from the VariaMos models (the Overall Model in our case) to constraint programs. The approach requires specifying the semantics of the SERENA Overall Model in a JSON specification form that assigns how each element and relationship in the model is translated into CLIF.

SERENA Semantic Definition

The approach requires a JSON specification of the SERENA Overall Model's semantics, detailing how each model element and relationship translates into the constraint program. An example JSON specification is shown below.

```
"Claim": {
  "param": [
    "C",
    "F",
    "Xs"
  ],
  "constraint": {
    "Claim": "(and (bool C) (iff (= C 1) (forall (x:Xs) (if
        (= x 1) (=< F edge(x)::Value) ) ) ) )"
  },
  "paramMapping": {
    "node": "C",
    "inboundEdges": {
      "var": "Xs",
      "unique": false
    },
    "outboundEdges": {
      "var": "F",
      "unique": true
    }
  }
```

} ,

The JSON snippet shows how a claim is translated into a CLIF constraint. It defines the parameters C, F, and Xs as variables in the constraint. Then, it defines the constraint using these parameters. Finally, paramMapping defines the source of the parameters: Xs are the inbound edges (relationship between operationalizations and the claim) of claim C, and F is the unique outbound edge (relationship between the claim and the softgoal) of claim C.

The model can be translated into CLIF code after defining the semantics of elements and relationships.

Using the CLIF code to get Minizinc model

Once the CLIF code of the SERENA Overall Model is obtained, it can be automatically transformed into constraint models for specific solvers like SWI and Minizinc. In our scenario, we used the Minizinc language as detailed in Section 2.1.2. The generated model lacks the necessary augmentations for conducting security analysis. Therefore, when using the code with the Minizinc solver, it is crucial to incorporate these augmentations, such as the total number of claims, the total security criteria, the total influences, and the method used to solve the model.

To solve the model, we chose to copy the generated Minizinc model to the Minizinc editor and then apply the augmentations. According to the solving equation, the solved Minizinc code gives the best configuration of the system, with the best operationalization and security score. This is detailed in Section 5.2.

4.3 Conclusion

This chapter introduces the proposed security formalization and analysis approach. It automatically transforms structured natural language requirements into a formal model for security analysis. The method ensures comprehensive and systematic security by integrating SERENA models and using constraint programs.

The VariaMos tool was used to transform and model processes, ensuring correct representation and coherence between the SECRET template and SERENA language. This automation reduces human error, enhancing the model's reliability and accuracy.

Formalizing requirements in the SERENA model enables rigorous analysis of security aspects using constraint programs. This includes evaluating security mechanisms, assessing risk impacts, and fulfilling non-functional "soft" goals.

The proposed approach automates the transformation of natural language requirements into a formal model and applies comprehensive security analysis. Leveraging the SERENA framework and VariaMos tool ensures coherent requirement representations, providing a solid foundation for evaluating and enhancing system security. This method streamlines the process and improves the accuracy and reliability of security analysis.

However, an issue to address in SERENA is the scalability and readability of complex diagrams inherited from Sawyer et al. [47]. While the SERENA model aims to provide a clear and structured representation of requirements and goals, its readability significantly diminishes when applied to complex systems. In scenarios involving numerous goals, actors, and relationships, the diagrams are cluttered with too many elements, making it hard for stakeholders to interpret relationships and dependencies between elements.

Chapter 5

Evaluation

This chapter presents empirical results evaluating the thesis proposals, specifically focusing on our approach's usability, functional adequacy, and implementations. This chapter details the experimental processes and discusses the results.

Usability Testing We conducted usability tests on two key components: from having requirements Specification Module and to having the SCORE ontology. These tests were designed to evaluate the user-friendliness, efficiency, and effectiveness of these components and identify any areas that need improvement.

Use Case This approach has been implemented according to the three key components of our framework: specification of security requirements, formalization, and analysis. Using the use case of a civilian surveillance vehicle, we evaluated each aspect of the framework: Security Requirements Specification, Formalization with SERENA and Security Analysis.

Security Requirements Specification (Step 1): We applied the SECRET schema and the SCORE ontology to exhaustively define the security requirements of the surveillance car. This initial step of the framework expanded our approach to the requirements specification to include detailed security considerations.

Formalization with SERENA (Step 2): In the next step, we used the models generated from the requirements to build formal representations using the SERENA language. The operational semantics of the SERENA language were formalized into CLIF through VariaMos' language semantics specification module. Thus, any instance of the SERENA language will have a formal representation that can be subsequently used to analyze the security properties of the corresponding system.

Security Analysis (Step 3): Finally, in the third part of the framework, we used solvers to analyze formal models and evaluate vehicle-specific security. This evaluation verified the car's compliance with the defined security requirements, demonstrating that this approach can be used in real-world cases.

The use of the concrete use case played a crucial role in testing our framework under practical conditions, thereby refining and validating each methodology step. This three-step approach strengthened the robustness of our security assessment framework.

5.1 Usability Testing

5.1.1 Security Criteria Ontology for Security Requirements Engineering (SCORE)

To validate the implementation results of the SCORE ontology and its web interface, three methods were used: testing an example ourselves, usability testing to assess the interface's functionality, and expert evaluation to assess the accuracy, completeness, and consensus of the ontology itself.

Usability Testing of the Web Interface

The ISO/IEC 25062:2006 usability testing standard was adopted to test the security criteria ontology web interface's usability. Real values must be mathematically perceived for the feature sets: product, users, goals, and usage context. With usability metrics, this gives us the following Cartesian product: usability = LEVEL x PRODUCT x USERS x GOALS x CONTEXT.

Test Objective The objective is to test the usability of the security requirements ontology. Therefore, we focus on the following characteristics:

— Additional requirements specification: the user can interact with the ontology to define additional requirements according to security criteria.

— Security mechanism suggestion: the user's ability to obtain security mechanism suggestions based on specified security criteria and domain.

Usability Metrics We want the tool to be effective, efficient, and satisfying for the user. For this, we define the following:

— Effectiveness:
  - Completion rate: measured according to the completion rate of the test where the participant was able to accomplish all tasks.
  - Error: an incomplete or incorrectly performed task.

— Efficiency:

- Task time: time required to complete each task.
- Task completion efficiency rate: average task completion rate compared to task time.

— Satisfaction: a post-questionnaire was given to participants after the test to assess their satisfaction. We evaluated ease of use, ease of learning, ease of memorization, and subjective satisfaction.

Usability Testing Process Figure 5.1 presents the usability testing process.

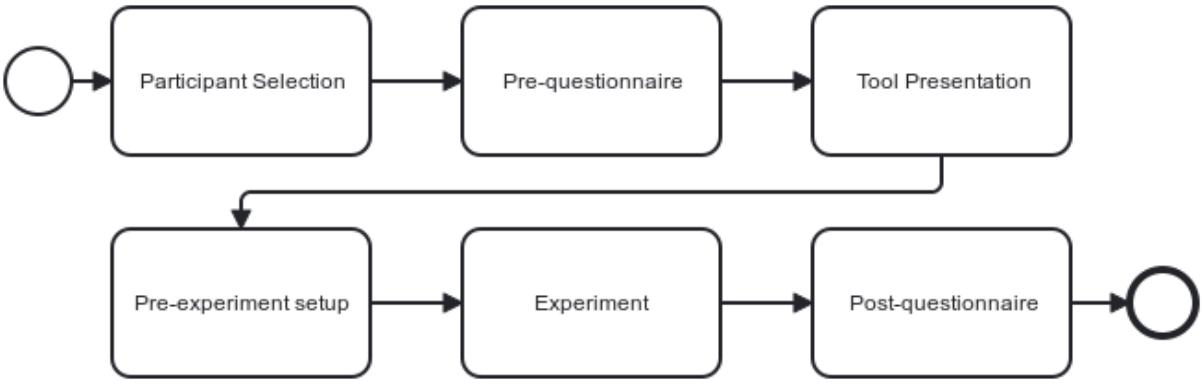


Figure 5.1 – SCORE Usability Test Process

— Participant selection: Nine participants were selected from the software engineering profile. Participants are divided according to country, language, level of education, profession/research, and study/expertise domain.

— Pre-questionnaire (5 minutes): A pre-questionnaire was distributed to participants at the beginning of the usability test to test their prior knowledge of security criteria and requirements engineering.

— Tool presentation (10 minutes): After the prequestionnaire, an introduction was given to participants regarding the tool to be tested and the concepts it contains, such as security criteria, security mechanisms, and security requirements.

— Pre-experiment setup:

— Experiment (15 minutes): After setting up the pre-experiment, a use case document was shared with participants. It includes the following tasks:

- Task 1: Use the ontology page.
- Task 2: Choose the smartphone domain and list the criteria.
- Task 3: Obtain the privacy criteria tree.
- Task 4: Write requirements from privacy using its tree.
- Task 5: Add security mechanisms to each requirement based on security criteria.

— Post-questionnaire (10 minutes): Participants answered another questionnaire after the experiment with questions aimed at addressing the following points: (i) overall satisfaction, (ii) experiment environment, (iii) ontology performance, (iv) general questions, and (v) ontology-specific questions.

Link to the Google Form

Results The results discussed in this section are based on participant performance and satisfaction.

— Performance results: All nine participants completed the task. The average task completion time was 19.5 minutes. As for errors, the average was 0.44, with one participant making four errors, as shown in Table 5.1.

— Satisfaction results: Overall satisfaction level was obtained from question 3.3 of the post-questionnaire. Post-questionnaire responses were also used to evaluate the test environment (questions 3.1 and 3.2), ease of use (questions 3.4 and 3.8), design aspects (questions 3.6 and 3.7), and ontology performance (question 3.5). Detailed results are presented in Table 5.2. Table 2 shows that, on average, participants were generally satisfied with the web interface of the security criteria ontology, as the average scores of the factors we used to evaluate participant satisfaction were all above 4 (out of 5) (see Figure 5.2).

— Participant Comments: We asked the participants for feedback on the web interface. Positives included the criterion tree's usefulness and the interface's ease of use. Criticisms, such as font size, colors, and terminology, have been addressed in an updated version. Some feedback was outside the project scope. All comments are available in an online Excel spreadsheet [1].

Expert Evaluation of SCORE Ontology

This evaluation aims to obtain the expertise of qualified individuals in the fields of cybersecurity, ontology, or requirements to evaluate the concepts and relationships of the SCORE ontology. Experts come from various fields, such as banking, judicial cybersecurity, aviation, ethical hacking, etc.

Evaluation Process To evaluate the completeness, correctness, validity, and consensus of the SCORE ontology, a questionnaire composed of 13 questions covering criteria such as eligibility, interpretability, consistency, clarity, accuracy, etc., was used. This

1. Link to the Results Analysis

Table 5.1 – Performance Results

Participant	Completion Rate (%)	Total Task Time (minutes)	Errors
P1	100	9	0
P2	100	11	0
P3	100	10	0
P4	100	12	0
P5	100	10	0
P6	100	19	0
P7	100	90	0
P8	100	9	0
P9	100	6	4
Mean	100	19.56	0.44
Standard Deviation	0	26.65	1.33
Standard Error	0	8.88	0.44
Min	100	6	0
Max	100	90	4

Table 5.2 – Participant Satisfaction Ratings

Participant	Overall Satisfaction	Test Environment	Ease of Use	Design Aspects	Ontology Performance
P1	5	5	5	5	5
P2	5	4.5	5	5	5
P3	5	5	4.5	5	5
P4	5	3.5	3.5	4.5	5
P5	4	4.5	4	4.5	5
P6	3	5	2.5	1.5	2
P7	5	3	5	5	4
P8	5	5	4.5	4.5	5
P9	5	5	4.5	5	5
Mean	4.67	4.5	4.28	4.44	4.56
Standard Deviation	0.71	0.75	0.83	1.13	1.01
Standard Error	0.24	0.25	0.28	0.38	0.34
Min	3	3	2.5	1.5	2
Max	5	5	5	5	5

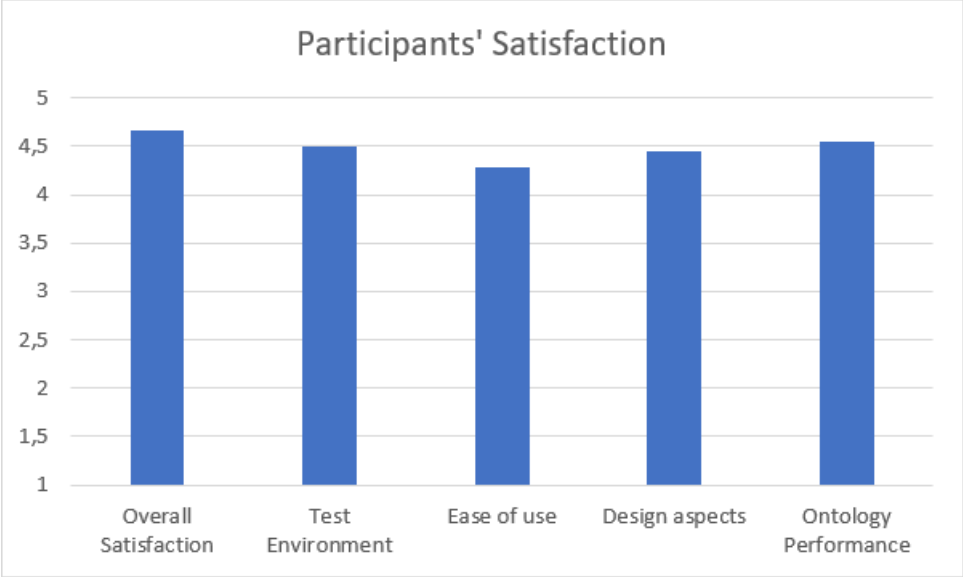


Figure 5.2 – Users' Satisfaction

questionnaire [2] was submitted to experts digitally, and their responses [3] were analyzed.

Participants Participants are experts in cybersecurity, ontology, or requirements, coming from various domains. To ensure the objectivity of the results, experts from different backgrounds, native languages, and age groups were contacted without requesting personal information.

Results We received nine responses to the SCORE ontology evaluation questionnaire. According to the questionnaire results (question Q12 - Do you find this ontology correct?), SCORE is considered 100% correct by experts. Similarly, consensus is also 100%, as shown in the results of question Q13 (Do you agree with the ideas in this ontology?). Participants elaborated on the correctness and consensus of the SCORE ontology by responding to questions Q12.1 (Why do you find this ontology correct or incorrect?) and 13.1 (Why do you agree or disagree with the ideas in this ontology?). Their responses are presented in Table 5.3.

Completeness of SCORE Ontology Representation of Diverse Domains: 77.78% of experts indicated that the ontology contains sufficient information to represent different domains.

2. Link to the Google Form
3. Link to the Results

Table 5.3 – Experts Comments on Correctness and Concensus

Question	**Answer**
12.1 - Why do you find this ontology correct or not?	(a) All concepts and relationships provide meaningful terms. (b) It seems to cover a wide spectrum of security requirements in areas. The additional criteria make it possible to identify sub-criteria or related criteria from "obvious" criteria. (c) I think it builds its correctness from a thorough literature review and analysis. (d) Because it seems motivated by arguments from the literature. (e) Useful for supporting requirements engineers. (f) With a high percentage, it is logical, based on state-of-art domains and criteria. (g) Concepts and relationships make sense and can be applied in a practical manner. (h) The main security criteria are included. However, I did not get how you came up with the security mechanisms for each criterion. (i) It is representative of several cases.
13.1 - Why do you agree or not agree with the ideas in this ontology?	(a) Provide security semantics to newcomers to at least - reach the objective of ontology. (b) This ontology is presented as a kind of dictionary of the different criteria, allowing the end user to specify and qualify these needs in terms of security. (c) I think you went into great detail, with clear definitions of concepts and justified relations between those concepts in terms of security criteria which I haven't seen before, despite the fact that quite few security ontologies exist in the literature. (d) Coherence. (e) I agree with the ideas of this ontology because the criteria and concepts were covered extensively and correctly. (f) It covers a very wide range of domains, mechanisms, and criteria in a well-relational and organized way. (g) as above (correspond to answer 7 in Question 12.1). (h) I agree because the ontology outlines the dependencies that exist between different security criteria. We rarely think about these dependencies. (i) NA.

