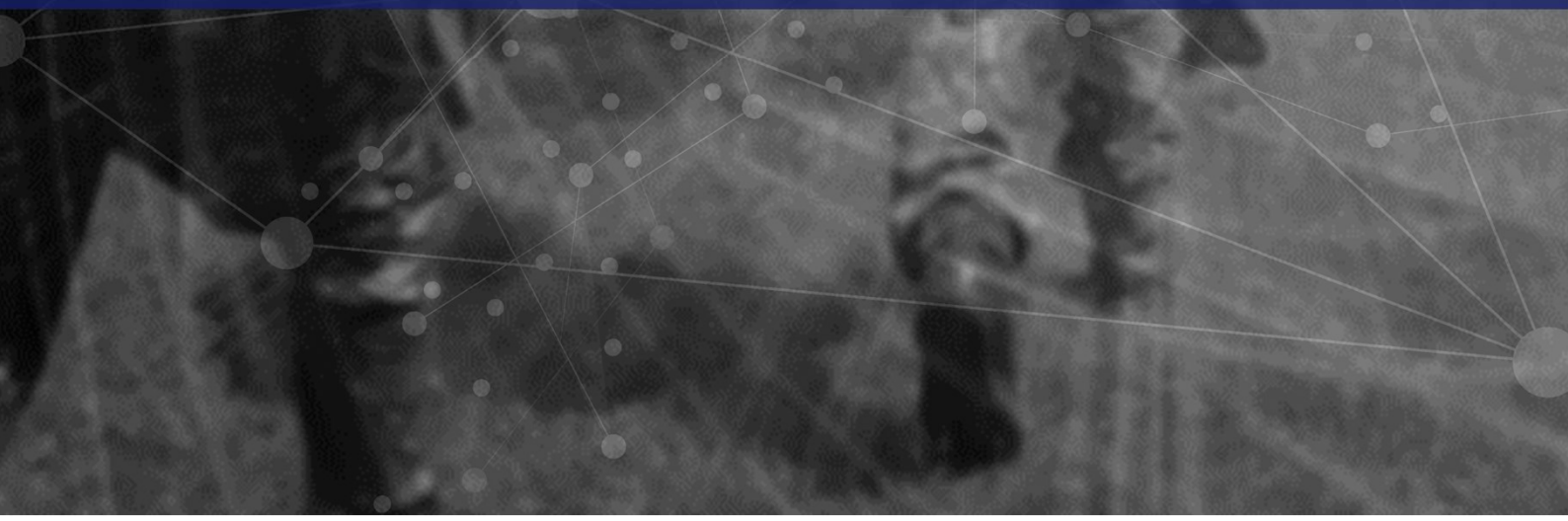
MODELLING SECURITY ARCHITECTURES OF COLLABORATIVE NETWORKS

MASTER THESIS

TJO-KIN MAN | MARCH 2022 | OPEN



UNIVERSITY OF TWENTE.

THALES

COLOPHON

Master Thesis

Title:	Modelling Security Architectures of Collaborative Networks
Date:	March 2022
Classification:	Open

Author

Name:	T. Man (Tjo-Kin)
E-Mail:	t.man@student.utwente.nl
Programme:	MSc Business Information Technology
Institute:	University of Twente PO Box 217 7500 AE Enschede The Netherlands

Graduation Committee

Name:	Prof. Dr. M.E. Iacob (Maria)
E-Mail:	m.e.iacob@utwente.nl
Organisation:	University of Twente
Faculty:	Behavioural, Management and Social Sciences (BMS)
Department:	Industrial Engineering and Business Information Systems (IEBIS)

Name:	Dr. M. Daneva (Maya)
E-Mail:	m.daneva@utwente.nl
Organisation:	University of Twente
Faculty:	Electrical Engineering, Mathematics and Computer Science (EEMCS)
Department:	Services and CyberSecurity (SCS)

Name:	Dr. S.M. Iacob (Sorin)
E-Mail:	sorin.iacob@nl.thalesgroup.com
Organisation:	Thales Nederland B.V.
Department:	Technical Directorate and Innovation

ABSTRACT

A Collaborative Network (CN) is an adaptive and scalable ecosystem in which heterogeneous, autonomous and asynchronous entities – originating from different security domains – are collaborating to achieve some common goals. This requires that security is asserted for both the atomic systems as well as the ecosystems which they are part of. To support the latter, this design science study presents a metamodel for CN security architectures, allowing vulnerabilities to be uncovered and attacks assessed. Using a fictional case on Collaborative Combat, it is illustrated how the generic security metamodel - including a modelling pattern and viewpoints - can be applied within the context of an architecture development methodology, security analysis methodology, modelling tool and modelling language. A small-scale evaluation with possible future end-users seems to indicate that the design artefacts have the potential for modelling relevant system security properties of CNs, but that further development is necessary.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

ADM	Architecture Development Method
CC	Collaborative Combat
C&C	Command and Control
CIS	Collaborative Information System
CMS	Combat Management System
CN	Collaborative Network
DSRM	Design Science Research Methodology
ECMA	European Computer Manufacturers Association
ESB	Enterprise Service Bus
IIS	Integrated Information System
MDE	Model Driven Engineering
NAF	NATO Architecture Framework
OGSA	Open Grid Security Architecture
OSI	Open Systems Interconnection
PC	Physical Component
SeaaS	Security as a Service
SOA	Service Oriented Architecture
SoS	System-of-Systems
SysML	Systems Modelling Language
TOGAF	The Open Group Architecture Framework
UML	Unified Modelling Language
UTAUT	Unified Theory of Acceptance and Use of Technology
VIEWS	VIsualising Enterprise-Wide Security

1 INTRODUCTION

In the past decades, enterprises tried to re-invent their businesses and maintained their competitive advantage by embracing collaborative structures with unified value propositions [1]. Although different manifestations of collaborations have emerged over time, e.g. virtual enterprises and professional virtual communities, they can be consolidated into the concept of Collaborative Networks (CNs). These are networks of autonomous organisations, people and resources that collaborate - based on agreed principles and interoperable infrastructures - with the purpose of achieving some common goals [2]. As a specialisation within the military domain, Collaborative Combat (CC) is a promising paradigm in which a military operation is carried out by different forces that interact with equipment from different sources, platforms and generations. Powered by digital revolution's technologies such as Artificial Intelligence, Data Analytics, Connectivity and Cybersecurity, CCs are aimed at instantaneous information sharing and processing, enabling continuous anticipation and offering strategic choices for engagement [3].

Underlying the concepts of CNs and CCs is the principle of *shared purpose* which is introduced by Max Weber as the basis for trust and organisational cohesion [4]. For CNs in general, and certainly for CCs in specific, this requires that the security is asserted for both the atomic systems as well as the ecosystems which they are part of. For atomic systems, security architectures are often embraced for this purpose, providing a holistic view of the organisational and system security [5]. For CN ecosystems, however, prior research shows that these systems have certain characteristics that introduce additional security challenges and architectural implications (see Subsection 2.1.1):

System Properties of Collaborative Networks

P1. Distributed entities: Entities are administered and controlled by independent or different organisations that have different security policies and mechanisms in place (e.g. [6, 7]).

P2. Collaborative: Entities cooperate with each other to carry out some tasks using exclusive or shared resources (e.g. [8, 9]).

P3. Heterogeneous entities: Entities have different overlapping and complementary capabilities that need to interoperate (e.g. [10, 11]).

P4. Adaptive and scalable ecosystem: Ecosystems can dynamically change in size and involved entities, and are flexible towards changing security policies and mechanisms (e.g. [12]).

P5. Autonomous and asynchronous entities: Entities act independently of each other while having limited to no mutual and environmental observability (e.g. [13]).

These system properties of CNs challenge confidentiality, integrity and availability objectives (see Subsection 2.1.2), causing the following security and architectural implications. The items marked with an asterisk (*) are different or new for CNs when compared to atomic systems:

General implications on security and architecture

G1. Mobile and platform entities: Entities are either mobile entities or platform entities [8]. Mobile entities are entities migrating through an ecosystem, accomplishing some tasks on behalf of their owners. If applicable, platform entities are stationary entities, hosting mobile entities while possibly also accomplishing some tasks on behalf of their owners [10, 14].

G2. Security domains*: Entities are organised in security-controlled domains [7] in which trust relations exist [9], propagating privileges and security knowledge [15] independent of individual security policies [16].

Intra-domain implications on security and architecture (within)

W1. Integrated centralised authorisation*: Mobile entity privileges are assigned and enforced by an authority at the security domain level [10]. This is based on security attributes and security policies that are specific to a certain interaction and between certain entities [17, 18]. Whenever possible, local authorisation and access mechanisms have to be preserved as much as possible [19].

W2. Adaptive policies*: Within a security domain, security policies can change based on entering and leaving entities [20, 21].

W3. Privilege delegation: Privileges can be delegated between mobile entities within a security domain. These privileges have temporal characteristics [12, 18].

W4. Integrated centralised accountability*: All intra-domain events have to be recorded by a domain-specific authority (i.e. auditing) [16, 20] in such a way that actions of specific entities can be proved (i.e. non-repudiation) [12, 22].

W5. Trust facilities*: For trust-centric solutions, a domain-specific trust authority is necessary for evaluating trust relations and subsequent trust decisions [23]. This can be neglected in case direct trust is assumed between all entities [10, 13].

Inter-domain implications on security and architecture (between)

B1. Information exchange*: Security information of entities and their domain-specific security policies have to be exchanged between security domains [16, 20].

B2. Privilege delegation*: Privileges can be delegated between entities of different security domains. These privileges have temporal characteristics [19].

B3. Federated decentralised authentication*: The identities and privileges of entities are asserted [16] federatively between security domains [9]. This includes expressing identities in a uniform way [18, 19], allowing cross-domain validation of security attributes [20].

B4. 3rd-party or trust-based accountability*: In theory, all inter-domain events are ideally recorded by third parties [14] (i.e. auditing) in such a way that actions of specific

entities can be proved (i.e. non-repudiation) [19]. However, in most cases, trust-based accountability has to be adopted as an alternative because of the dynamic nature of collaborative ecosystems [10].

Because of these security and architectural implications, it is not always possible to assert the security of whole CN ecosystems based on the security properties of the atomic systems. While different security architecture frameworks exist that describe security services and underlying mechanisms (e.g. [16] and [20], see Subsection 2.2.1), they do not provide specific metamodels or modelling guidelines. Hence, the goal of this novel study is to integrate security properties within CN system architectures, asserting security and promoting security by design. More specifically, the goal of this study is to develop a metamodel that supports the creation of CN security architectures and to illustrate how the metamodel can be applied to assert the security of CNs.

This study assumes that all collaborating human entities are behaving benevolently (intentionally and unintentionally), that the security architectures of the atomic systems are known and can be considered secure, and that direct trust exists within CN ecosystems. This allows the security architectures of atomic systems to be used as a starting point for the creation of CN security architectures - without the introduction of trust authorities (see W5) and without considering human interventions. For the metamodel, constructs and relationships have to be defined. While these modelling elements can be defined by adapting an existing security metamodel (e.g. [24] and [25], see Section 2.3) based on the security and architectural implications as presented above, the value of the resulting metamodel is determined by its utility. As the purpose of the design artefact is to assert the security of CNs, the utility value of the metamodel should be determined based on its capability to model systems such that their . Hence, this study starts with an exploration of vulnerable system parts, related attacks and vulnerabilities, giving the first research question:

RQ1 What are relevant vulnerabilities of Collaborative Networks?

RQ1.1 What are vulnerable parts of CN ecosystems?

RQ1.2 What are attacks on and vulnerabilities of these system parts?

The constructs and relationships are defined as an extension to the existing modelling elements of Eclipse Capella [26] and its Capella Cybersecurity viewpoint [27]. Eclipse Capella is an open-source and MDE-ready [28] solution that is adopted in various industrial domains [29] for modelling i.a. system architectures. The built-in modelling language is similar to SysML [30] while the diagrams and constructs are inspired by the UML [31] and SysML standards [32]. As an add-on, the Capella Cybersecurity viewpoint [27] provides limited means for modelling security concerns, mechanisms, policies and threats of complex componentised systems. This study extends the existing modelling elements of Eclipse Capella to support modelling CNs system architectures. The Capella Cybersecurity viewpoint [27] is extended to model the security properties on top of system architecture descriptions - realising security architecture descriptions. In addition, a modelling pattern is suggested that illustrates the intended use of the CN metamodel and viewpoints are presented for modelling specific security aspects. This gives the second research question and subquestions:

RQ2 How does a metamodel for CN security architectures look like?

RQ2.1 What are relevant constructs and relationships for modelling CN system architectures?

RQ2.2 What are relevant constructs and relationships for modelling security properties within CN system architectures, realising CN security architectures?

RQ2.3 What are relevant modelling patterns and viewpoints of the metamodel for CN security architectures?

Collaborative Combat is adopted as the main application area, illustrating how the metamodel can be applied to assert the security of CNs by design. More specifically, this study shows how the metamodel can be applied to create security architectures within the context of the Arcadia architecture development methodology [33]. This methodology is the natural choice as it is created in conjunction with the Eclipse Capella tool. The methodology covers the environmental, developmental and integrational considerations similar to the TOGAF ADM [34] and the NAF development methodology [35] (see Subsection 2.2.2). Even though this methodology does not include post-integration considerations, the scope is sufficiently broad as the main objective of this study is to model the security properties of CN ecosystems and not the governance or migration aspects thereof. Furthermore, this study illustrates how security analysis techniques (see Subsection 2.2.3) can be applied to assess the level of security in system architectures that are created based on the CN metamodel. This gives the following research question and subquestions:

RQ3 How can the metamodel for Collaborative Networks be applied to assert the security of CC architectures by design?

RQ3.1 How can the security metamodel for Collaborative Networks be used within the context of the Arcadia architecture development methodology?

RQ3.2 How can the architectures that are created with the CN metamodel be analysed for their level of security?