Table 5.4 – Questions on the Completeness of SCORE

Question Number	**Description**	**Answers**
7	Does the ontology contain enough information to represent security criteria in different domains?	Refer to Figure 18
7.1	What is missing from the ontology? (If any)	— "Operational Security maybe" — "none" — "Consistency"
10	Does the ontology contain all the necessary concepts to represent security criteria?	Refer to Figure 18
10.1	What are the missing concepts? (If any)	— "Cross check validation with existing security ontology from different models could really help. Maybe choose 5-10 would be enough." — "The detailed coverage of security criteria that you provide in your ontology." — "none"

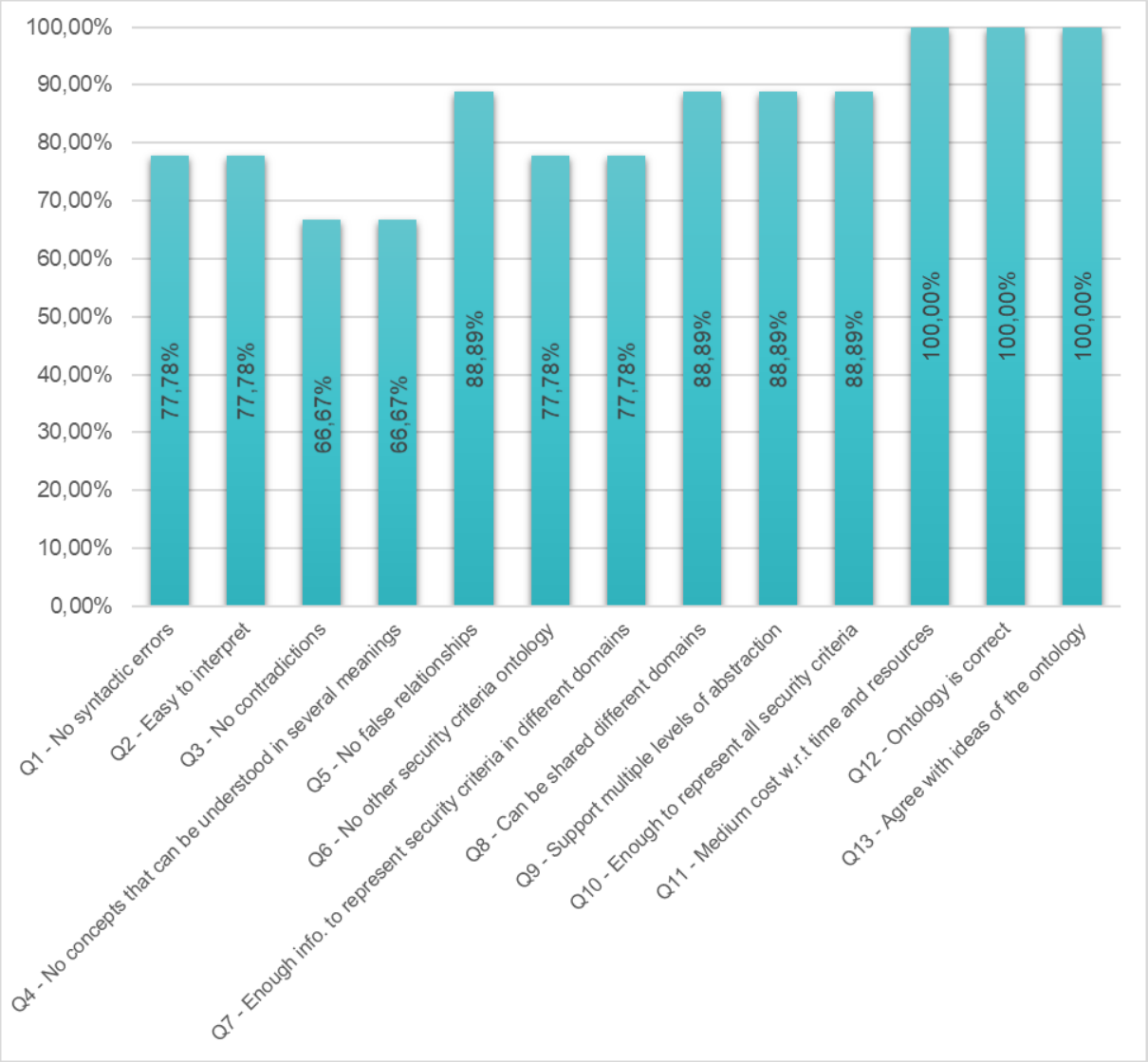


Figure 5.3 – Experts Results

Representation of Security Criteria: 88.89% of experts responded that the ontology contains sufficient information to represent security criteria.

Despite improvement suggestions, demonstrating the rigor of the evaluation, no additional proposals for security criteria or relationships were put forward. The ontology was considered complete in its coverage of relevant aspects. This thorough evaluation attests to the robustness of the SCORE ontology as a tool suitable for representing diverse domains and security criteria, thus meeting the specific requirements of each context.

Therefore, after analyzing the results presented in Figure 5.3 and considering the suggestions from experts as shown in Table 5.4 (see questions 7.1 and 10.1), we con-

sider the SCORE ontology that we implemented, based on the literature review, to be 100% complete. The ontology is validated according to the completeness, correctness, and consensus criteria obtained. Participants suggested modifications to some concepts and relationships, such as trust and responsibility. However, since the ontology is considered correct by all participants, these proposals were addressed by refining the definitions of security criteria without altering the ontology itself.

5.1.2 Requirements Specification Module (SECRET-SCORE)

To evaluate the usability of the requirements engineering module using the SECRET and SCORE models, we adopted the ISO/IEC 25062:2006 standard for usability testing. Mathematically, this evaluation involves the perception of real values for feature sets, including the product, users, goals, and usage context. Using usability metrics, this translates into the following Cartesian product:

$$\text{Usability} = \text{LEVEL} \times \text{PRODUCT} \times \text{USERS} \times \text{GOALS} \times \text{CONTEXT}$$

Product Description **Product:** VariaMos Requirement Specification Languages, version 1.

Website: `http://develop.variamos.com:8000/dashboard`

Target Users: Researchers, students, software developers, and professionals interested in specifying functional, non-functional, and security requirements. This includes individuals with experience in requirements specification and learners interested in requirements specification.

Assistive Technologies: OWL, Apache Jena, JAX-RS, JWT, React, NodeJS, ReactFlow.

Evaluated Parts:

(a) Functional Requirements Specification,

(b) Security Requirements Specification,

(c) Additional Security Requirements Specification,

(d) Requirements Document Generation,

(e) Specification of other non-functional requirements.

Test Objectives The objective is to test the usability of domain and application requirement specification models. Thus, we focus on the following features:

(a) Requirement Specification: The user's ability to interact with requirement specification languages to define requirements.

(b) Additional Security Requirement Suggestions: The user's ability to get suggestions for security requirements based on specified security criteria.

(c) Requirements Document Generation: The ability to generate the SRS or SyRS (requirement document) from the requirements.

Usability Metric We aim to provide results, efficiency, and user satisfaction with the tool. With this in mind, we define the following elements:

Efficiency:

— Completion Rate: Measured based on the test completion rates where participants could accomplish all tasks.

— Error: A task that was incomplete, completed, or incorrectly completed.

Effectiveness:

— Task Time: Time taken to accomplish a task.

— Completion Rate Efficiency: Average completion rate per task time.

Satisfaction: Following the test, participants were asked to complete a post-questionnaire evaluating their satisfaction based on usability, ease of learning, ease of memorization, and subjective satisfaction.

Usability Test Process

(a) **Participant Selection:** A group of 14 participants (students) was carefully selected considering their experience in software engineering profiles or knowledge.

(b) **Pre-questionnaire (5 minutes):** During the usability test, participants received a pre-questionnaire Usability Test Form (`https://forms.gle/QwNijJVyAvfdF8LW6`) to assess their prior knowledge of security and requirements engineering.

(c) **Introduction to the Tool (30 minutes):** After the pre-questionnaire, participants were introduced to the language being tested using a tutorial.

(d) **Experimentation (15 minutes):** After pre-experiment setup, participants received a sample document to reproduce the example we created in the tutorial, covering the following tasks: Task 1: Specify the first confidentiality security requirement. Task 2: Use the related requirements feature to obtain additional security requirements at each step. Task 3: Generate the SRS or SyRS with all requirements.

(e) **Post-questionnaire (10 minutes):** After the experience, participants filled out a questionnaire aiming to answer the following questions: (i) overall satisfaction, (ii)

experimentation environment, (iii) language features, (iv) general questions, and (v) specific questions about the languages.

Results The results presented in this section are based on participant performance and satisfaction. The exhaustive list of results can be found in the online Excel sheet [4].

— Performance: Performance results are based on participants' task completion. The first task, including a tutorial exercise, took an average of 38 minutes. The second task was completed on average in 22 minutes, while the third task took an average of 6 minutes (see Table 5.5). All eight participants successfully completed the assigned tasks with an average error rate of 0.1, demonstrating the tool's efficiency from a user performance perspective.

— Satisfaction: The overall satisfaction level was determined based on participants' responses to question 3.3 in the post-questionnaire. The questionnaire also provided information on the evaluation of the testing environment (questions 3.1 and 3.2), ease of use (question 3.8), design aspects (questions 3.6 and 3.7), and language performance (question 3.4). The results are presented in Table 5.6. Table 5.6 shows that participants were generally satisfied with the language requirements on average, with an average of 4.13 out of 5 for the factors used to evaluate their satisfaction.

— Participant Comments: We also asked participants to share their feedback on the positive and negative aspects of the languages. Most of the positive aspects were related to related requirements and SRS generation. As for the negative points or areas for improvement, they mainly focused on the user interface. For example, some of the comments received are "Problem with the user interface" and "Refresh every time we want to add a requirement". These points have been taken into account in newer versions of the tool.

5.2 Use Case

To evaluate the applicability and robustness of our proposed framework, we applied it to the use case of a civilian vehicle [5]. This use case was chosen due to its practical relevance and the comprehensive security requirements involved.

The surveillance vehicle effectively monitors and protects various terrains and weather conditions. It is crucial for border surveillance, detecting intruders and vehicles, managing

4. Link to Results

5. https://www.iom.int/sites/g/files/tmzbdl486/files/Technical%20Specifications$_{M}$*obile*%20*Surveillance*%20*System*%20
IR%20*cameras*%20*and*%20*Radar*$_{I}$*tem*%202.*pdf*

Table 5.5 – Participant Performance

Participant	Completion Rate (%)	Total Task 1 Time	Total Task 2 Time	Total Task 3 Time	Errors
P1	100	5	20	1	1
P2	100	90	55	5	0
P3	100	30	30	30	0
P4	100	4	3	1	0
P5	100	20	90	1	0
P6	100	5	15	5	0
P7	100	35	5	5	0
P8	100	45	8	3	1
P9	100	60	30	1	0
P10	100	40	10	15	0
P11	100	120	4	0	0
P12	100	45	10	10	0
P13	100	2	30	5	0
P14	100	35	2	15	0
Mean	100	38.29	22.29	6.93	0.14
Standard Deviation	0	34.02	24.52	8.28	0.36
Standard Error	0	9.09	6.55	2.21	0.10
Minimum	100	2	2	0	0
Maximum	100	120	90	30	1

border guards, and executing tasks like prosecution and blocking. The mobile system enhances border services' capacity to combat migratory pressures at the EU's external borders.

The use case scope includes delivering and integrating surveillance equipment, training personnel, and ensuring operational readiness. The mobile surveillance system must work seamlessly in various environments and provide real-time data for decision-making.

This use case tested our framework in real-world conditions, ensuring the methodology is both theoretically sound and practically effective.

Since the requirements of the surveillance vehicle are specified for a system rather than a domain, the framework's domain requirements specification and system configuration steps are skipped in this use case. However, domain requirements specification has been evaluated in the usability test. This thesis has not addressed The system configuration step and thus will not be evaluated in this use case.

Table 5.6 – Participant Satisfaction

Participant	Overall Satisfaction	Test Environment	Ease of Use	Design Aspects	Languages Performance
P1	4	4	3	5	5
P2	2	3.5	1	3.5	3
P3	4	3	4	2.5	3
P4	4	5	3	4	4
P5	4	4.5	4	4.5	4
P6	4	3.5	3	4.5	4
P7	4	5	4	5	4
P8	5	5	5	5	4
P9	4	3.5	3	4	4
P10	3	4	3	4	4
P11	5	4.5	4	5	5
P12	5	4	5	4.5	5
P13	4	5	5	5	4
P14	5	5	5	4	5
Mean	4.07	4.44	3.71	4.32	4.14
Standard Deviation	0.83	2.28	1.14	0.72	0.66
Standard Error	0.22	0.76	0.30	0.19	0.18
Min	2	3	1	2.5	3
Max	5	5	5	5	5

5.2.1 Requirements Specification

The system's requirements document encompasses many requirements addressing various components and functionalities. Nevertheless, this study concentrates solely on data management and security. This focus was chosen because data management is a vital element of the surveillance system, and securing it is crucial for the system's overall effectiveness and dependability. The data management requirements are presented in section 5.4 of the requirements document. The VariaMos project was created for the transportation domain since the system under study is a vehicle.

Functional Requirements

Using the requirements specification document, we start with the specification of the prose requirements and adapt them to the SECRET template using the auto-complete

service. Figure 5.4 represents the functional requirement 5.4.1.a *"The data shall be recorded on a digital video recorder (DVR)"* respecified using the SECRET template. Figure 5.5 shows a section of the functional requirements and their relationships. The full is available by uploading the JSON file [6] in VariaMos.

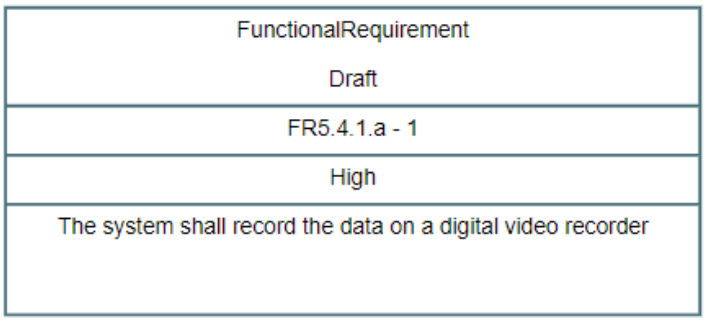


Figure 5.4 – Functional Requirement

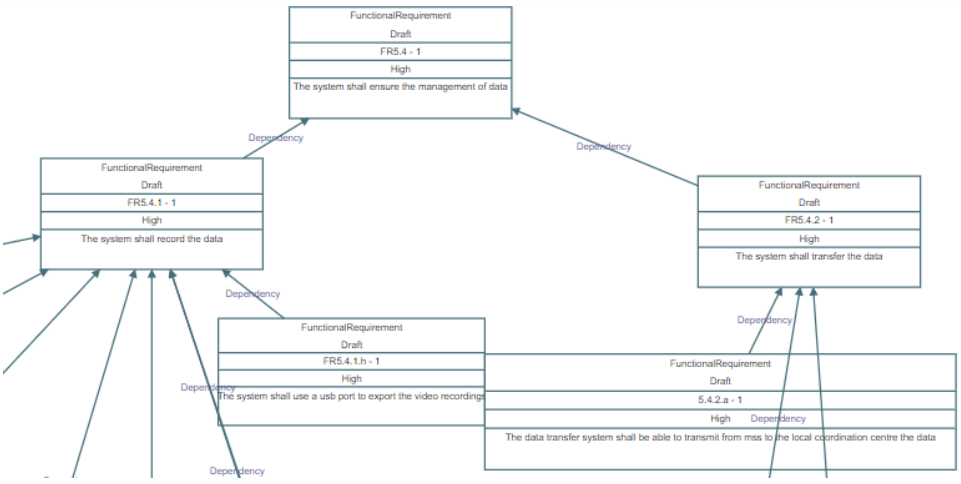


Figure 5.5 – An example section of the functional requirements

Security Requirements

The requirements document of the surveillance vehicle does not specify any security for the system. However, when specifying the security requirements for the surveillance vehicle, we focused on integrity. This criterion is particularly crucial as it is directly related to controlling surveillance data and communication.