For this study, the Design Science Research Methodology (DSRM) of Peffers, Tuunanen, *et al.* [36] is embraced and the chapters are organised accordingly. This chapter defines the research problem and justifies the intended solution (i.e. the metamodel) in terms of its utility value. Chapter 2 presents the related work based on which objectives of the metamodel are directly or indirectly determined. Similarly, objectives of the metamodel are determined based on Chapter 4 and Chapter 5, presenting a typical CN case on CC and attacks/vulnerabilities that are relevant to CNs, respectively. Chapter 6 shows the design and development activities that results in the CN metamodel as well as its patterns and viewpoints. Chapter 7 illustrates how the metamodel can be applied to create security architectures within the context of the Arcadia architecture development methodology and how security analysis techniques can be applied to assess the level of security in these architectures. Chapter 8 assesses to what extent the design artefacts can be used to model relevant system and security properties of CNs such that their vulnerabilities can be uncovered and attacks assessed. Finally, Chapter 9 concludes and discusses the results of this design science study.

2 RELATED WORK

2.1 System and Security Properties

Throughout the years, different research paradigms emerged that discuss evolutions of security-oriented system models, each providing system properties, system elements, classes of security threats and security risk concerns that are relevant to CNs.

2.1.1 High-Level System and Threat Models

In the mid-1980s, the European Computer Manufacturers Association (ECMA) developed a reference model for open distributed systems. Characterising for such systems is their distributed nature, referring to systems in which entities are administered and controlled by independent or different organisations that have diverse security policies and mechanisms in place [6]. The main security challenge is to protect the communication between entities, independent of their security policies and mechanisms [16]. For this purpose, the concept of security domains is introduced, representing groups of security-controlled entities [7], propagating trust, privilege and security knowledge within open distributed ecosystems [15].

As a direct successor that emerged around the turn of the century, mobile agent systems are ecosystems in which mobile agent entities are migrating through a network of hosts, accomplishing tasks on behalf of their owners [10]. These systems are having collaborative properties, referring to multiple entities cooperating with each other to carry out some tasks using exclusive or shared resources [8]. As agent entities are mobile within a distributed context, they are performing actions independently of each other while having limited to no mutual and environmental observability [13, 14]. This causes heterogeneity within ecosystems, referring to entities having different overlapping and complementary capabilities that needs to interoperate [10],
Because of these system properties, not only secure data transfers within security domains have to be considered. As illustrated by Karnik and Tripathi [17], there is a need to secure agent entities, host platforms, and their (communication, execution and calling [14]) relationships. More specifically, threats can be classified as agent against host platform, agent (host) platform against agent, agent against agent, and other entities against agent system [8].
In conjunction with cryptography-based solutions, trust-centric solutions can be applied, providing more certainties regarding the identity and intentions of mobile agents [22]. Three trust types can be distinguished: direct trust, recommended trust and derived trust [13]. For the last two types of trust, a trusted authority on the security domain level

has to be introduced which evaluates authentication, execution and mobile code trust relations [10], and subsequent trust decisions [23]. Both types of security solutions respond differently to internal and external system exploitations or damages [37].

A more recent paradigm that is evolved in parallel with mobile agent systems is the paradigm of virtual organisations - organised in open grid systems. A virtual organisation is a collection of individuals and organisations that establishes a security domain in which trust relations exist between their entities and services [9]. Such virtual organisations have the purpose of collaboratively providing services to entities that are part of other security domains [11]. Hence, while research on mobile agent systems focuses on entities migrating through networks within a security domain, this paradigm focuses on ecosystems that span multiple security domains [21, 38]. Built on open distributed systems, virtual organisations are enabled by grid technologies, realising heterogeneous, autonomous and asynchronous open grid systems. As open grid systems can change in size and involved entities, they have to be flexible towards changing security polities and mechanisms [12]. Threats in such adaptive and scalable ecosystems can be classified into threats from malicious grid design, threats during data exchanges or application runs (i.e. critical states), and threats during non-critical states [39].

The paradigms of mobile agent systems and open grid systems can be combined into mobile grid systems. In fact, this last type of system extends the concept of virtual organisations by allowing entities to instantaneously migrate to other (real or virtual) organisations [40]. This paradigm does not introduce new system properties but does introduce security dimensions with regards to mobility [19].

2.1.2 Security Risk Concerns

From a high level, all these paradigms have the same set of security risk concerns in common, often referred to as CIA & AAA. These concerns can be directly mapped to classes of threats as classified by the STRIDE model [41], allowing system models to be decomposed and analysed for their susceptibility to threats.

The CIA concerns (i.e. Confidentiality, Integrity and Availability) can be considered as higher-level security objectives [42]. While confidentiality refers to ensuring that only authorised entities can access certain other entities or perform certain (inter)actions, integrity refers to protecting these entities and interactional transmissions from illegitimate alterations or modifications. Availability refers to assuring access to entities and interactions when needed. These three concerns relate to the STRIDE threats Information disclosure, Tampering and Denial of service.

The AAA concerns (i.e. Authentication, Authorisation and Accountability) are supporting the CIA objectives [42], guiding processes used for protecting entities and data [43]. Due to their system properties, CNs have specific implications on the AAA concerns. Authentication is perhaps the most fundamental and mostly discussed security risk concern [14], referring to asserting the identity and authenticity of entities performing certain (inter)actions. This relates to the STRIDE threat Spoofing. As CNs have distributed

and collaborative properties, entities involved in any interaction might be malicious [17]. Hence, there is a need to ensure the integrity and confidentiality of these entities and their (inter)actions [10, 22]. Because of the heterogeneous character of CNs, authentication should likely be implemented by a federation model [9] with a uniform way of expressing identities [18, 19].
Authorisation is commonly mentioned together with access control, referring to defining privileges of entities, and controlling access to entities or interactions. This relates to the STRIDE threat Elevation of privilege. Because of the distributed properties of CNs, there is a need for an authority on the security domain level [10], assigning and revocating rights to/of entities based on their authenticated identity [17, 18]. As CNs are adaptive and scalable, delegation and freshness of privileges have to be considered [12] while preserving local authorisation and access mechanisms as much as possible [19]. In some cases, sandboxing mechanisms can be applied, creating very restricted environments in which actions are strictly controlled [8, 14]. Any authorisation beyond the defined privileges may cause malicious or unintentional violations of confidentiality, integrity and availability [42].
Accountability refers to recording security-relevant events (i.e. auditing) and proving that (inter)actions are performed by specific entities (i.e. non-repudiation). This relates to the STRIDE threat Repudiation. For distributed systems in general, and autonomous and asynchronous systems in specific, accountability allows breaches of confidentiality, integrity and availability to be monitored and investigated [22]. Hence, this is often part of security assurance and policy enforcement [9, 19], possibly performed by third parties [14].

2.2 Security Architecture Frameworks

Based on the security-oriented systems models, different architecture frameworks were created by both industry communities and academics. These frameworks establish common practices for creating, interpreting, analysing and using security architectures [44] that address one or more security risk concerns. From literature, three types of architecture frameworks can be distinguished: collections of security services in Service Oriented Architectures (SOAs), architecture development methodologies and architecture security analysis frameworks.