6. https://drive.google.com/file/d/1CnqNy1otmZwpwuP8m6gfKfVrzdqk5BTF/view?usp=drive*l**ink*

To further enhance security, we utilized the additional security requirements feature, which helped identify complementary requirements specific to our domain of surveillance. These additional requirements contribute to the overall security measures of the surveillance vehicle, ensuring adequate protection in various scenarios.

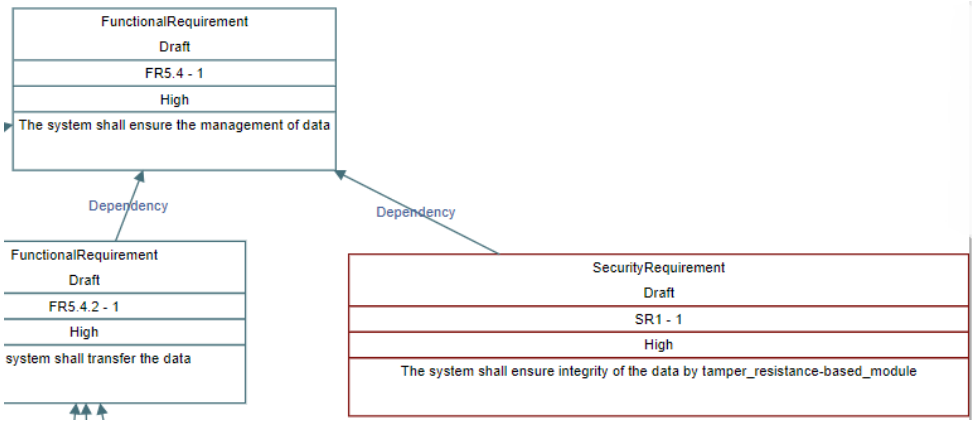


Figure 5.6 – Security Requirement

We begin by incorporating the initial security requirement that addresses the integrity criteria, as shown in Figure 5.6. Subsequently, we utilize the related requirements feature provided by VariaMos, which automatically includes the additional security requirements and their respective priorities. Finally, we define the security mechanism for each newly added security requirement, as illustrated in Figure 5.7. This step can be repeated for each generated security requirement to also get its related security requirements. The full specification can be found in the JSON file [7]

Using the ontology through the related security requirements feature, we were able to obtain the following security requirements:

— **SR1:** The system shall ensure the integrity of the data by tamper-resistance-based module.
  - **SR1.1:** The system shall ensure access control of the data by module-specific firewall.
    — **SR1.1.1:** The system shall ensure authorization of the data by SCMS (Security Certificate Management System).
      — **SR1.1.1.1:** The system shall ensure authentication of the data by message authentication code.
      — **SR1.1.1.2:** The system could ensure trust of the data by trusted platform module.

7. https://drive.google.com/file/d/1uqvDosRkJxKvdFehp4pe-P1McYrgqKBe/view?usp=sharing

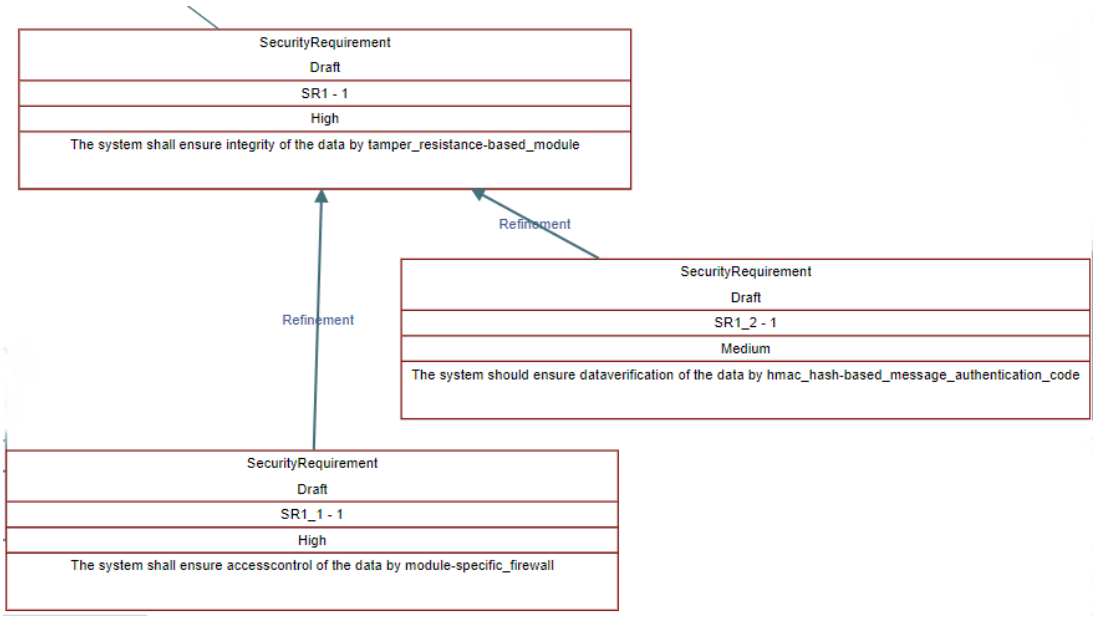


Figure 5.7 – Adding related security requirements

— **SR1.1.2:** The system shall ensure physical security of the data by system-on-chip design and non-detachable connections.

- **SR1.2:** The system should ensure data verification of the data by HMAC (hash-based message authentication code).

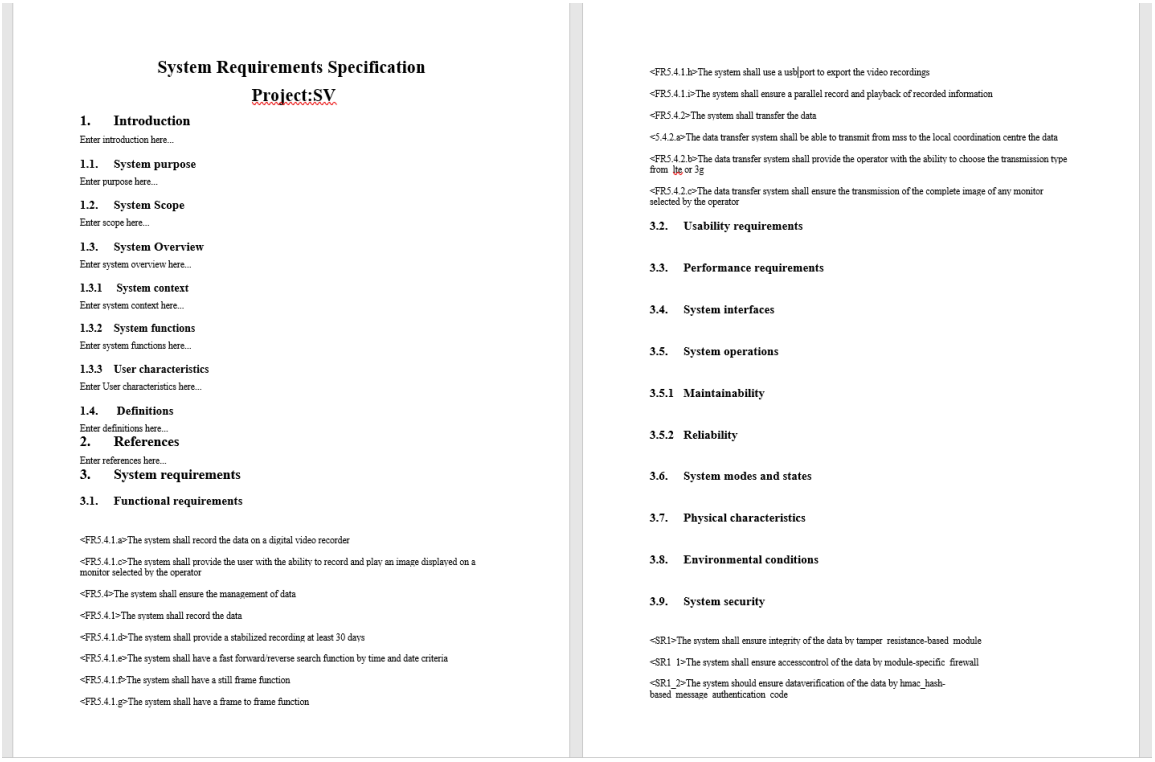
Requirements Document Generation

After specifying the functional and security requirements, we generated a modifiable Word document, as in Figure 5.8, containing the specified requirements and other sections that can be supplemented with additional project information. This document provides a consolidated view of the system requirements, facilitating communication and documentation within the project's development process. The user can thus add additional details, annotations, or other relevant information to gain a thorough understanding of the system requirements.

5.2.2 Requirements Formalization

To formalize the requirements, we start by generating the SERENA criteria model.

SERENA Criteria Model In this step, we identify and formally represent the relevant security criteria that will guide the analysis and design of the system. Using the



System Requirements Specification

Project:SV

1. Introduction

Enter introduction here...

1.1. System purpose

Enter purpose here...

1.2. System Scope

Enter scope here...

1.3. System Overview

Enter system overview here...

1.3.1 System context

Enter system context here...

1.3.2 System functions

Enter system functions here...

1.3.3 User characteristics

Enter User characteristics here...

1.4. Definitions

Enter definitions here...

2. References

Enter references here...

3. System requirements

3.1. Functional requirements

<FR5.4.1.a>The system shall record the data on a digital video recorder

<FR5.4.1.c>The system shall provide the user with the ability to record and play an image displayed on a monitor selected by the operator

<FR5.4>The system shall ensure the management of data

<FR5.4.1>The system shall record the data

<FR5.4.1.d>The system shall provide a stabilized recording at least 30 days

<FR5.4.1.e>The system shall have a fast forward/reverse search function by time and date criteria

<FR5.4.1.f>The system shall have a still frame function

<FR5.4.1.g>The system shall have a frame to frame function

<FR5.4.1.h>The system shall use a usb|port to export the video recordings

<FR5.4.1.i>The system shall ensure a parallel record and playback of recorded information

<FR5.4.2>The system shall transfer the data

<5.4.2.a>The data transfer system shall be able to transmit from mss to the local coordination centre the data

<FR5.4.2.b>The data transfer system shall provide the operator with the ability to choose the transmission type from lte or 3g

<FR5.4.2.c>The data transfer system shall ensure the transmission of the complete image of any monitor selected by the operator

3.2. Usability requirements

3.3. Performance requirements

3.4. System interfaces

3.5. System operations

3.5.1 Maintainability

3.5.2 Reliability

3.6. System modes and states

3.7. Physical characteristics

3.8. Environmental conditions

3.9. System security

<SR1>The system shall ensure integrity of the data by tamper resistance-based module

<SR1 1>The system shall ensure accesscontrol of the data by module-specific firewall

<SR1_2>The system should ensure dataverification of the data by hmac_hash-based message authentication code

Figure 5.8 – Requirements Document

related security requirements feature, it is observed that the generated criteria model, Figure 2.7, conforms to the security criteria tree of the SCORE ontology presented in Figure 5.9 while preserving the values of the relationships.

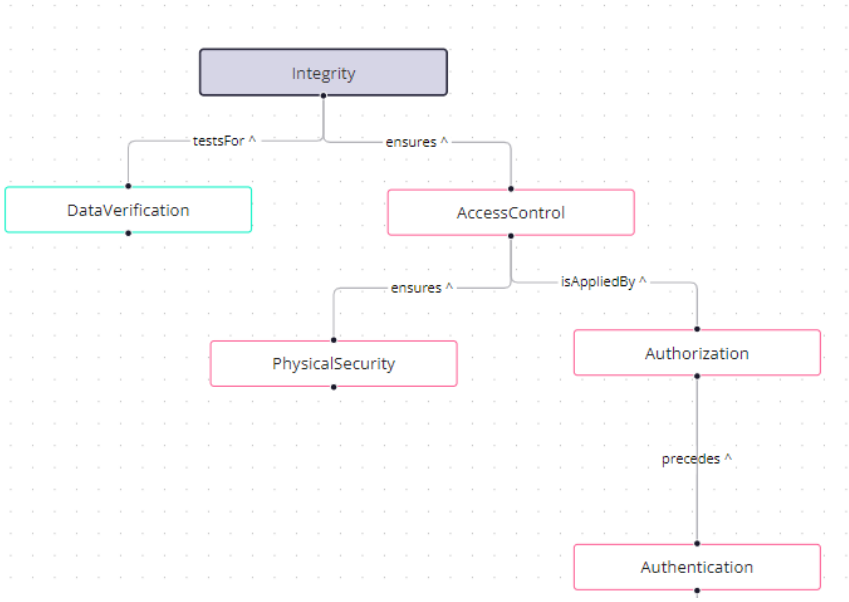


Figure 5.9 – Security Criteria Tree from SCORE

This conformity is essential to ensure that the security criteria modeled in VariaMos accurately and consistently reflect the structure defined in the SCORE ontology. The use of the linked requirements feature facilitates this consistency by ensuring that the generated security criteria are aligned with the semantics defined in the ontology.

This observation reinforces the SERENA criteria model's quality, thereby creating a solid foundation for subsequent steps in the process, such as formalization and security analysis. It also illustrates how VariaMos, by integrating the logic of the SCORE ontology, helps maintain coherence between security specifications and the generated models. The model can be found in the JSON file [8].

SERENA Goal Model The automatic generation of the SERENA objectives model from the application requirements model allows for the representation of the system's functional objectives, Figure 5.11, and their operationalizations. This step is crucial for establishing a detailed link between the overall objectives and concrete actions, thereby facilitating a thorough security analysis. The complete model can be found in the JSON

8. https://drive.google.com/file/d/1SHBH$_d$C0b5ZLui3lJxgh8blyKhjMhG35/view?usp = sharing

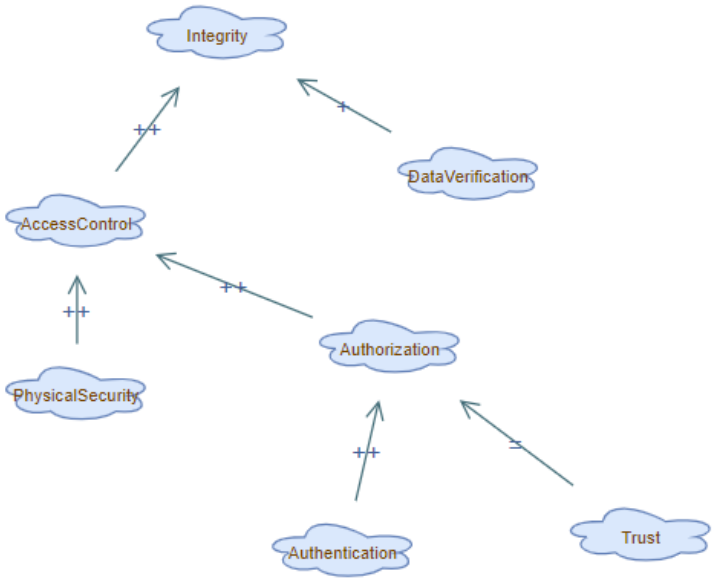


Figure 5.10 – SERENA Criteria Model

file [9].

SERENA Risk Model The SERENA risk model allows users to define potential vulnerabilities that could compromise security. In addition, it also enables the identification of specific threats that could exploit these vulnerabilities.

After applying the NIST 800-30 risk assessment to the threats, as explained in Section 2.1.2, Tables 5.7 and 5.8 were deduced. Figure 5.12 shows a part of the resulting risk model. The full model can be found in the JSON file [10].

SERENA Treatment Model The generated treatment model considers the security mechanisms identified in the security requirements. For each threat identified in the risk model, the user can specify the relevant security mechanisms that can be used to mitigate this threat. Subsequently, the user can define the security value contributed by each security mechanism to the associated security criteria. This step allows for the evaluation of how each security mechanism contributes to reducing the risks associated with the identified threats, thereby enhancing the system's overall security. This step is

9. https://drive.google.com/file/d/1-TXQGWGe3$_W$*cttQyLOgh*2*GfegiqLuuf*5/*view?usp* = *sharing*
10. https://drive.google.com/file/d/1oPQMroyWRDbVn5Ng9$_A$*do*8*NjK*$_8$*QPYp*2/*view?usp* = *sharing*

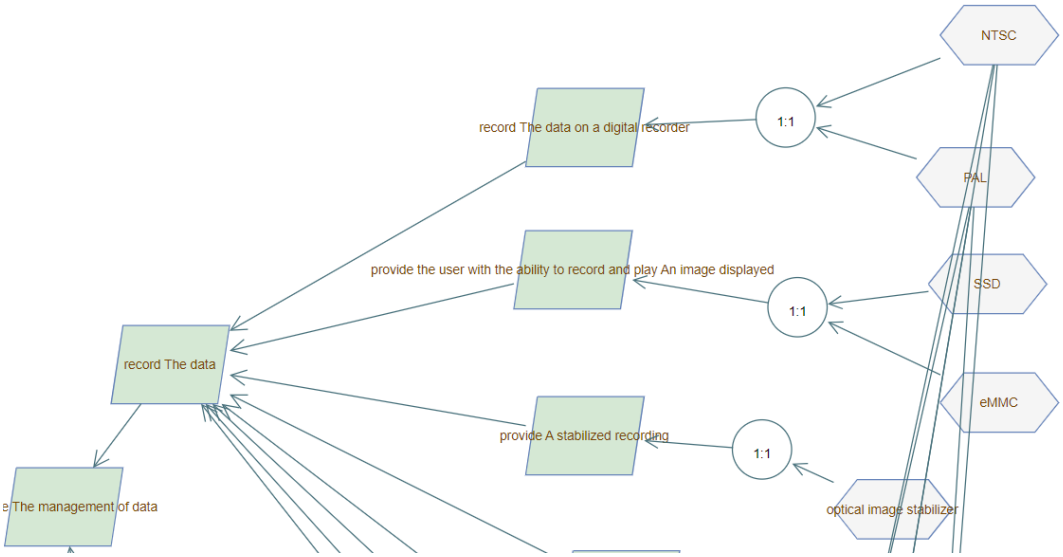


Figure 5.11 – SERENA Goal Model

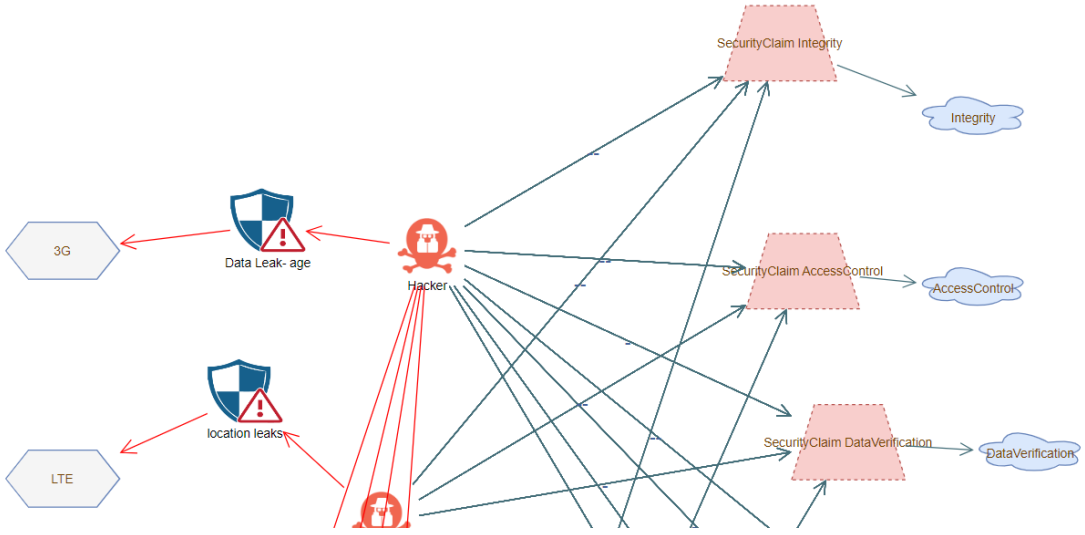


Figure 5.12 – SERENA Risk Model

Threat Event	Threat Sources	Threat Source Characteristics			Relevance	Likelihood of Attack Initiation	Vulnerabilities and Predisposing Conditions	Severity and Pervasiveness	Likelihood Initiated Attack Succeeds	Overall Likelihood	Level of Impact	Risk
		Capability	**Intent**	**Targeting**								
Modify system behaviour	Hacker	High	Obstruct Functioning	3G, USB-A, PAL, SSD, OpenCV	High	Moderate	Data Leakage, Unprotected Alternate Channel, ...	High	Low	Low	High	High
Detect confidential data	MITM	Moderate	Data detection	LTE, NTSC	High	Moderate	location leak, malicious payload	High	Moderate	Moderate	Moderate	High
Stop or modify services	Malware	Moderate	Obstruct Functioning	eMMC, USBC, TCP	High	High	Firmware Insecurity, USB Killer, hijacking	High	Moderate	Moderate	High	High

Table 5.7 – Adversarial Risks Evaluation

	Integrity	Access Control	Data Verification	Authorization	Physical Security	Authentication	Trust
Hacker	–	–	-	–	-	–	-
MITM	–	–	-	=	–	-	-
Malware	–	–	–	-	=	-	-

Table 5.8 – Mapping of Security Criteria to Threats

also part of the security analysis. The analysis is presented in Tables 5.9 and 5.10. Figure 5.13 presents a part of the model. The complete model can be found in the JSON file [11].

	Hacker	MITM	Malware
Tamper Resistance-based module	X		X
Module-specific Firewall	X	X	X
HMAC Hash-based Message Authentication Code	X	X	
SCMS Security Certificate Management System	X	X	
system-on-chip design and non-detachable connections		X	
Message Authentication Code	X	X	
Trusted Platform Module	X	X	

Table 5.9 – Security Mechanisms Mitigate Threats

SERENA Overall Model In the SERENA Overall Model, security values from the risk and treatment diagrams are aggregated to provide an overview of the system's security. This helps stakeholders understand the overall security level and contributions of security mechanisms. Including contextual variables allows consideration of external factors influencing security. Connecting to "soft" objectives highlights other important system aspects, offering a holistic view of its requirements and objectives.

The two context variables identified for the surveillance vehicle are Data Sensitivity and Communication Bandwidth.

— Data Sensitivity

- High Sensitivity: The vehicle handles highly sensitive data requiring stringent encryption, access controls, and monitoring to ensure confidentiality and security.
- Low Sensitivity: The vehicle handles non-sensitive, publicly accessible data. Security measures focus on integrity and availability and can be less stringent than

11. https://drive.google.com/file/d/12xOWN-wh5I5Al-v0NRbYBELL93$_P$l0Q4/$view?usp = sharing$

	Integrity	Access Control	Data Verification	Authorization	Physical Security	Authentication	Trust
Tamper Resistance-based module	++	=	++	=	+	=	+
Module-specific Firewall	++	++	+	++	=	++	+
HMAC Hash-based Message Authentication Code	+	+	+	+	=	++	+
SCMS Security Certificate Management System	+	++	+	++	=	+	+
system-on-chip design and non-detachable connections	+	++	=	+	++	=	+
Message Authentication Code	++	=	+	=	=	++	+
Trusted Platform Module	+	+	=	+	+	=	=

Table 5.10 – Security Mechanism Contribution to Security Criteria

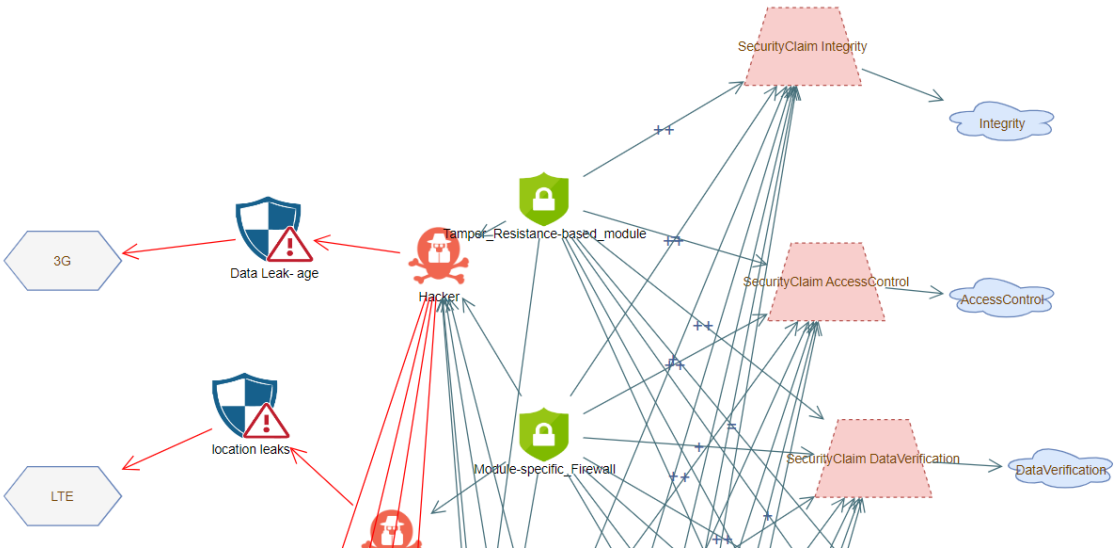


Figure 5.13 – SERENA Treatment Model

those for classified data.

— Communication Bandwidth

- High Bandwidth: The vehicle uses high-bandwidth channels to rapidly transmit large datasets like video feeds, requiring strong encryption and efficient compression.
- Low Bandwidth: The vehicle uses low-bandwidth channels, limiting data transmission. Critical data must be prioritized, lightweight encryption used, and essential data securely communicated despite bandwidth constraints.

The next step is to define the system's soft influence at various context levels. We assess how different conditions affect security requirements and determine the appropriate security level needed. This ensures secure operation across various environments and scenarios. Table 5.11 represents the soft influences of the context variables. Figure 5.14 shows a part of the overall model. The complete model can be found in the JSON file [12]

5.2.3 Security Analysis

After constructing the SERENA model, we generate CLIF (Common Logic Interchange Format) code, which includes all relevant requirements, goals, operationalizations, and risk values. This CLIF code enables in-depth security analysis using logical solvers like SWI-Prolog and constraint solvers such as MiniZinc, ensuring the system's consis-

12. https://drive.google.com/file/d/1XdCycG9VyI80Hh3mYKgGw6R8UNJNLOY4/view?usp=sharing

Context Variable		Security Criterion						
		Integ-rity	Access Con-trol	Data Verifi-cation	Authori-zation	Physical Secu-rity	Authent-ication	Trust
Data Sensitivity	Low	+	+					
	High	++	++	++	++	+	++	+
Communication Bandwidth	Low	+		+	+		+	
	High	++		+	+		+	+

Table 5.11 – Context Variable Soft Influence

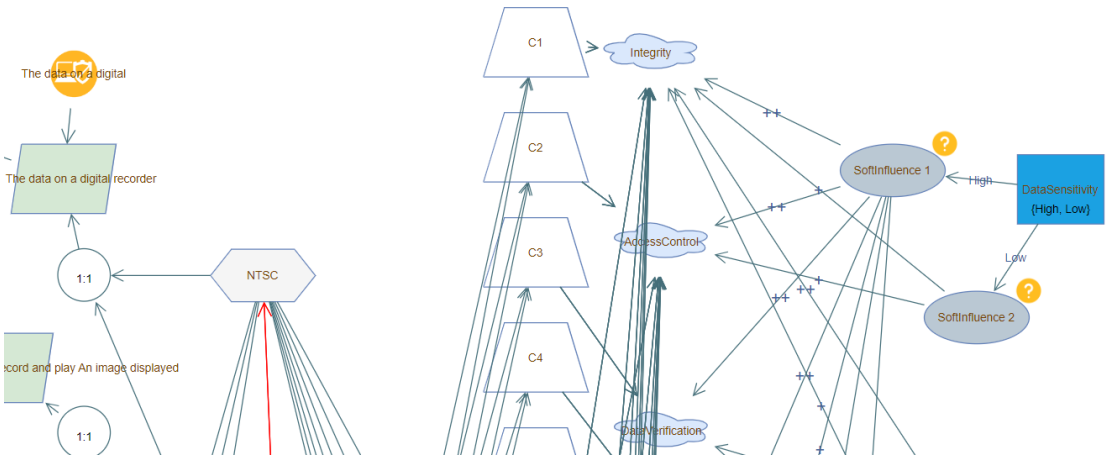


Figure 5.14 – SERENA Overall Model

tency, efficiency, and robustness regarding security objectives. To obtain the generated code, the values of the context variables must be specified. These context variables are essential because they determine the conditions under which the system or service will be evaluated. By providing the appropriate values for these variables, users can obtain customized code that accounts for the different configurations or environments in which the system must operate. This ensures that the generated code is tailored to the specific operational context, enhancing its relevance and effectiveness for security analysis.

Once the values of the context variables are specified, the user can return to the CLIF tool and choose to synchronize the CLIF semantics. This action triggers the generation of the CLIF code associated with the specified context variable values. The generated CLIF code is displayed on the "CLIF Semantics" tab, as in Figure 5.15. This process ensures that the CLIF code accurately reflects the specific conditions and configurations in which the system is intended to operate, facilitating a thorough and context-sensitive security analysis.

In the "Solver-Specific Semantics" tab, the general CLIF code can be automatically transformed into code suitable for different solvers, such as MiniZinc, which we will use in this case. To do this, simply select MiniZinc as the solver from the dropdown menu and click on "Get MiniZinc Model." This generates the model adapted to MiniZinc, ready to be used for constraint analysis and solving specific problems (see Figure 5.16). This step streamlines the process by providing a ready-to-use model for the selected solver, facilitating efficient analysis and problem resolution. The generated code can be copied into the MiniZinc tool for process analysis or queries, or the queries can be performed directly within VariaMos. In this example, we will use the MiniZinc tool.