2.2.1 Service Oriented Security Architectures

Constitutive to many architecture frameworks is the Open Systems Interconnection (OSI) reference model [45] which prescribes communication operations in seven abstraction layers, promoting system interoperability without regarding underlying system structures. To ensure system security and secure data transfers, the OSI security architecture [46] provides a set of security services along with supporting security mechanisms and governing security policies. This realises Service Oriented Architectures (SOAs) consisting of collections of decoupled and composable services, implementing one or more security mechanisms that contribute to system security [47].

For each of the following OSI-based system paradigms, a SOA-based security architecture framework exist of which the aggregated services and cross-service mechanisms are presented on top of Table 1.
For open distributed systems, the ECMA [16] presented security services related to security information, security control and security monitoring. While authentication, security attribution and interdomain services allow entities to be verified and security attributes to be exchanged between security domains, secure association and authorisation services allow sessions with end systems to be created where entities have certain privileges. In addition, three cross-service mechanisms support recovery, cryptography and user processes [6].
For mobile agent systems, Reiser and Vogt [14] created a security architecture in which trust centre and cryptographic mechanisms run on host platforms while security services run within a sandbox of mobile agents. Sandboxing prevents other agents from unauthorised access and prevents any action out of control of host platforms. The rights delegation service is introduced to allow mobile entities to migrate from host to host with temporarily granted privileges. Although interdomain and association are not included within the scope of this framework, such services are likely crucial for connectivity purposes.
For open grid systems, the Global Grid Forum created the Open Grid Security Architecture (OGSA) [20] based on which Demchenko, Gommans, *et al.* [21] created a layered security architecture framework. Because of the adaptive and scalable properties of open grid systems, security services have to concurrently enforce security policies across security domains while establishing trust relationships by federating security mechanisms [18]. This results in additional cross-domain considerations for many existing security services. Furthermore, services for converting credentials and mapping identities are introduced along with a mechanism to support changing security policies.

Apart from these cryptography-based frameworks, MobileTrust [10, 13] is introduced as a layered security framework for trust-centric solutions in mobile agent systems. Next to the security management layer, which provides security services similar to those in cryptography-based models, a trust management layer is introduced that provides trust services to the security management layer. This layer consist of a trust decision service, valuating trust relations based on a trust policy base with support of trust inference and management services.
Furthermore, in contrast to frameworks that are based on endpoint security, Hafner, Memon, *et al.* [47] suggested a reference architecture that realises Security as a Service (SeaaS), addressing security concerns through a single component crossing different security domains. Central in this model is an Enterprise Service Bus (ESB) to which a SeaaS server and a security server are connected. The SeaaS server contains a service that evaluates security based on security policies and on cryptography-based services that are employed on the security server.
The services of these security-enriching frameworks are listed on the bottom of Table 1.

		Open Distr.	Mobile Agents	Open Grid
CRYPTOGRAPHY-BASED SECURITY SERVICES				
Authentication	Verify credentials of entities and provide identity information.	X	X	X
Security attribution (authentication)	Based on identity information, return privileges and other security attributes that can be validated by different security domains.	X	X	X
Authorisation and access control	Using security attributes, make access control decisions based on security policies.	X	X	X
Rights delegation (authorisation)	Delegate privileges for a certain period of time and enforce them.		X	X
Accountability	Record all security activities over time (i.e. auditing), creating permanent proof of actions (i.e. non-repudiation).	X	X	X
Interdomain	Establish sessions between security domains and control the exchange of security/policy information.	X		X
Secure association	Create associations between entities, verify authorisations and apply communication/transport security.	X		X
Credential conversion	Converting credentials from one type to another (between security domains).			X
Identity mapping	Associating identities of different security domains.			X
CRYPTOGRAPHY-BASED CROSS-SERIVCE SECURITY MECHANISMS				
Security recovery	A set of rules on how to react to any (suspected) breach of security and how to recover thereafter.	X		
Cryptographic support	Cryptography facilities for information protection.	X	X	
Subject sponsor	Facilities for mediating user interactions, incl. profiling.	X		X
Sandboxing	Restricted environments in which actions are controlled.		X	
Security policies	Facilities for managing changing security policies.			X
TRUST-CENTRIC SECURITY SERVICES				
Trust decision	Determine the trustworthiness of entities and their behaviour for a certain domain and time.			
Trust inference	Evaluate trust relations based on direct trust, recommended trust and derived trust.			
Trust mngmt.	Updating trust and recommendation protocols.			
SECURITY AS A SERVICE SERVICES				
SeAAS	Evaluate security based on security policies and using to crypto-based security services.			

Table 1: Analysis of Security Services and Mechanisms

2.2.2 Architecture Development Methodologies

For guiding architecture development processes, different methodologies exist that have complementary elements.

As the core of The Open Group Architecture Framework (TOGAF), the Architecture Development Method (ADM) [34] provides a recommended sequence of iterative phases for establishing an architecture framework, developing architecture content, and planning transition, implementation and management activities. Security and risks aspects can be integrated within the TOGAF ADM. This includes the establishment of the security and business context; the creation of trust frameworks, risk assessments and security service catalogues; the implementation of security mechanisms; and the planning of risk governance, monitoring and mitigations [48].
Similarly, the NATO Architecture Framework (NAF) [35] describes a methodology that outlines an approach for architecture governance, management, description and evaluation within the military and business context. Inspired by the TOGAF ADM, the stages include the establishment of an architectural landscape and vision; the description and evaluation of architecture alternatives; the development of migration plans; and the governance of, decision-making on and monitoring of architecture changes.
The NAF methodology does not have specific guidelines for creating security architectures. However, in contrast to TOGAF, NAF is intended to be used in the military domain and is created to support the development of system architectures. TOGAF is preliminarily aimed at creating enterprise architectures within the business domain.

A supplementary methodology for integrating security within architecture models is the Model Driven Engineering (MDE) approach. In this approach, system security architectures are part of larger system development processes in which the architecture models are used for the semi-automated construction of systems [49]. The main advantage is that security concerns can be defined and integrated during early stages of (high-level) system design [24], and that the mapping between security concerns and their design realisations can be verified [50].
For mobile grid systems, Rosado, Fernandez-Medina, *et al.* [19] suggested a MDE system development process that consists of a planning phase, a development phase (including analysis, design and construction) and a maintenance phase. For the design tasks within the development phase, five activities were defined. This includes designing the mobile grid architecture, the security architecture (with security services) and the integration between them; followed by specifying the resulted security architecture, and the validation and verification thereof.

A methodology that is adopted in both military and non-military domains is the Arcadia methodology [33]. Originally developed by Thales for creating system, hardware and software architectures, this MDE-ready methodology distinguishes three interrelated activities. This includes the analysis and modelling of operational, functional and non-functional needs; the creation and validation of a logical and a physical architecture; and the management of requirements. While logical architectures show how a system fulfils user needs, physical architectures show how system development, integration,

verification and qualification processes look like. In contrast to the TOGAF ADM and the NAF [35] methodology, post-integration considerations such as architecture migration and governance descriptions are not part of this methodology.

2.2.3 Security Analysis

With regards to security analysis, the level of security in system architectures can be analysed both qualitatively and quantitatively. For qualitative analyses, threat models such as the STRIDE model can be applied, manually assessing architectures for different classes of threats. For this purpose, the Mitre Corporation created the very extensive ATT&CK cybersecurity framework [51], organising threats into tactics, (sub) techniques, procedures and mitigations. These aspects refer to the tactical objectives of threats, the ways to achieve them, the real-world instances of threats, and the mitigation techniques therefor [52].
A more quantitative approach for analysing information security is suggested by Grusho, Grusho, *et al.* [53]. In this approach, security architectures are hierarchically decomposed into subsystems of subjects that perform some actions on objects. By mapping, valuating and weighting all possible actions on each combination of indivisible objects and subjects, the security efficiency can be calculated. This approach is based on a process model, similar to the O-PDRR architecture reference model as presented in [54]. More specifically, the security efficiency is determined by the capacity of the monitoring function to detect parameters of actions (Detect), the capacity of the management function to analyse and classify security risks (React), the capacity of the control function to define security mechanisms (React), and the capacity of the security mechanisms function to decrease security risks (Protect). The system Recovery capabilities are not considered in the security efficiency score but might be relevant.
However, as CN ecosystems are highly complex systems, such an extensive quantitative approach seems to be not reasonably achievable for larger systems as it requires systems to be decomposed until all objects, actions and subjects are indivisible. Instead of analysing the security efficiency of whole CN ecosystems, it is also possible to limit the security analysis to identifying the weakest links and threats that have the highest business impact within an ecosystem. For this purpose, Breu, Innerhofer-Oberperfler, *et al.* [55] suggested a security analysis framework that shows how layered threat graphs can be used to quantify the ratio of attacks that results in successful breaches of security requirements; the propagation effect of successful attacks; and the business losses caused by failure to fulfil security objectives.

2.3 Architecture Modelling Languages

While an architecture embodies the fundamental concepts, properties and relationships of a system in its environment, an architecture description is a work product used to express the architecture [44]. Three modelling languages prevail for which different extensions exist, adding security considerations into architecture descriptions. In some cases, viewpoints are presented as well, establishing ”conventions for the construction, interpretation and use of architecture views to frame specific system concerns” [44].

Originated from the field of software engineering, Unified Modelling Language (UML) [31] is considered the de-facto standard for object-oriented modelling. Using a set of constructs, embodied in formal diagrams, the structural, behavioural and interactional properties of software systems can be graphically depicted. UMLsec, SecureUML and Security4UML are UML profiles, defining new constructs, tagged values and constraints relevant to security architectures.
UMLsec [56] is created with the purpose of annotating, and facilitating the analysis and formal verification of CIA & AAA concerns within UML models. In contrast, SecureUML [50] is limited to specifying concerns related to authorisation but it allows role-based access control mechanisms and policies to be modelled, specifically in the context of distributed systems. Security4UML [49] is similar in scope as SecureUML but focuses exclusively on UML class diagrams of software architectures with additional considerations for signed/encrypted resources and security protocols. Both SecureUML and Security4UML support MDE while UMLsec does not.

Originally developed as a UML profile, Systems Modelling Language (SysML) [30] has become a widely adopted modelling language in the field of systems engineering, specifying requirements, structures, behaviours, allocations and constraints of complex systems.
Created for embedded systems, SysML-Sec [57] is an MDE extension which integrates confidentiality, integrity and authentication concerns into system models. The extension includes a new diagram for modelling security requirements and their dependencies, and extends the SysML parametric diagram with an attack construct for modelling threats and their logical or temporal causal relationships.
Created for connected System-of-Systems (SoS), SoSSec [24] facilitates the discovery, analysis and resolution of cascading attacks in the early stages of architecture development. The MDE extension introduces the goal and threat constructs within the SysML block diagram, connecting this diagram to the new goal and security diagrams that depict (security) goals and threat pre/post conditions, respectively.
To model threats more generally, Ruiz, Harjani, *et al.* [58] proposed two metamodels for UML/SysML diagrams. While the core security metamodel presents threats and tests based on security requirements on the conceptual level, the threat model presents threats and tests on the instance level which includes their attributes.
Created for the Arcadia methodology, Eclipse Capella is a solution [26] for modelling i.a. system architectures. The modelling language used in this tool has many similarities with SysML [59], and the constructs and diagrams are mainly inspired by the UML and SysML standards [32]. In the Capella Cybersecurity viewpoint [27], additional constructs and diagrams are introduced, allowing CIA concerns and threats to be modelled. The diagrams include a threat diagram; an architecture diagram allocating functions to components; an exchange scenario diagram presenting information flows, and an information asset diagram.

Created for describing, analysing and communicating enterprise architectures, ArchiMate [60] is a modelling language in which constructs are classified based on layers (representing enterprise levels) and aspects (representing active, behaviour and passive structures).

To model the organisational security of multi-agent systems, Blangenois, Guemkam, *et al.* [25] created an ArchiMate specialisation in which agent services (i.e. behaviour structures) are governed by responsibility driven policies, giving active structural elements certain rights and obligations with respect to the passive structural elements. The metamodel is adapted such that it allows each agent operation to be mapped with a policy execution.
To model threats within ArchiMate, Band, Engelsman, *et al.* [61] created an extension to depict the interactions between assets at risk and the implementations of control measures. This includes the modelling of security requirements, control measures and vulnerabilities.
Alternatively, VIsualising Enterprise-Wide Security (VIEWS) [62] is a modelling language specifically developed for creating architecture descriptions that visualise the AAA security features of distributed systems and their environment, supporting the detection of vulnerabilities. This includes the modelling of important system constructs, the position of security mechanisms and services, and the security-relevant information flows.

In Table 2, a summary of the main characteristics of the discussed modelling languages is presented. From this table, it can be concluded that none of the discussed modelling languages or extensions has the capabilities to model all AAA security concerns as well as security mechanisms, policies and threats of CNs while supporting MDE.
Developed by the Mitre Corporation, VIEWS is the modelling language that, in theory, has the most sophisticated properties when it comes to modelling security architectures of CNs. However, a severe limitation of this language is the isolated foundation, creating security models separately from other architecture models without any means of integration. Hence, this modelling language will not likely fit in any architecture development methodology in which security is only one of the considerations.
The SecureUML profile is the next modelling language that has, in theory, the best properties. However, this profile is mainly aimed at distributed software architectures rather than distributed system architectures. Furthermore, the profile is limited to only modelling authorisation constraints and mechanisms.
With regards to the ArchiMate, this modelling language is originally created for atomic enterprises and the modelling of enterprise architectures. Even though an extension exists for multi-agent systems, the language is not a good candidate as it lacks cross-domain considerations.
Of all families of languages, SysML seems to be the most suitable one for modelling CN security architectures. By its foundation, the language supports the componentisation of complex systems, suitable to be extended for modelling distributed systems. Especially the SoSSec profile and Capella Cybersecurity viewpoint are interesting candidates, already having limited capabilities in modelling security mechanisms and policies. A slight advantage of the Capella Cybersecurity viewpoint over the SoSSec profile is that the Capella Cybersecurity viewpoint is already implemented in an open-source tool, Eclipse Capella, that is adopted in various industrial domains [29].