However, we first use a query in the CLIF module in VariaMos that aims to determine whether the model is satisfiable, that is, whether there exists an assignment of values to the model's variables that makes all constraints true. This helps to verify whether the model is consistent and if there is a solution that meets all the specified constraints. This verification is crucial to ensure that the model contains no contradictions and that it is possible to find at least one solution (see Figure 5.17). Once the query is submitted, the system processes the information and displays the results in the dedicated tab. In this case, the results indicate that the model is satisfiable, meaning that there is at least one solution that satisfies the constraints specified in the model. This confirms that the conditions stated in the model are not contradictory and that it is possible to find a coherent solution (see Figure 5.18). With the model satisfiable, we will copy the generated MiniZinc code into the MiniZinc tool and solve it to maximize claims, soft influences, and security. This will help us find the optimal configuration that meets security requirements

Figure 5.15 – CLIF code

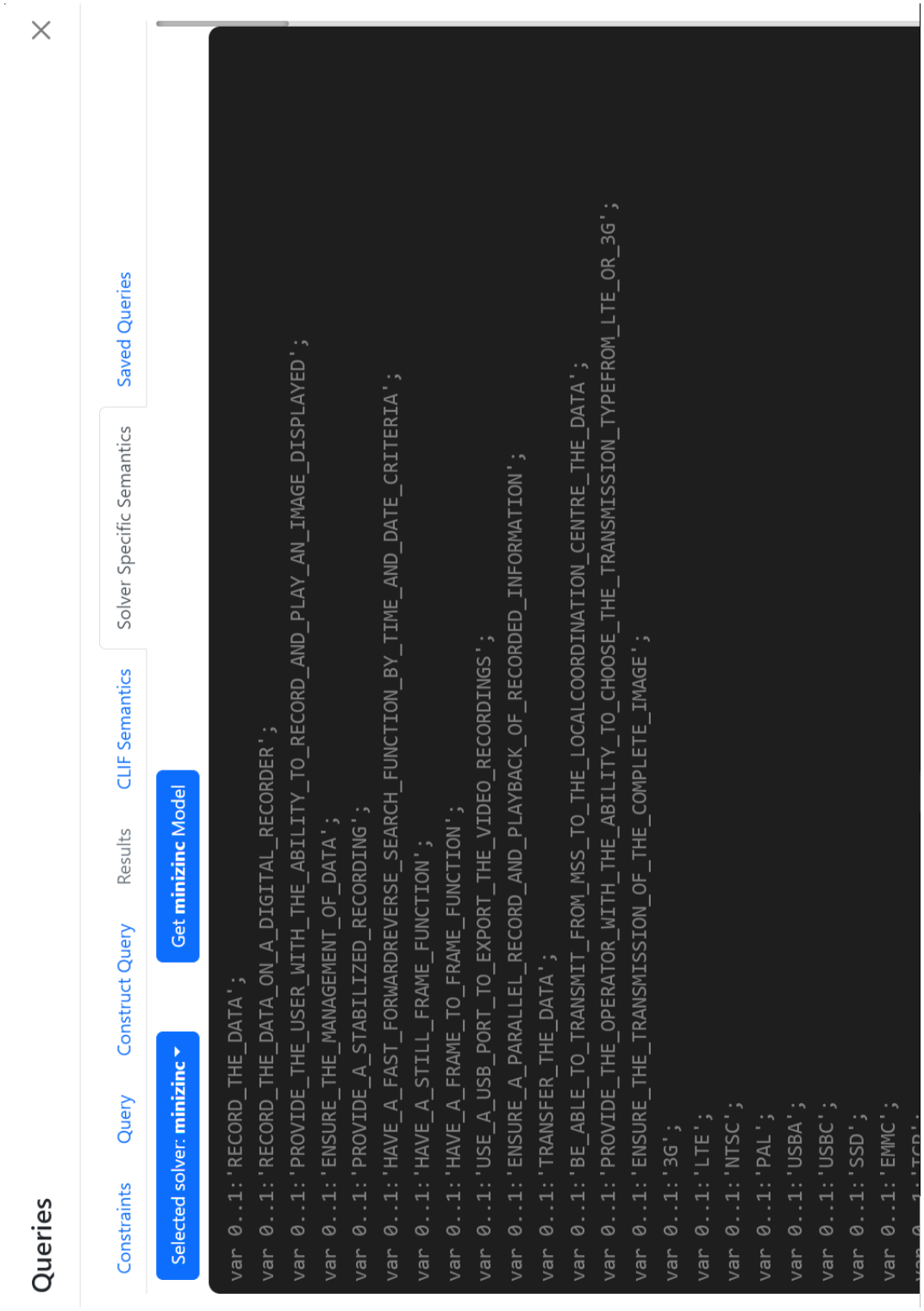


Figure 5.16 – Minizinc code

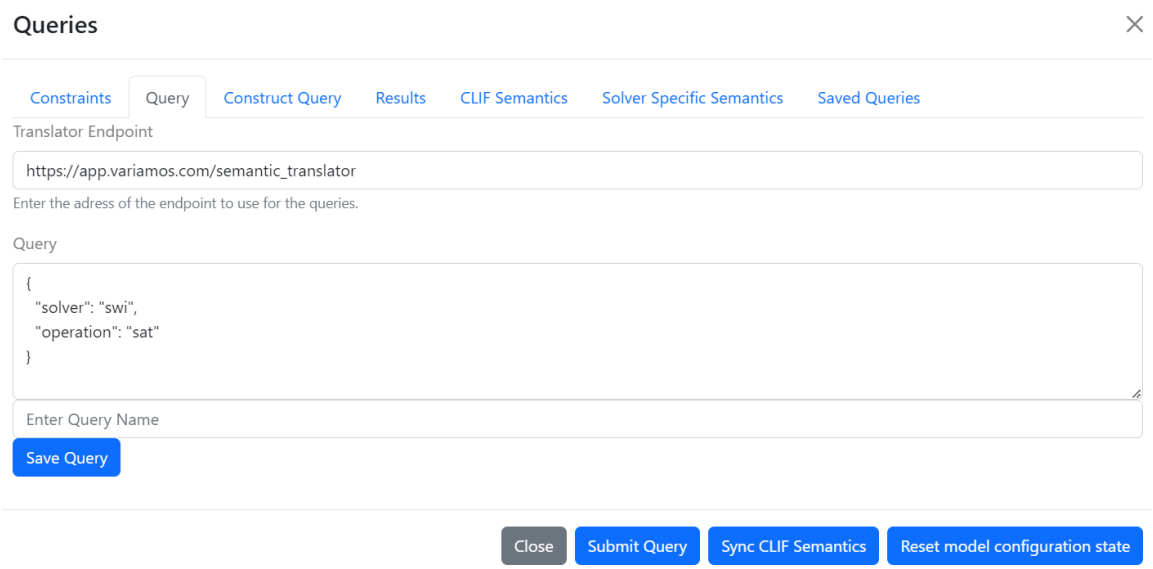


Figure 5.17 – Model Satisfability

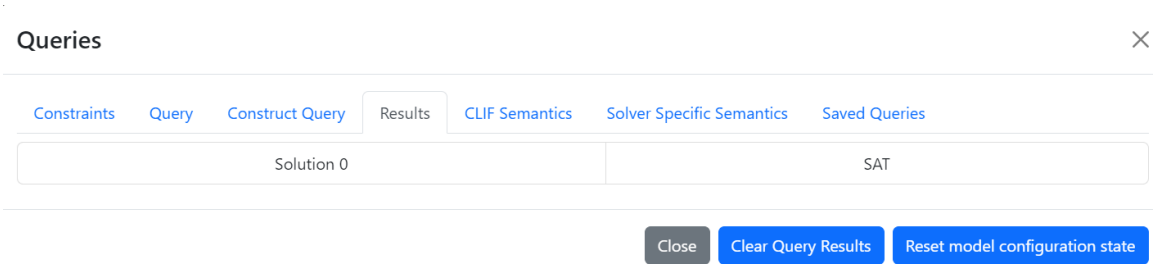


Figure 5.18 – Satisfability Result

while considering the specified influences and claims. The generated code is available in the online file [13]. However, to apply the analysis it is important to add the augmentations that represent the total of the claims, soft influences, and security criteria (softgoals). In this case, due to the different representations of functional goals between the MiniZinc code we wrote manually for the running example and the generated MiniZinc code, it was important to add the the sum of the functional goals also to the maximization equation. The augmented MiniZinc model can be found in the online file [14]

```
var int:TotS;var int:TotC;var int:TotSI;var int:goal;
TotC=C1+C2+C3+C4+C5+C6+C7+C8+C9+C10+C11+C12+C13+C14+C15+C16+C17
    +C18+C19+C20+C21+C22+C23+C24+C25+C26+C27+C28+C29+C30+C31+C32
    +C33+C34+C35+C36+C37+C38+C39+C40+C41+C42+C43+C44+C45+C46+C47
    +C48+C49;
TotS='INTEGRITY'+'ACCESSCONTROL'+'DATAVERIFICATION'+'
    AUTHORIZATION'+'PHYSICALSECURITY'+'AUTHENTICATION'+'TRUST';
TotSI=SOFTINFLUENCE_1+SOFTINFLUENCE_2+SOFTINFLUENCE_3+
    SOFTINFLUENCE_4;
goal='RECORD_THE_DATA'+'RECORD_THE_DATA_ON_A_DIGITAL_RECORDER
    '+'
    PROVIDE_THE_USER_WITH_THE_ABILITY_TO_RECORD_AND_PLAY_AN_IMAGE_
      DISPLAYED'+'ENSURE_THE_MANAGEMENT_OF_DATA'+'
    PROVIDE_A_STABILIZED_RECORDING'+'
    HAVE_A_FAST_FORWARDREVERSE_SEARCH_FUNCTION_BY_TIME_AND_DATE_
      CRITERIA'+'HAVE_A_STILL_FRAME_FUNCTION'+'
    HAVE_A_FRAME_TO_FRAME_FUNCTION'+'
    USE_A_USB_PORT_TO_EXPORT_THE_VIDEO_RECORDINGS'+'
    ENSURE_A_PARALLEL_RECORD_AND_PLAYBACK_OF_RECORDED_INFORMATION
    '+'TRANSFER_THE_DATA'+'
    BE_ABLE_TO_TRANSMIT_FROM_MSS_TO_THE_LOCALCOORDINATION_CENTRE
      _THE_DATA'+'
    PROVIDE_THE_OPERATOR_WITH_THE_ABILITY_TO_CHOOSE_THE_TRANSMISSION
      _TYPEFROM_LTE_OR_3G'+'
    ENSURE_THE_TRANSMISSION_OF_THE_COMPLETE_IMAGE';
solve maximize (10000*goal)+(1000*TotC)+(100*TotSI)+(100*TotS);
```

After executing the code, several solutions were provided, with the optimum presented

13. https://drive.google.com/file/d/12Z-lsKUOOcG5n9cMH1a7g7JB713rZdew/view?usp=sharing
14. https://drive.google.com/file/d/1B7jJ4W-YFXd3UoqbsPCyrKWF5dqknKAP/view?usp=sharing

last. This indicates that the solver explored different combinations of values to maximize claims, soft influences, and security, and it identified the optimal solution that satisfies all the model's constraints. This process demonstrates the solver's capability to evaluate various scenarios and determine the best configuration for achieving the desired security and performance goals.

```
.
.
.
'3G' = 1;
LTE = 0;
NTSC = 1;
PAL = 0;
USBA = 1;
USBC = 0;
SSD = 1;
EMMC = 0;
TCP = 1;
OPTICAL_IMAGE_STABILIZER = 1;
OPENCV = 1;
INTEGRITY = 2;
ACCESSCONTROL = 2;
DATAVERIFICATION = 2;
AUTHORIZATION = 2;
PHYSICALSECURITY = 2;
AUTHENTICATION = 2;
TRUST = 2;
.
.
.
TotS = 14;
TotC = 49;
TotSI = 2;
goal = 14;
```

The important factors for our analysis are represented in the above results. It shows the chosen operationalizations with value '1' (eg. '3G'=1), the score for each security criterion, and the total security score corresponding to the optimal system configuration.

5.2.4 Conclusion

In this scenario, we successfully defined both functional and security requirements using the requirements specification module in VariaMos. We leveraged its guided features, including an auto-complete service, additional security requirement suggestions, and the ability to generate comprehensive requirements documents. This process allowed us to progress from having no specified security requirements to identifying seven crucial security requirements.

Using automatic transformations from textual requirements to SERENA models, we specified the operationalizations, performed risk assessments, defined the treatments, and ultimately created an overall model with the final security value of each operationalization. We also specify the context variables. These transformations significantly reduced the modeling time, as elements and relationships were automatically added, requiring only minor additions to the models. This efficiency was particularly evident when generating the overall model, where 49 claims were generated with their respective values.

For the security analysis, the CLIF module enabled us to check the satisfiability of the overall model and generate MiniZinc code for a detailed security analysis. The entire automated process reduced the time required for specification, formalization, and analysis, while also helping to avoid human errors in all three steps. This streamlined approach ensured a robust and efficient workflow for the protection of the system.

Chapter 6

Conclusion

The main objective of this thesis is to answer the research question *"How can a unified framework be developed for the specification, formalization, and analysis of secure systems?"*. To answer this question, this thesis proposes:

(a) A state of the art in methodologies for writing security requirements, methodologies for formalizing security requirements, and methodologies for security analysis.

(b) A requirements specification module and its implementation in VariaMos.

   i. A security requirements template using semi-structured natural language (SECRET) for guided security and other requirements specification types.

   ii. A security criteria otology to enrich the template with security knowledge and suggest additional security requirements to improve the security coverage in the system.

(c) A new security modeling language, its transformations, and implementation to formalize the requirements specified using the SECRET template.

(d) A security analysis approach using constraint programming to get an objective security score.

(e) An implementation of the framework.

(f) Evaluations of the framework with usability tests and a use case.

6.1 State of the Art

The state of the art shows that:

— Most requirement specification approaches overlook security requirements (Table 1.1).

— The requirement specification approaches considering security do not cover all aspects (Table 1.1).

— None of the proposed security requirements modeling or formalization methodologies is usable with a security requirements specification template.

— None of the proposed security requirements formalization approaches consider the system's context.
— The proposed security requirement formalizations lack accompanying security analyses that offer objective system security scores.
— Most security analysis approaches lack a usable tool.

The following questions arise:

— How to specify clear, non-complex security requirements that explicitly indicate security aspects? And how to couple this specification with security knowledge?
— How to formalize the specified requirements while preserving the correct representation of their content?
— How to apply security analysis to the formalized model that gives an objective security score and considers context variables?
— How to provide a usable tool for the three steps of the framework?

6.2 Security Requirements Specification

We proposed the following contributions to security requirements specification:

6.2.1 SECRET

The SECRET template for security requirements specification offers guidance for defining requirements. The template yields several results:

— Guide the specification of security requirements, facilitating the requirements specification process for requirement engineers.
— Distinguish between problem and solution spaces in requirements. This enables the reuse of the solution for different problems.
— Differentiate domain and application requirements, which provide the ability to configure specific systems from the requirements of a domain or a product line.
— Provide traceability through the meta-data and the relationships between the requirements.

6.2.2 SCORE

The security criteria ontology enriches the approach with security knowledge. The onology contributes to security requirements specification in several ways:

— Guide with the specification of the security criteria and their related security requirements, according to a specified domain.

— Suggest additional security criteria to consider when specifying security requirements. This helps improve the security coverage in the system.

6.2.3 SECRET-SCORE

SECRET-SCORE is a guided approach that integrates the use of the SCORE ontology and the SECRET template. It serves to answer two questions:

— *(i) how to specify the security requirements?*

— *(ii) What security requirements to specify to achieve a high level of security?*

The first question (i) is answered using the SECRET template that defines the structure of the requirements. However, the second question (ii) is addressed by integrating the SCORE ontology, which holds the security knowledge, with the SECRET template. The result of coupling these two approaches is the guided specification of semi-structured security requirements that respect and consider security knowledge in a specific domain.

6.2.4 Requirements Specification Module in VariaMos

The SECRET-SCORE approach was implemented into VariaMos. This implementation provides the following:

— A guided specification approach based on the SECRET template and the SCORE ontology using auto-complete service to facilitate the specification.

— A related security requirements service that uses the ontology and suggests additional security requirements.

— A requirements specification document generation service that gives a document with all the specified requirements.

6.3 Security Requirements Formalization and Analysis

For security requirements formalization and analysis, we proposed several contributions.

6.3.1 SERENA

The goal-oriented security modeling language SERENA offers a security requirements formalization approach adapted to be used with the SECRET-SCORE approach. The

SERENA modeling language produced various outcomes:

— A language that formalizes security requirements while respecting the concise representation of the textual requirements specified using the SECRET template. This formalization aids in security analysis.
— A risk assessment in the SERENA Risk model using the NIST 800-30, which helps to get the objective security score later on.
— A treatment evaluation in the SERENA Risk model.
— An Overall model that represents the security values of each operationalization and associates the model to context variables. This also helps in the security analysis.
— Automatic transformation from textual requirements and federation between the models to avoid any human error.

6.3.2 Automatic generation of Constrain Programs

A constraint program is generated from the SERENA model. It has several results:

— Security analysis that gives the best operationalizations according to the values of the context variables values.
— An objective security score for the solutions after the solvers' analysis.
— The use of the CLIF module in VariaMos provides the automatic generation of the CP. It also enables the use of different solvers and operations or requests.

6.4 Multi-paradigm Framework

The approach offers a comprehensive framework for secure systems design and implementation. It combines several paradigms, including semi-structured natural language requirements specification, to capture security requirements in a human-readable and structured format. This method enhances clarity, simplicity, and completeness, reducing ambiguities and misinterpretations.

The application of ontologies in this method offers an organized knowledge framework that assists in the systematic detection and definition of security requirements. Ontologies provide an extensive and uniform vocabulary for describing security concepts and their interrelations, ensuring uniformity and aiding in identifying security criteria specific to the domain. This organized method enhances overall security coverage by delivering a thorough overview of all pertinent security elements.

This method also includes formal models, providing a clear and exact depiction of the system's security needs. These formal models facilitate thorough analysis and validation,

guaranteeing that the defined security requirements are logically coherent and exhaustive. They serve as the foundation for employing formal security analysis methods, which can systematically identify and address potential security weaknesses.

Incorporating constraint programming within the framework facilitates the automated examination of formalized security models. Techniques from constraint programming are employed to address intricate security issues by determining variable values that meet a series of constraints. Converting security models into constraint programs enables the approach to simulate and evaluate various configurations, guaranteeing a thorough and flexible security analysis procedure.

The implementation of this framework is facilitated through tools like VariaMos, which ensure correct representation and coherence between the different aspects of the SECRET template and the SERENA language. This tool-based implementation automates the transformation and modeling processes, minimizing human errors and enhancing the reliability and accuracy of the formalized models.

Moreover, each part of the framework can function independently within both the approach and the tool, facilitating easier extension and adaptation of the framework as required. This modularity is a significant advantage, particularly because it allows different components of the framework to be used in various contexts without the need to rely on the entire system. For example, if another ontology needs to be employed, the framework can easily accommodate this change by substituting or integrating the new ontology without disrupting the whole structure.

6.5 Evaluation of the Framework

To assess the developed tools and methodology, we conducted two usability tests and implemented a practical use case. The usability tests evaluated the effectiveness, efficiency, and user satisfaction with the tool and methodology. Participants from various fields, including security engineers and system designers, were tasked with performing various activities using the tool. Their feedback was gathered through questionnaires and interviews, concentrating on factors such as ease of use, interface clarity, and overall satisfaction.

The outcomes of the usability tests were very favorable. Participants indicated an overall satisfaction rating exceeding 4 out of 5. They valued the guided natural language specification, which simplified the process of capturing and formalizing security requirements. The tool's structured approach, which integrates ontologies, was also well-received, as it enhanced the comprehensive understanding of security aspects and in-

creased the precision of the specifications.

We implemented a practical use case alongside the usability tests to further assess the framework. This use case involved a real-world system where we used the framework's steps to design and evaluate its security. The use case allowed us to gauge the framework's effectiveness in a practical setting, showcasing its strengths and identifying potential areas for enhancement.

We adhered to the framework's methodology during the use case, beginning with the semi-structured natural language requirements specification. This phase involved documenting the system's security requirements precisely and organized. Ontology was utilized to identify and define all pertinent security criteria, guaranteeing thorough coverage. Subsequently, we converted these specifications into formal models, which were examined using constraint programming techniques.

Each phase of the framework was meticulously assessed during the use case. We noted that the organized methodology enabled a comprehensive and detailed examination of the security requirements. The formal models accurately depicted the security elements, permitting thorough verification and validation. The constraint programming methods allowed for efficient and effective analysis of diverse security scenarios, enhancing security.

The findings from the use case validated the advantages noted in the usability evaluations. The framework demonstrated robustness and flexibility, effectively managing intricate security demands and offering significant insights into the system's security status. Integrating guided natural language specifications, ontologies, formal models, and constraint programming resulted in a thorough and efficient method for designing and analyzing secure systems.

Chapter 7

Future Work

Looking forward, there are many opportunities for the project's development and improvement. These future directions aim to enhance the approach's importance, efficiency, and flexibility, thus creating new opportunities and progress in the field of security engineering. This chapter outlines some research directions and necessary future work for designing secure systems.

7.1 Future Work in the SCORE Ontology

Future developments in the SCORE ontology involve enhancing relationships and conducting comprehensive evaluations. Additionally, it encompasses the following:

— Extend the SCORE ontology with the concepts of vulnerabilities, threats, and risk levels. This automatically applies the risk assessment in the SERENA Risk model.

— Enhance security analysis by populating the ontology with data from MITRE databases like ATT&CK, CAPEC, and CWE, leveraging comprehensive and up-to-date threat intelligence. The process of populating the ontology involves several steps:

(a) Data Extraction: Retrieve pertinent information from the MITRE databases. This encompasses comprehensive details on adversary tactics, techniques, and procedures (TTPs) from MITRE ATT&CK, typical attack patterns from CAPEC, and identified software vulnerabilities from CWE.

(b) Mapping to Ontology: Associate the retrieved data with the relevant concepts in the ontology. For instance:
- MITRE ATT&CK TTPs can be linked to the concepts of "Attack Techniques" and "Tactics".
- CAPEC attack patterns can be associated with "Attack Patterns".
- CWE entries can be connected to "Software Weaknesses" or vulnerabilities.

(c) Ontology Population: Fill the ontology with the associated data. This process includes generating instances of the pertinent concepts and forming connections between them based on the gathered information. For instance, a particular attack

technique from MITRE ATT&CK can be associated with the relevant software weakness from CWE and the attack pattern from CAPEC.

(d) Security Analysis Enhancement: Leverage the enhanced ontology for a more thorough security analysis. The combined threat intelligence from the MITRE databases allows for detecting more subtle vulnerabilities and creating more robust security measures.

7.2 Future Work in Security Requirements Formalization

We aim to extend the SECRET model by adapting it to various functional and architectural models. This will enable more flexible use in system design and analysis, providing a unified approach to defining security requirements alongside other software elements.

7.2.1 Expanding Functional Models

Integrating the SECRET model with functional models bridges the gap between security requirements and functional specifications. This alignment of security goals and constraints with system functionalities offers several advantages:

— **Enhanced Security Awareness in Functional Design:** Integrating security requirements into system functionalities ensures secure software from the start.

— **Consistency and Traceability:** Linking security requirements with functional specifications ensures all security issues are addressed and aids in tracking them through implementation.

— **Early Conflict Detection:** Identify and resolve conflicts between functional and security requirements early, reducing security flaws.

7.2.2 Integrating with Architectural Models

Adapting the SECRET model to architectural frameworks allows for a thorough security evaluation at the architectural stage by embedding security requirements into architectural elements and their interactions. Benefits include:

— **Holistic Security Integration:** Security elements are integrated into the architecture, ensuring security is a core component of system design rather than an afterthought.

— **Improved Risk Management:** By comprehending the influence of security requirements on the architecture, designers can more effectively evaluate and address risks linked to various architectural decisions.
— **Scalable Security Solutions:** Architectural models enable the creation of scalable security measures that can be consistently implemented across various system components and layers.

7.2.3 An Integrated Approach to System Design

The suggested enhancements strive to develop a unified framework that includes security, functionality, and architecture. This comprehensive method of system design and evaluation provides numerous significant advantages:

— **Comprehensive Requirement Specification:** Security requirements are detailed together with functional and architectural requirements, guaranteeing a comprehensive and balanced evaluation of all system aspects.
— **Streamlined Development Process:** By utilizing an integrated model, the complexity of handling distinct security, functionality, and architecture requirements is minimized, thereby simplifying the development process.
— **Improved System Quality:** By integrating security requirements with functional and architectural considerations, the system's overall quality and robustness are improved.

7.3 Enhancing Security Analysis

We aim to enhance SERENA's security analysis by introducing new risk and treatment evaluation methods, adopting advanced assessment techniques like formal analysis, and integrating simulation features for complex threat scenarios.

7.3.1 Integration of Simulation Features

We suggest incorporating simulation capabilities into the SERENA model to enhance comprehension and address intricate threat scenarios. This integration facilitates a dynamic and interactive examination of possible attack vectors and their impacts. The primary advantages of this method are:

— **Scenario-Based Analysis:** Simulating threat scenarios assesses system resilience, identifies vulnerabilities, and tests security mechanisms in a controlled environment.

— **Impact Evaluation:** Simulations offer insights into the impact of security incidents on system operations, including the consequences of attacks and the effectiveness of response strategies.
— **What-If Analysis:** What-if analysis allows security analysts to evaluate the impact of various security policies and configurations, aiding in optimizing measures and informed risk management.

7.3.2 Improving Risk Treatment Strategies

Enhancing the SERENA model's security analysis includes improving risk treatment strategies. This involves developing better methods for mitigating risks and ensuring long-term system security. Key elements include:

— **Adaptive Security Measures:** Implementing adaptive security measures that dynamically respond to threats using real-time data and analytics to adjust controls and policies.
— **Comprehensive Mitigation Plans:** Creating comprehensive mitigation plans to address immediate and long-term security needs, including technical controls, organizational policies, and incident response strategies.

7.4 Using SERENA Language for Digital Twin Creation

This viewpoint examines the potential of employing the SERENA language to develop digital twins of actual systems. We consider this initiative as an initial step to utilize the SERENA language for creating digital twins of real systems [64]. This can be achieved by utilizing integrated data collection points that are currently not utilized in the global SERENA model. These points can act as gateways to input sensor data or other elements of a real system into the SERENA model, thereby forming a digital twin.

Utilizing the SERENA language to create digital twins signifies a novel enhancement of its functionalities. Digital twins are virtual representations of a physical system, allowing for dynamic modeling and simulation of its operations. The following explains how the SERENA language can be applied in this scenario:

— **Integrated Data Collection Points:** The SERENA model uses data collection points to gather real-time information from sensors, monitoring systems, or other sources.

— **Feeding SERENA Model:** The collected data updates the SERENA model with the real system's current state and behavior.
— **Creating Digital Twins in VariaMos:** VariaMos, a modeling and simulation platform, creates digital twins from the enriched SERENA model, using sensor data and real system aspects for parameterization and validation.
— **Simulation and Analysis:** Once the digital twin is created, different scenarios and operating conditions can be simulated for performance analysis, impact prediction, and optimization of operations and security.

7.4.1 Potential Benefits

Creating digital twins using the SERENA language offers several benefits:
— **Dynamic Modeling:** Enables real-time modeling of systems, aiding decision-making and performance optimization.
— **Risk Reduction:** Tests modifications and strategies virtually, reducing risks and costs.
— **Enhanced Security:** Analyzes security scenarios to detect and fix vulnerabilities before they affect the real system.

7.5 Future Work for Evaluation: Case Study on Real Systems for Applicability

Future work will involve a comprehensive case study [65] on real systems to evaluate the SERENA language and its models. This will validate the methods in practical settings and provide insights into their performance. The evaluation will focus on the following key areas:

7.5.1 Case Study Objectives

The primary objectives of the case study are:
— **Validation of SERENA Models:** Evaluate SERENA models' accuracy and completeness in real systems, ensuring they capture all aspects of security and functionality.
— **Effectiveness of Security Analysis:** Assess the effectiveness of security analysis with SERENA models, covering risk assessment, threat mitigation, and security mechanism implementation.

— **Integration with Real Data:** Integrate real-time sensor data into SERENA models, ensuring they adapt dynamically to changing conditions and accurately represent the system.
— **Performance and Scalability:** Assess the SERENA models' performance and scalability in managing large, complex systems, ensuring applicability to diverse environments without significant performance degradation.
— **User Experience and Usability:** Gather feedback from operators and analysts on the usability and experience of SERENA tools, identifying areas for improvement.

Primary sources

[1] J. Geismann, C. Gerking, and E. Bodden, « Towards ensuring security by design in cyber-physical systems engineering processes », *in Proceedings of the 2018 International Conference on Software and System Process*, ser. ICSSP '18, Gothenburg, Sweden: Association for Computing Machinery, 2018, pp. 123–127, ISBN: 9781450364591. DOI: 10.1145/3202710.3203159. [Online]. Available: https://doi.org/10.1145/3202710.3203159.

[2] N. Fuchs and R. Schwitter, « Attempto controlled english (ace) », *CoRR*, vol. cmp-lg/9603003, Mar. 1996.

[3] A. van Renssen, « Gellish: an information representation language, knowledge base and ontology », *ESSDERC 2003. Proceedings of the 33rd European Solid-State Device Research - ESSDERC '03 (IEEE Cat. No. 03EX704)*, pp. 215–228, 2003.

[4] A. Mavin, P. Wilkinson, A. Harwood, and M. Novak, « Easy approach to requirements syntax (ears) », English, *in Proceedings of Requirements Engineering Conference, 2009. RE '09. 17th IEEE International*, 2009 17th IEEE International Requirements Engineering Conference ; Conference date: 31-08-2009 Through 04-09-2009, United States: IEEE, Nov. 2009, pp. 317–322.

[5] M. Kamalrudin, N. Mustafa, and S. Sidek, « A template for writing security requirements », *in APRES*, 2017.

[6] R. Mazo, C. M. Z. Jaramillo, P. Vallejo, and J. M. Medina, « Towards a new template for the specification of requirements in semi-structured natural language », *J. Softw. Eng. Res. Dev.*, vol. 8, p. 3, 2020.

[7] K. Pohl and C. Rupp, *Requirements Engineering Fundamentals: A Study Guide for the Certified Professional for Requirements Engineering Exam - Foundation Level - IREB Compliant*, 1st. Rocky Nook, 2011, ISBN: 1933952814.

[8] J. Whittle, P. Sawyer, N. Bencomo, B. H. Cheng, and J.-M. Bruel, « Relax: incorporating uncertainty into the specification of self-adaptive systems », *in 2009 17th IEEE International Requirements Engineering Conference*, 2009, pp. 79–88. DOI: 10.1109/RE.2009.36.

[9] Y. U. Pabuccu, I. Yel, A. B. Helvacioglu, and B. N. Asa, « The requirement cube: a requirement template for business, user, and functional requirements with 5w1h approach », *International Journal of Information System Modeling and Design (IJISMD)*, vol. 13, *1*, pp. 1–18, 2022.

[10] M. Esser and P. Struss, « Obtaining models for test generation from natural-language-like functional specifications », *Proc. of 18th International Workshop on Principles of Diagnosis*, 2007.

[11] N. Mahmud, C. Seceleanu, and O. Ljungkrantz, « Resa tool: structured requirements specification and sat-based consistency-checking », *in 2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, 2016, pp. 1737–1746.

[12] D. Firesmith, « Specifying reusable security requirements. », *J. Object Technol.*, vol. 3, *1*, pp. 61–75, 2004.

[13] D. Firesmith, « Engineering security requirements », *The Journal of Object Technology*, vol. 2, 2003. DOI: 10.5381/jot.2003.2.1.c6.

[14] D. Firesmith, « Generating complete, unambiguous, and verifiable requirements from stories, scenarios, and use cases », *Journal of Object Technology*, vol. 3, *10*, pp. 27–39, 2004, ISSN: 1660-1769. DOI: 10.5381/jot.2004.3.10.c3. [Online]. Available: http://www.jot.fm/contents/issue_2004_11/column3.html.

[15] D. Firesmith, « Specifying good requirements », *Journal of Object Technology*, vol. 2, pp. 77–87, 2003. DOI: 10.5381/jot.2003.2.4.c7.

[16] A. Souag, R. Mazo, C. Salinesi, and I. Comyn-Wattiau, « Using the amanda method to generate security requirements: a case study in the maritime domain », *Requirements Engineering*, vol. 23, *4*, pp. 557–580, 2018.

[17] J. Jürjens, « Umlsec: extending uml for secure systems development », *in UML 2002 — The Unified Modeling Language*, J.-M. Jézéquel, H. Hussmann, and S. Cook, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 412–425.