	UML			
	UMLsec	**SecureUML**		**Security4UML**
AAA concerns	Authentication Authorisation Accountability	Authorisation		Authorisation
Modelling security concerns	X	X		X
Modelling security mechanisms and policies		X		X
Modelling threats	(Limited)			
Support MDE		X		X
Support CN		X		
	SysML			
	SysML-Sec	**SoSSec**	**Ruiz e.a.**	**Capella cybersecurity viewpoint**
AAA concerns	Authentication	?	Authentication Authorisation Accountability	?
Modelling security concerns	X	X	(Limited)	X
Modelling security mechanisms and policies		(Limited)		(Limited)
Modelling threats	X	X	X	X
Support MDE	X	X		X
Support CN	(Limited)	(Limited)	(Limited)	(Limited)
	ArchiMate			**VIEWS**
	Blangenois e.a.	**Band e.a.**		
AAA concerns	?	?		Authentication Authorisation Accountability
Modelling security concerns		X		X
Modelling security mechanisms and policies	X	(Limited)		X
Modelling threats		X		
Support MDE				
Support CN	(Limited)			X

Table 2: Analysis of Architecture Modelling Languages

3 METHODOLOGY

3.1 Synopsis of the Design Science Methodology

For this study, the Design Science Research Methodology (DSRM) of Peffers, Tuunanen, *et al.* [36] is embraced. This methodology provides a nominal but non-rigid iterative process which consists of six activities that constitute a design science study:

- Activity 1: Problem Identification and Motivation
- Activity 2: Define the Objectives for a Solution
- Activity 3: Design and Development
- Activity 4: Demonstration
- Activity 5: Evaluation
- Activity 6: Communication

As the goal of this study is to develop a metamodel that supports the creation of CN security architectures and to illustrate how this metamodel can be applied to assert the security of CNs, the entry point of this study is *design and development centred initiation* (i.e. activity 3). The following section elaborates on each of the activities and presents the alignment with both the research questions as well as the structure of this study. A summary of the research methodology is graphically depicted in Figure 1. While the right part presents a sequence with the main design activities (e.g. activity 3, 4 and 5), the left part presents the knowledge components the design activities are built upon.

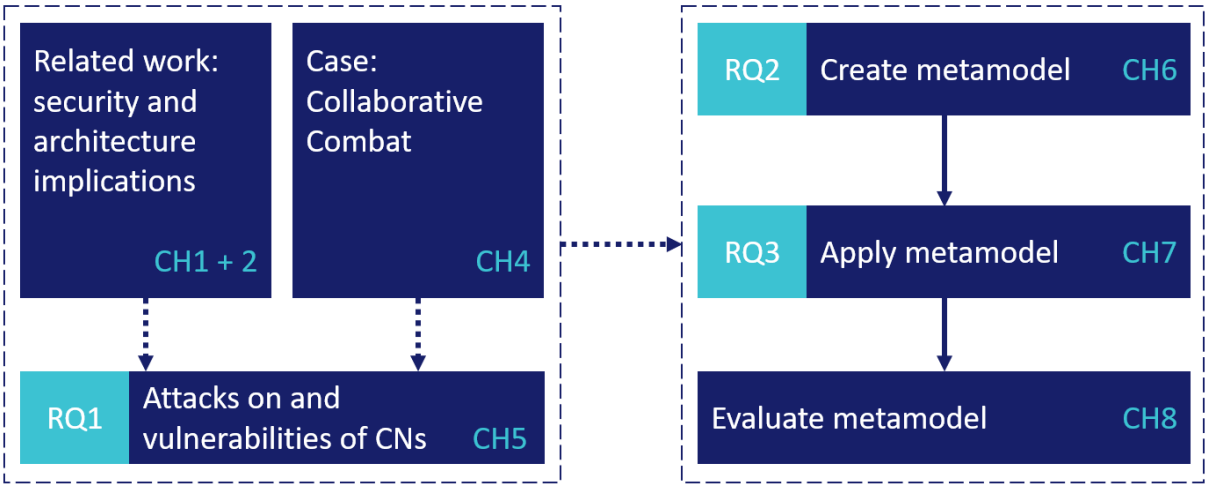


Figure 1: Research Methodology

3.2 Description of the Design Science Activities

3.2.1 Activity 1: Problem Identification and Motivation

This activity includes the definition of a specific research problem as well as the justification of the solution in terms of value.
In Chapter 1, the following research problem is introduced along with its context: *integrating security properties within CN system architectures*. To solve this problem, one knowledge question on vulnerabilities (i.e. RQ1) and two design questions on the creation and usage of a new CN metamodel (i.e. RQ2 and RQ3) are defined.
In the same chapter, the intended value of the metamodel can be summarised as: *to assert the security of CN ecosystems and promoting security by design*. Asserting security in this case refers to the capability of the metamodel to model systems such that their vulnerabilities can be uncovered and attacks assessed. Activity 5 evaluates to what extent the created design artefacts (i.e. the CN metamodel with its modelling pattern and viewpoints) solves the research problem.

3.2.2 Activity 2: Define the Objectives for a Solution

This activity includes the inference of objectives of a solution from the problem definition and knowledge of what is possible and feasible.
The manifestation of this activity is depicted as three knowledge components on the left part of Figure 1. These components are fundamental to all three main design activities that follow, as design objectives can be inferred from. Presented in Chapter 1, the first component embodies the security and architecture implications as extracted from related work (see Chapter 2). Presented in Chapter 4, the second component embodies the case on CC, a typical CN that is adopted as the motivating example throughout this study. The third component embodies the attacks and vulnerabilities that are relevant to CNs. This includes inducing vulnerable system parts from the case and deducing their vulnerabilities based on CN attacks that can be found in related work. This act of induction and deduction constitutes knowledge question RQ1 and is described in Chapter 5.

3.2.3 Activity 3: Design and Development

This activity includes the determination of artefact properties and the creation of the actual artefact.
As the entry point of this design science study, this is the first main design activity of this study in which the CN metamodel is designed and developed. The constructs and relationships that are necessary for modelling CN system architectures are derived from an existing system architecture language (see Section 2.3) that is adapted based on the architectural implications as extracted from the related work. The modelling elements that are necessary for modelling security properties on top of CN system architectures are derived from the vulnerabilities that are relevant to CNs while considering the security implications as extracted from the related work. In addition, a modelling pattern is suggested that illustrates the intended use of the generic metamodel and viewpoints

are presented for modelling specific security aspects. Described in Chapter 6, iterations of this activity constitute design question RQ2.

3.2.4 Activity 4: Demonstration

This activity includes a demonstration of the usage of the artefact to solve one or more instances of the problem.
This is the second main design activity of this study in which the CN metamodel is applied to assert the security of CNs by design. More specifically, this study illustrates how the metamodel can be applied to create security architectures within the context of the Arcadia architecture development methodology and how security analysis techniques can be applied to assess the level of security in these architectures. The vulnerabilities of CNs, applied on the case on CC, establish a generalisable problem instance. This activity constitutes design question RQ3 and is described in Chapter 7.