[18] T. Lodderstedt, D. Basin, and J. Doser, « Secureuml: a uml-based modeling language for model-driven security », *in UML 2002 — The Unified Modeling Language*, J.-M. Jézéquel, H. Hussmann, and S. Cook, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 426–441, ISBN: 978-3-540-45800-5.

[19] L. Liu, E. Yu, and J. Mylopoulos, « Security and privacy requirements analysis within a social setting », *in Proceedings. 11th IEEE International Requirements Engineering Conference, 2003.*, 2003, pp. 151–161. DOI: 10.1109/ICRE.2003.1232746.

[20] H. Mouratidis and P. Giorgini, « Secure tropos: a security-oriented extension of the tropos methodology », *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, Apr. 2007. DOI: 10.1142/S0218194007003240.

[21] L. Apvrille and Y. Roudier, « Sysml-sec: a sysml environment for the design and development of secure embedded systems », *in APCOSEC 2013, Asia-Pacific Council on Systems Engineering, September 8-11, 2013, Yokohama, Japan*, IEEE, Ed., Yokohama, 2013.

[22] D. Munante, L. Gallon, and P. Aniorté, « An approach based on model-driven engineering to define security policies using orbac », Sep. 2013, pp. 324–332. DOI: 10.1109/ARES.2013.44.

[23] J. E. Hachem, V. Chiprianov, M. A. Babar, T. A. Khalil, and P. Aniorte, « Modeling, analyzing and predicting security cascading attacks in smart buildings systems-of-systems », *Journal of Systems and Software*, vol. 162, p. 110 484, 2020, ISSN: 0164-1212. DOI: https://doi.org/10.1016/j.jss.2019.110484. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0164121219302584.

[25] R. Fredriksen, M. Kristiansen, B. Gran, K. Stølen, T. Opperud, and T. Dimitrakos, « The coras framework for a model-based risk management process », Sep. 2002, pp. 94–105, ISBN: 978-3-540-44157-1. DOI: 10.1007/3-540-45732-1_11.

[26] C. Alberts, A. Dorofee, J. Stevens, and C. Woody, « Introduction to the octave approach », *Carnegie Mellon Institute*, 2003. [Online]. Available: http://www.itgovernanceusa.com/files/Octave.pdf.

[27] B. L. Paul Saitta and M. Eddington, « Trike v.1 methodology document [draft] », *Trike v.1 Methodology Document [Draft]*, 2003-2005. [Online]. Available: https://www.octotrike.org/papers/Trike_v1_Methodology_Document-draft.pdf.

[28] R. Stillions, « The dml model », *Ryan Stillions' Blog*, 2014. [Online]. Available: http://ryanstillions.blogspot.com/2014/04/the-dml-model_21.html.

[29] P. Manadhata and J. Wing, « A formal model for a system's attack surface », *in* Aug. 2011, pp. 1–28, ISBN: 978-1-4614-0976-2. DOI: 10.1007/978-1-4614-0977-9_1.

[30] S. Ouchani and G. Lenzini, « Attacks generation by detecting attack surfaces », *Procedia Computer Science*, vol. 32, pp. 529–536, 2014, The 5th International Conference on Ambient Systems, Networks and Technologies (ANT-2014), the 4th International Conference on Sustainable Energy Information Technology (SEIT-2014), ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2014.05.457. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050914006577.

[31] T. N. Sun, B. Drouot, F. R. Golra, *et al.*, « A Domain-specific Modeling Framework for Attack Surface Modeling », *in ICISSP 2020 : 6th International Conference on Information Systems Security and Privacy*, ser. Proceedings of the 6th International Conference on Information Systems Security and Privacy, vol. 1, Valetta, Malta, Feb. 2020, pp. 341–348. DOI: 10.5220/0008916203410348. [Online]. Available: https://hal.science/hal-02502387.

[32] D. Dolev and A. C. Yao, « On the security of public key protocols », *in 22nd Annual Symposium on Foundations of Computer Science (sfcs 1981)*, 1981, pp. 350–357. DOI: 10.1109/SFCS.1981.32.

[33] S. Jajodia, A. K. Ghosh, V. Swarup, C. Wang, and X. S. Wang, *Moving Target Defense: Creating Asymmetric Uncertainty for Cyber Threats*. New York, USA: Springer, 2011. DOI: 10.1007/978-1-4614-0977-9.

[34] National Institute of Standards and Technology, « Nist sp 800-30 risk assessment methodology », *Special Publication 800-30*, 2002.

[35] E. Peterson, « Dagger: modeling and visualization for mission impact situation awareness », *MILCOM 2016 - 2016 IEEE Military Communications Conference*, pp. 25–30, 2016. [Online]. Available: https://api.semanticscholar.org/CorpusID:31155935.

[36] F. Register, « Threat and hazard identification and risk assessment (thira)-stakeholder preparedness review (spr) unified reporting tool », *Federal Register*, 2022. [Online]. Available: https://www.federalregister.gov/documents/2022/01/20/2022-00986/agency-information-collection-activities-proposed-collection-comment-request-threat-and-hazard.

[37] U. of Illinois at Chicago, « Threat and hazards identification risk assessment », *University of Illinois at Chicago*, 2024. [Online]. Available: https://ready.uic.edu/planning/identification-and-risk-assestment/threat-and-hazard-identification-and-risk-assessment-thira/.

[38] H. E. M. Agency, « Threat and hazard », *Hawaii Emergency Management Agency*, 2023. [Online]. Available: https://dod.hawaii.gov/hiema/files/2023/01/21-1227-Threat-and-Hazard-Identification-and-Risk-Assessment-1.pdf.

[39] F. E. M. Agency, « National risk and capability assessment », *Federal Emergency Management Agency*, 2024. [Online]. Available: https://www.fema.gov/emergency-managers/national-preparedness/goal/risk-capability-assessment.

[40] O. H. Security, « Threat and hazard identification and risk assessment », *Oklahoma Homeland Security*, 2024. [Online]. Available: https://oklahoma.gov/homeland-security/critical-infrastructure/threat-and-hazard-identification-and-risk-assessment.html.

[41] F. E. M. Agency, « National risk and capability assessment », *Federal Emergency Management Agency*, 2024. [Online]. Available: https://www.fema.gov/emergency-managers/national-preparedness/goal/risk-capability-assessment.

[42] O. M. Christopher Osazuwa, « Confidentiality, integrity, and availability in network systems: a review of related literature », *International Journal of Innovative Science and Research Technology*, vol. 8, *12*, pp. 1946–1955, 2023.

[43] S. Samonas and D. L. Coss, « The cia strikes back: redefining confidentiality, integrity and availability in security », *in Proceedings of the 47th Hawaii International Conference on System Sciences*, 2014, pp. 4927–4936.

[44] S. Samonas and D. L. Coss, « Redefining confidentiality, integrity and availability in security », *ResearchGate*, 2014.

[45] DNV, *The three-pillar approach to cyber security: data and information protection*, `https://www.dnv.com/article/the-three-pillar-approach-to-cyber-security-data-and-information-protection-178315`, 2020.

[46] *The origin of the cia triad (confidentiality, integrity, availability) is unclear, but the underlying concepts have been used in military contexts for centuries*, No specific publication details available.

[47] P. Sawyer, R. Mazo, D. Diaz, C. Salinesi, and D. Hughes, « Using constraint programming to manage configurations in self-adaptive systems », *IEEE Computer Journal (cover feature)*, vol. 45, Oct. 2012. DOI: `10.1109/MC.2012.286`.

[48] F. Rossi, P. Van Beek, and T. Walsh, *Handbook of constraint programming*. Elsevier, 2006.

[49] R. Dechter, *Constraint processing*. Morgan Kaufmann, 2003.

[50] C. Correa, J. Robin, and R. MAZO, « Generating Constraint Programs for Variability Model Reasoning: A DSL and Solver Agnostic Approach », *in ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, Lisboa, Portugal, Oct. 2023. DOI: `10.1145/3624007.3624060`. [Online]. Available: `https://hal.science/hal-04330769`.

[51] H. Hnaini, R. Mazo, P. Vallejo, A. Lopez, J. Champeau, and J. Galindo, « Secret: a new security requirements specification template », *in Information Technology and Systems*, Á. Rocha, C. Ferrás, J. Hochstetter Diez, and M. Diéguez Rebolledo, Eds., Cham: Springer Nature Switzerland, 2024, pp. 235–246, ISBN: 978-3-031-54256-5.

[52] R. P. O'Brien, « An overview of the methodological approach of action research », 2008.

[53] R. Wieringa and A. Moralı, « Technical action research as a validation method in information systems design science », *in Design Science Research in Information Systems. Advances in Theory and Practice*, K. Peffers, M. Rothenberger, and B. Kuechler, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 220–238, ISBN: 978-3-642-29863-9.

[55] G. I. Susman, « Action research: a sociotechnical systems perspective », *Beyond method: Strategies for social research*, vol. 95, *113*, p. 95, 1983.

[56] E. Koshy, V. Koshy, and H. Waterman, *Action Research in Healthcare*. SAGE Publications, 2010, ISBN: 9781446247792. [Online]. Available: https://books.google.fr/books?id=Vb1w8mKAbScC.

[57] S. Jaiswal and D. S. Gupta, « Security requirements for internet of things (iot) », *Advances in intelligent systems and computing*, vol. 508, pp. 419–427, 2017.

[58] K. Großer, V. Riediger, and J. Jürjens, « Requirements document relations: a reuse perspective on traceability through standards », *Software and Systems Modeling*, vol. 21, Jan. 2022. DOI: 10.1007/s10270-021-00958-y.

[59] M. Kassab, O. Ormandjieva, and M. Daneva, « A traceability metamodel for change management of non-functional requirements », Sep. 2008, pp. 245–254, ISBN: 978-0-7695-3302-5. DOI: 10.1109/SERA.2008.37.

[60] G. Spanoudakis, A. Zisman, E. Pérez-Miñana, and P. Krause, « Rule-based generation of requirments traceability relations », *Journal of Systems and Software*, vol. 72, pp. 105–127, Jul. 2004. DOI: 10.1016/S0164-1212(03)00242-5.

[61] M. Khosrow-Pour, *Encyclopedia of Information Science and Technology, Third Edition*. IGI Global, 2014, ISBN: 9781466658899. [Online]. Available: https://books.google.fr/books?id=MJd_BAAAQBAJ.

[62] H. Hasrouny, A. E. Samhat, C. Bassil, and A. Laouiti, « Vanet security challenges and solutions: a survey », *Vehicular Communications*, vol. 7, pp. 7–20, 2017, ISSN: 2214-2096. DOI: https://doi.org/10.1016/j.vehcom.2017.01.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209616301231.

[63] ISO, « Iso/iec/ieee international standard - systems and software engineering – life cycle processes – requirements engineering », *ISO/IEC/IEEE 29148:2018(E)*, pp. 1–104, 2018. DOI: 10.1109/IEEESTD.2018.8559686.

[64] R. Stark, C. Fresemann, and K. Lindow, « Development and operation of digital twins for technical systems and services », *CIRP Annals*, vol. 68, May 2019. DOI: 10.1016/j.cirp.2019.04.024.

[65] A. B. Rebolj, « The case study as a type of qualitative research », *Journal of Contemporary Educational Studies*, pp. 28–43, Mar. 2013.

[66] P. Bicon, *De la memoire*. Paris: Serigraph, 1901.

[67] M. Astair, *La pendule et le fou*. Paris: Firgoult, 1895.

[68] P. Histrion and J.-M. Gaston, « Les phrases du début à la fin », *Revue internationale de linguistique*, vol. 50, pp. 52–98, 1925.

[69] F. Denys and H. Gellee, « Voyages en anatolie », *Revue des colonies*, vol. 13, pp. 45–85, 1905.

[70] F. Sbirre and J. Heloi, « Etudes épidémiologiques sur la paresse ambiante », *Revue du bon vieux temps*, vol. 23, *4*, pp. 53–102, 1906.

[71] E. Doule and M. Bady, « Les castes dans les indes britanniques », *Revue des pays lointains*, vol. 15, *7*, pp. 223–258, 1887.

[72] H. Kohler, *L'alsace, un paradis financier*. Paris: Hopla, 1903.

[73] G. Walther, *Dialogues sur la pluie et le beau temps*. Paris: Editeur local, 1896.

[74] G. Walther, *Controverses sur le soleil*. Paris: Editeur local, 1897.

[75] J. Laurent, *Constructions et ouvrages d'art*. Paris: Les arts et métiers, 1925.

[76] Z. Simon, *As-tu pensé à fermer le gaz?* Paris: Editions de l'oubli, 1903.

[77] K. R. Apt, *Principles of constraint programming*. Cambridge university press, 2003.

[78] J.-b. Gao, B. Zhang, X.-h. Chen, and Z. Luo, « Ontology-based model of network and computer attacks for security assessment », *Journal of Shanghai Jiaotong University (Science)*, vol. 18, pp. 554–562, Oct. 2013. DOI: 10.1007/s12204-013-1439-5.

Appendix A: Publications

Journals

— Hiba Hnaini, Raúl Mazo, Joël Champeau, Paola Vallejo, and Jose Galindo. E-score: A web-based tool for security requirements engineering. SoftwareX, volume 26, page 101704, 2024.

Conferences

— Hiba Hnaini, Raúl Mazo, Paola Vallejo, Andres Lopez, Joël Champeau, and Jose Galindo. Secret: A new security requirements specification template. In Álvaro Rocha, Carlos Ferrás, Jorge Hochstetter Diez, and Mauricio Diéguez Rebolledo, editors, Information Technology and Systems, pages 235–246, Cham, 2024. Springer Nature Switzerland

— Hiba Hnaini, Raúl Mazo, Paola Vallejo, Jose Galindo, and Joël Champeau. Taxonomy of Requirements Specification Templates. In SoftEng 23, Venice, Italy, April 2023

.

Appendix B: Implementation Details

Several tools were developed to validate our approach and hypothesis. Most of these tools are presented in this thesis. The next section provides details on each tool.

Requirements Specification Module

The requirements specification module is implemented as a group of VariaMos languages and NodeJS microservices. It has three main functionalities:

— Guiding the specification of requirements using the SECRET template and the SCORE ontology; implemented as an auto-complete microservice.
— Suggesting additional security requirements using the SCORE ontology; implemented as a microservice.
— Generating an editable file with the requirements; implemented as a microservice.

Languages

The languages Application Requirements AC and Domain Requirements AC, where AC stands for autocomplete, are defined using two JSON files. The first file is the abstract representation of the language which defines the syntax. The second is the concrete definition, which defines the graphics.

The abstract definition of the Application Requirements - AC language includes the definition of three types of requirements, functional, security, and other non-functional. It also defines the relationsships between the requirements. It is defined as follows, the full definition can be found in the online file [1]:

```
{
  "elements": {
  "FunctionalRequirement": {
```

1. https://drive.google.com/file/d/1Q7O-$_A$7FAT0HSOVctD06ysQIqnHfnD8q/view?usp = sharing

```
      "properties": [
      ...
      ]
    }
    "SecurityRequirement": {
      "properties": [
      ...
      ]
    },
    "NonFunctionalRequirement": {
      "properties": [
      ...
      ]
    }
  },
  "restrictions": {
   ...
  },
  "relationships": {
    "SecurityRequirement_SecurityRequirement": {
      "max": 9999999,
      "min": 0,
      "source": "SecurityRequirement",
      "target": [
        "SecurityRequirement",
        "NonFunctionalRequirement"
      ],
      "properties": [
        {
          "name": "Type",
          "type": "String",
          "possibleValues": "Generalization,Refinement,
              Evolution,Interactivity,Overlap,Conflicting,
              Rationalization,Contribution"
        }
      ]
```

```
    },
    ...

  }
}
```

The concrete definition of the language defines the graphics to be displayed for each element and relationship. In this definition, we indicate the endpoint to use for the autocomplete of each requirement type in the "autocomplete_services" field. These endpoints are the ones developed in the SECRET-SCORE approach. The full definition can be found in the online file[2].

```
    {
  "elements": {
    "SecurityRequirement": {
      "draw": ...,
      "icon": ...,
      "label": "SecurityRequirement",
      "width": 250,
      "design": "shape=SecurityRequirement",
      "height": 125,
      "label_property": "None",
      "autocomplete_services": [
        {
          "endpoint":
          "https://vms-requirements-autocomplete2024.
             azurewebsites.net/security/suggest/",
          "property": "Description"
        }
      ]
    },
    "FunctionalRequirement": {
        ...
      "autocomplete_services": [
        {
          "endpoint":
```

2. https://drive.google.com/file/d/1O8afGTvAfJFlKqpB0qEzIFcCjXmYIUdO/view?usp=sharing

```
            "https://vms-requirements-autocomplete2024.
                azurewebsites.net/security/suggest/functional",
            "property": "Description"
          }
        ]
      },
       "NonFunctionalRequirement": {
        ...
        "autocomplete_services": [
          {
            "endpoint":
            "https://vms-requirements-autocomplete2024.
                azurewebsites.net/security/suggest/functional",
            "property": "Description"
          }
        ]
      }
    },
    "relationships": {
      ...
    }
}
```

The JSON files for the abstract definition [3] and the concrete definition [4] for Domain Requirements - AC are defined similarly but using the endpoint specific for domain requirements.