3.2.5 Activity 5: Evaluation

This activity includes the observation and measurement of how well the artefact supports a solution to the problem.
This is the third and last main design activity of this study in which the utility value of the CN metamodel, modelling pattern and viewpoints is determined. The evaluation is based on a comparison between the application of the metamodel (i.e. activity 4), and the intended values and objectives thereof (i.e. activity 1 and 2). In other words, this activity assesses to what extent the CN metamodel, modelling pattern and viewpoints can be used to model relevant system and security properties of CNs such that their vulnerabilities can be uncovered and attacks assessed.

3.2.6 Activity 6: Communication

This activity includes the communication of the problem and its importance, the artefact, its utility and novelty, the rigour of its design, and its effectiveness to researchers and other relevant audiences.
This written report is the main communication format of this study and is structured based on the activities as presented above. In addition, presentations are given to relevant stakeholders within the University of Twente and Thales Nederland in which the study results are shared and discussed.

4 CASE ON COLLABORATIVE COMBAT

Adopted as the motivating example in this study, the fictional case below illustrates a typical instance of Collaborative Combat in which multiple military units form a coalition force to achieve some common goal.

COALITION X: Information and Battlefield Superiority through Collaboration
Coalition X is a military collaboration between the coast country Atlantis and their neighbouring country Zembla. Due to recent trespasses by unknown entities in the territorial waters of Atlantis, their state-of-the-art navy requested the national air force to assist in the sky. However, due to structural cutbacks in the Atlantisian air force, they are running on outdated legacy systems, limiting them to only assist in monitoring the coastline with short-range radar systems. Hence, the more modern Zemblian air force is asked to assist in monitoring the rest of the territorial sea and to assist in combat when necessary. As the Zemblian air force has the same coastline monitoring capabilities as the Atlantisian air force, the latter unit should be able to join and leave the coalition at any time.

The main information systems of the coalition and their relationships are depicted in Figure 2. In this coalition, the navy of Atlantis is the leading unit and is supported by the air forces of Atlantis and Zembla. All units have their own commanders, controlling their own soldiers and systems - both instances of intelligence-gathering sensors and/or effectors. Although the physical and information systems of the Atlantisian navy and air force are controlled exclusively by their own commanders, they are deployed with similar security requirements and are subject to the same security policies. Hence, all military systems within Atlantis are part of the same security domain. As the systems of the Zemblian air force are governed and controlled by a third party, they are part of another security domain. In the figure, the outer solid rectangle represents the CN ecosystem while the inner solid rectangles represent the security domains. Within the security domains, the dashed rectangles represent the military units.
With regards to the physical systems, both the navy of Atlantis and the air force of Zembla have diverse stationary/mobile sensors and a range of effectors in place. In addition, the air force of Atlantis deploys additional redundant short-range radar systems that can be considered sensors. Next to operating on land, the Atlantisian navy assigned a single vessel for this operation while the Zemblian air force has two aircrafts ready - each having its own arsenal of sensors and effectors on board. For communication and control purposes, each physical system has an information system integrated that provides interfacing capabilities. In the figure, the dark blue rectangles represent the physical systems, including their atomic Integrated Information Systems (IISs).

With regards to the Collaborative Information Systems (CISs), three main systems are deployed for this mission along with a number of Command and Control (C&C) systems. Firstly, all effectors and sensors that are not deployed on a vessel or an aircraft are controlled by one or more C&C centres. For this study, a simplified scenario is assumed in which each C&C centre has a single (legacy) information system in place that can control effectors, sensors or both but always within the domain of a particular military unit. Secondly, the NavyOS Combat Management System (CMS) is deployed by the navy of Atlantis. This system controls their vessel and onboard effectors, and combines all onboard sensory data with historical data to enhance the view on the sea and to perform automated situational assessments. Thirdly, two instances of a similar CMS - SkyControl - are deployed by the air force of Zembla, each controlling the sensors and effectors on one of their aircrafts. For internal communication purposes, both instances are connected to each other, sharing sensory data to improve the detection and identification of flying objects. Lastly, as the leading unit, the Atlantisian navy also deploys the InterAct radio system which enables - hierarchical and transverse - communication and collaboration capabilities among all units of the coalition. This system interfaces to all CMS and C&C systems, facilitating the communication and information sharing among all units. Furthermore, due to tight integrations with the modern CMSs, InterAct can directly or indirectly control the physical systems connected to NavyOS and SkyControl, facilitating true collaborations in tactical situations. In the figure, the light blue rectangles represent the information systems while the arrows represent the relationships between all Integrated and Collaborative Information Systems.

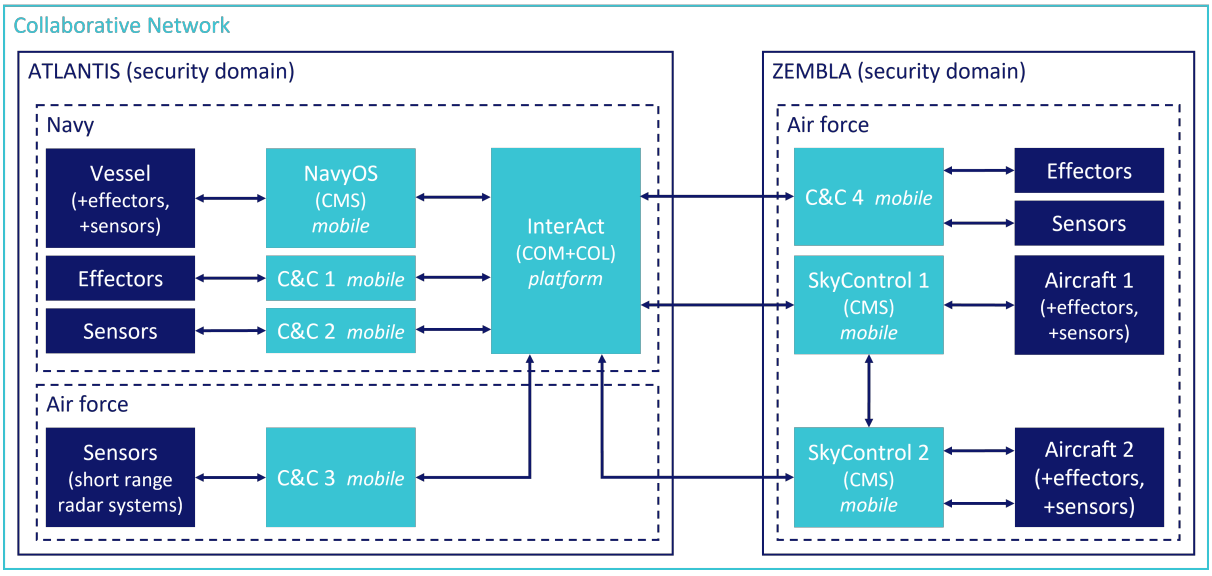


Figure 2: CC Case - Main Information Systems and Relationships

(Note that ”mobile” and ”platform” in this figure does not refer to the military connotation regarding the physical mobility of entities. Instead, these terms should be interpreted in line with G1 of Chapter 1 regarding the collaborative characteristics of entities.)