Microsevices

Four microservices were developed to achieve the requirements specification module:

— **SCORE Ontology Microservice [5]:** The microservice queries the OWL file of the SCORE ontology and sends the response. An example of a SPARQL query is getting all the domains in the ontology as follows:

```
/**
* Method to get all domains in the ontology
```

3. https://drive.google.com/file/d/176lN6OiYhKm1LipRR$_h$$GW5cC1w2Hx6xh/view?usp = sharing$
4. https://drive.google.com/file/d/1KTE2C3AaxtXvw9VYmWkgdfjmIHwvkUPP/view?usp=sharing
5. https://github.com/variamosple/vms_escore_api/tree/main

```
     *
     * @return List of the domains
     */
    public List<String> getAllDomains() {
        try {
            if (model == null)
                init();
            String getDomainsQuery =
                    "PREFIX rdf: <http://www.w3.org
                       /1999/02/22-rdf-syntax-ns#>\n" +
                            "PREFIX  : <http://www.
                               semanticweb.org/user/
                               ontologies/2021/7/security-
                               criteria-ontology#>\n" +
                            "SELECT      ?ad\n" +
                            "WHERE       { ?ad rdf:type :
                               ApplicationDomain }";

            return executeQuery(getDomainsQuery, "ad");

        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }
        return null;
    }
```

These query methods are called by the endpoints of the microservice as follows:

```
@Path("/domain")
public class DomainResource {
//OntologyController api = new OntologyController();

    /**
     *
     * @return all domains
     */
```

```
@GET
@Main.Secured
@Produces(MediaType.APPLICATION_JSON)
public String getDomains() {
    return JSONArray.toJSONString(Main.getApi().
        getAllDomains());
}
```

To obtain data from the ontology, nine GET access points have been created.

- `/criteria`
  To obtain all security criteria in the ontology.
- `/criteria/{relation}/{criteria}`
  Obtain all criteria that have a given relation with a specified criterion.
- `/mechanism`
  To obtain all security mechanisms in the ontology.
- `/domain`
  To obtain all domains in the ontology.
- `/domain/{domain}/mechanisms`
  Obtain the security mechanisms of a specific domain in the ontology.
- `/domain/{domain}/criteria`
  To obtain the security criteria of a specific domain in the ontology.
- `/domain/{domain}/{criteria}`
  Obtain the security mechanisms of a specific security criterion in a specific domain of the ontology.
- `/domain/{domain}/{relation}/{criteria}`
  Obtain security criteria that have a given relation with a specified security criterion in a given domain of the ontology.
- `/domain/{domain}/{relation}/all`
  To obtain additional security criteria that have a given relation with all security criteria in a given domain of the ontology.

— **Auto-complete Microservice**[6]**:** The auto-complete service implements the parts of the template as JSON. It then uses these JSON definitions to detect their presence in the input text or the requirement where it suggests the next part or word accordingly. However, for the security criteria and security mechanisms, the service connects to the SCORE microservice to get the data. There are 4 endpoints for application requirements, application security requirements, domain requirements, and

6. https://github.com/variamosple/vms_requirements_autocomplete/blob/main/services/autocompleteServiceV2.js

application security requirements. An example of the JSON definitions would be the following, which describes the conditions part of the template.

```
    const conditions1 = {
"If": {
    "placeholder": "<Condition>",
    "next": "then"
},
"When": {
    "placeholder": "<Condition>",
    "next": "then"
},
"While": {
    "placeholder": "<Activation State>",
    "next": ""
},
"During": {
    "placeholder": "<Activation State>",
    "next": ""
},
"In Case": {
    "placeholder": "<Included Feature>",
    "next": "is included"
},
"After": {
    "placeholder": "<Behavior>",
    "next": ""
},
"Before": {
    "placeholder": "<Behavior>",
    "next": ""

},
"As soon as": {
    "placeholder": "<Behavior>",
    "next": ""
```

```
    }
};
```

— **Additional Security Requirements Microservice** [7] [8]**:** This microservice extracts the security criterion from the selected security requirement which is used to query the SCORE ontology microservice for its related security criteria. The service then reformulates the new security requirements using the additional security requirements. Finally, it automatically adds these requirements to the model.

```
relatedRequirement.properties.find(prop => prop.name ===
    "Description").value = newReq.requirement;
relatedRequirement.properties.find(prop => prop.name ===
    "Description").metadata.input = newReq.requirement;
relatedRequirement.properties.find(prop => prop.name ===
    "Description").metadata.secret.securityCriteria =
    newReq.criteria;
relatedRequirement.properties.find(prop => prop.name ===
    "Description").metadata.secret.securityMechanism = "";
if(newReq.priority=="High")relatedRequirement.properties.
    find(prop => prop.name === "Description").metadata.
    secret.priority = "shall";
if(newReq.priority=="Medium")relatedRequirement.
    properties.find(prop => prop.name === "Description").
    metadata.secret.priority = "should";
if(newReq.priority=="Low")relatedRequirement.properties.
    find(prop => prop.name === "Description").metadata.
    secret.priority = "could";
relatedRequirement.properties.find(prop => prop.name ===
    "StakeholderPriority").value = newReq.priority;
```

— **Requirements Documents Generation** [9]**:** The service extracts the requirements from the model and adds them to a word file. It also considers the difference between systems and software in file format.

7. https://github.com/variamosple/vms_requirements_autocomplete/blob/main/services/relatedRequirementsApplicationService.js
8. https://github.com/variamosple/vms_requirements_autocomplete/blob/main/services/relatedRequirementsService.js
9. https://github.com/variamosple/vms_requirements_autocomplete/blob/main/services/generateSRS.js

```
new Paragraph({
        children:env,
    }),
    new Paragraph({
        text: "System security",
        heading: HeadingLevel.HEADING_2,
        numbering: {
            reference: "my-crazy-numbering",
            level: 1,
        },

    }),

    new Paragraph({
        children:security,
    }),

    new Paragraph({
        text: "Information management",
        heading: HeadingLevel.HEADING_2,
        numbering: {
            reference: "my-crazy-numbering",
            level: 1,
        },
    }),
```

SERENA Models

The five SERENA Models are implemented in Variamos with an abstract definition and a concrete definition each. The transformations are implemented as a NodeJS microservice [10] that has five endpoints.

10. https://github.com/variamosple/vms_domain_application/blob/main/services/serenaTransformations.js

Criteria Model

The criteria model has abstract [11] and concrete [12] definitions of the security criteria (softgoals) and their relationships.

```
"elements": {
"SoftGoal": {
  "properties": [

  ]
}
},
```

Goal Model

It has the abstract [13] and concrete [14] definitions of goals, cardinalities, and operationalizations and their relationships.

```
"elements": {
"Goal": {
  "properties": [

  ]
},
"Cardinality": {
  "properties": [
    {
      "name": "From",
      "type": "String",
      "comment": "M in M:N"
    },
    {
      "name": "To",
      "type": "String",
```

11. https://drive.google.com/file/d/1PGtBYRK6d-thWKRCWe5Lq1BHwlTbgVIJ/view?usp=sharing
12. https://drive.google.com/file/d/1hHvxpwVkX7CFV9qfUjFCC71gMtMr7Lx3/view?usp=sharing
13. https://drive.google.com/file/d/1Rp43202e7Z1zkNUUDvoJKn23rDcbZydk/view?usp=sharing
14. https://drive.google.com/file/d/1bNvkGRIeQS0E$_0XniBNlBq2ZHJ5WFWNZ/view?usp$ = *sharing*

```
        "comment": "N in M:N"
      }
    ]
  },
  "Operationalization": {
    "properties": [

    ]
  }
```

Risk Model

It has the absract [15] and concrete [16] definitions of operationalizations, vulnerabilities, threats, security claims, and softgoals (security criteria) and their relationships.

```
  "elements": {
  "Threat": {
    "properties": [

    ]
  },
"SecurityClaim": {
    "properties": [

    ]
  },
  "SoftGoal": {
    "properties": [

    ]
  },
  "Vulnerability": {
    "properties": [

    ]
```

15. https://drive.google.com/file/d/1r1r6UWps4hVJKTebPHDiUQe$_e$*xSHuo7L/view?usp = sharing*
16. https://drive.google.com/file/d/11CJhmgkrvjgVHsM4m78jR4p4T4woQY62/view?usp=sharing

```
    },
    "Operationalization": {
      "properties": [

      ]
    }
  },
```

Treatment Model

It has the absract [17] and concrete [18] definitions of operationalizations, vulnerabilities, threats, security claims, security mechanisms, and softgoals (security criteria) and their relationships.

```
      "elements": {
    "Threat": {
      "properties": [

      ]
    },
    "SoftGoal": {
      "properties": [

      ]
    },
  "SecurityClaim": {
      "properties": [

      ]
    },
    "Vulnerability": {
      "properties": [

      ]
    },
```

17. https://drive.google.com/file/d/1-0oFRrBEYeqUYOgm-kNGnOGedkiwGUEZ/view?usp=sharing
18. https://drive.google.com/file/d/1ChsIxohKk5Y$_S$m1O$_s$19sXSpWaxF428o/view?usp = sharing

```
    "SecurityMechanism": {
      "properties": [

      ]
    },
    "Operationalization": {
      "properties": [

      ]
    }
  },
```

Overall Model

It has the absract [19] and concrete [20] definitions of goals, operationalizations, cardinalities, vulnerabilities, threats, claims, security mechanisms, softgoals (security criteria), context variables, soft influences, and their relationships. In addition to abstract and concrete definitions, the overall model has a semantic definition [21] that serves for the CLIF code generation.

```
      "relationPropertySchema": {},
  "elementTranslationRules": {
    "Goal": {
      "param": "F",
      "constraint": "(bool UUID_F)"
    },
    "ContextVariable": {
      "param": "C",
      "constraint": "(enum (Xs) UUID_C)",
      "enumMapping": {
        "var": "Xs",
        "attribute": "PossibleValues"
      }
    },
```

19. https://drive.google.com/file/d/1qfa1KJT90is46cE08Tf243X9l$_u$*MEN*0*n*/*view*?*usp* = *sharing*
20. https://drive.google.com/file/d/1JJDgP4JpjVrgWoYgwgXWGUxEqcJzvv$_q$/*view*?*usp* = *sharing*
21. https://drive.google.com/file/d/1R9pD-QbIwM-43zNWkGHEoQ$_G$*gB*2*TEsLA*/*view*?*usp* = *sharing*

```
    "Operationalization": {
      "param": "F",
      "constraint": "(bool UUID_F)"
    }
  },
```

COLLEGES SCIENCES
BRETAGNE POUR L'INGENIEUR
LOIRE ET LE NUMERIQUE

ENSTA BRETAGNE

Titre : Vers un cadre unificateur pour la spécification, formalisation et l'analyse d'architectures matérielles et logicielles sécurisées

Mot clés : Systèmes sécurisés, conception sécurisée, spécification, formalisation, analyse, modélisation multi-paradigme

Résumé : Dans le monde actuel, l'utilisation généralisée des systèmes numériques souligne l'importance de mesures de sécurité solides. Le concept de "Secure by Design" est devenu de plus en plus populaire en tant que stratégie proactive qui promeut l'incorporation des principes de sécurité au cœur de l'architecture du système. Cette thèse explore la spécification, la formalisation et l'analyse d'architectures matérielles et logicielles sécurisées dans le cadre du "Secure by Design". Elle aborde l'évolution du paysage de la sécurité en soulignant l'importance des mesures de sécurité proactives contre l'utilisation nominale et les menaces potentielles. Les ingénieurs et les architectes de systèmes sont invités à intégrer la sécurité de manière globale tout au long du cycle de vie du système, en abandonnant la notion dépassée de sécurité en tant qu'option ou exigence tardive. Alors que le concept de "Secure by Design" devient de plus en plus populaire, sa mise en œuvre efficace nécessite des investissements importants dans la recherche et le développement. Cela implique la création de théories, d'idées, de langages, de procédures, d'outils et de stratégies de sécurité fondamentales. L'aspect interdisciplinaire de la protection des systèmes contemporains souligne la nécessité d'une théorie cohérente et d'approches de conception structurées. Cette thèse propose un cadre de modélisation multi-paradigme pour la spécification, la formalisation et l'analyse des systèmes sécurisés. Elle vise à contribuer à la maturation de l'approche "Secure by Design" en fournissant une approche complète, ouvrant la voie à son adoption généralisée dans l'ingénierie des systèmes modernes. En outre, la thèse présente la mise en œuvre d'un outil basé sur ce cadre, offrant aux ingénieurs et aux architectes de systèmes un moyen pratique d'intégrer la sécurité dans la conception des systèmes. L'outil facilite la spécification, la formalisation et l'analyse, rationalisant l'adoption des principes de "Secure by Design" dans les pratiques d'ingénierie du monde réel. La thèse étend son champ d'application à l'évaluation de l'approche proposée et de l'outil mis en œuvre par le biais de tests techniques et d'essais par les utilisateurs, en documentant les résultats en vue d'une diffusion et d'une reproduction plus larges. En se concentrant sur une évaluation rigoureuse, le cadre démontre son applicabilité et son efficacité dans la sécurisation des systèmes.

Title: Towards a unifying framework for the specification, formalization and analysis of secure hardware and software architectures

Keywords: Secure systems, Secure by Design, Specification, Formalization, Analysis, Multi-paradigm modeling

Abstract: In today's world, the widespread use of digital systems highlights the importance of strong security measures. The concept of "Secure by Design" has become increasingly popular as a proactive strategy that promotes incorporating security principles at the core of system architecture. This thesis explores the specification, formalization, and analysis of secure hardware and software architectures within the framework of "Secure by Design." It addresses the evolving security landscape by emphasizing the importance of proactive security measures against nominal use and potential threats. Engineers and system architects are urged to integrate security comprehensively throughout the system life cycle, abandoning the outdated notion of security as an optional or late-stage requirement. As the concept of "Secure by Design" becomes more popular, its effective implementation requires significant investments in research and development. This involves creating theories, ideas, languages, procedures, tools, and fundamental security strategies. The cross-disciplinary aspect of safeguarding contemporary systems highlights the need for cohesive theory and structured design approaches. This thesis proposes a multiparadigm modeling framework for the specification, formalization, and analysis of secure systems. It aims to contribute to the maturation of "Secure by Design" by providing a comprehensive approach, paving the way for its widespread adoption in modern systems engineering. Furthermore, the thesis presents the implementation of a tool based on this framework, offering engineers and system architects a practical means to integrate security into system design. The tool facilitates specification, formalization, and analysis, streamlining the adoption of "Secure by Design" principles in real-world engineering practices. The thesis extends its scope to evaluate the proposed approach and the tool implemented through technical tests and user trials, documenting the results for broader dissemination and replication. By focusing on rigorous evaluation, the framework demonstrates its applicability and efficacy in securing systems.