Powered by the CISs, this system of systems can be considered a Collaborative Network as defined in Chapter 1. The three military units are collaborating (P2) to monitor and protect the territorial waters of Atlantis while the involved systems are crossing two security domains (G2). With regards to monitoring the territorial waters, the Atlantisian navy and the Zemblian air force are both able to monitor the whole tactical area while the Atlantisian air force is only able to monitor the coastline, redundantly (P3). With regards to protecting the territorial waters, the Atlantisian navy is able to combat on the sea while the Zemblian air force is able to combat in the air, complementary (P3).
As presented in Figure 3, all information systems are controlled by a commander from the respective unit. This means that all soldiers, effectors and vessels/aircraft are controlled within a unit (P1), independently of other units and with limited environmental observability (P5). In the figure, the dark blue people represent commanders while the light blue people represent soldiers. The arrows represent the information exchanges. The information systems NavyOS, SkyControl and InterAct are typical examples of CC CISs because of their unique properties. For all these systems hold that they are facilitating the real-time horizontal exchange and interpretation of large quantities of mission-critical information - and perhaps even providing response choices by AI. Furthermore, these systems are highly interoperable with existing and legacy systems, hiding complexity while allowing the coalition to face all kinds of threats without the need of having a specific weapons system for each threat. In this case, InterAct facilitates the sharing of sensory data amongst all units within the coalition, allowing intelligent systems such as SkyControl and NavyOS to improve their information accuracy whenever this is necessary (P4). By combining these capabilities with a shared hierarchical control over a part of the effectors, increasingly accurate information and intelligence can be used for real-time battlefield response. This makes InterAct the platform of the ecosystem which dynamically host all other - mobile - systems (G1), providing commanders with information while significantly raising the tempo of operations. Coalition X is an example of effective command and control in time-sensitive operations crossing different units: information and battlefield superiority through collaboration.

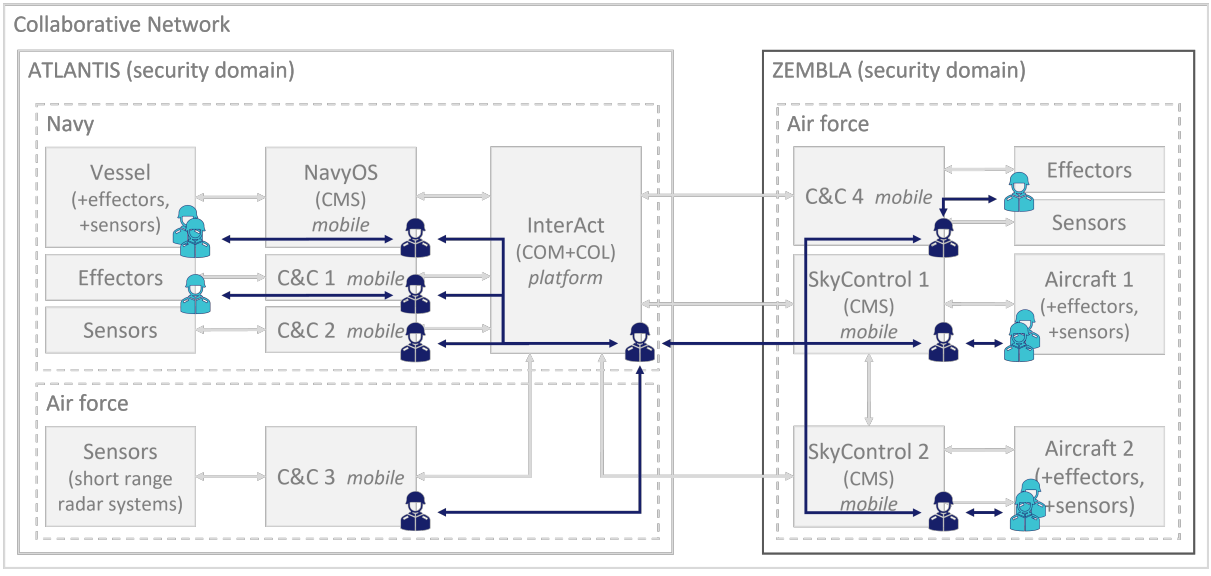


Figure 3: CC Case - Human Entities and Information Exchanges

5 VULNERABILITIES OF CN ECOSYSTEMS

This chapter describes vulnerabilities that are commonly found in CN ecosystems. This includes inducing vulnerable system parts from the case on CC (see Chapter 4) and deducing their vulnerabilities based on CN attacks that can be found in related work (see Chapter 2). This act of induction and deduction answers research question 1:

RQ1 What are relevant vulnerabilities of Collaborative Networks?

RQ1.1 What are vulnerable parts of CN ecosystems?

RQ1.2 What are attacks on and vulnerabilities of these system parts?

5.1 Vulnerable Parts of CN Ecosystems

5.1.1 Entities within CN Ecosystems

Based on the case on CC, entities within a CN ecosystem can be distinguished into:

- **IIS**: Integrated Information Systems (IISs) that are embedded on or are closely related to a physical system (e.g. the software running on a radar system).
- **CIS**: platform or mobile Collaborative Information Systems (CISs) that have distributed, heterogeneous, autonomous and asynchronous characteristics (e.g. the InterAct radio system).
- **Actor**: Humans that interact with either a IIS or a CIS, e.g. commanders.

This study assumes that the security architectures of the atomic information systems are known and that all collaborating atomic and human entities can be considered secure. Hence, the remainder of this study focuses on the vulnerabilities related to CISs.

5.1.2 Transport and Trust Relationships within CN Ecosystems

As illustrated in the Ajanta Mobile Agent System [17], not only entities are susceptible to attacks but the relationships between those entities are vulnerable as well. Based on the Interaction and Components model of [14], three types of relationships can be distinguished that have different kinds of vulnerabilities. Sorted from loosely to tightly coupled, the transport relationship types are:

- **Communication relations:** transporting messages (without directly interfering local resources).
- **Calling relations:** transporting tasks without directly interfering local resources.

- **Execution relations:** transporting tasks that directly manipulate local resources.

A further distinction can be made based on whether a message or task originates from an entity within the security domain or from another security domain. Figure 4 presents the entity and relationship types within the case on CC. From this figure, two types of relationship pairs can be distinguished, IIS–CIS and CIS–CIS.
As IISs and CISs are tightly coupled and operate within the same security domain, the relationship pair IIS–CIS is likely either a calling or an execution relation, depending on the function. In this case, controlling effectors require execution relationships while sensors only need to be called. Evidently, a communication relation is also possible even though this is not illustrated in this particular case.
In contrast, two CISs can be tightly or loosely coupled and might cross security domains. For tightly coupled CISs within a security domain, e.g. NavyOS with InterAct, all relationship types are likely to occur. In this case, calling and execution relationships are used for calling onboard sensory information and controlling onboard effectors, respectively. For tightly coupled CISs crossing security domains, e.g. SkyControl 1 and InterAct, the relationship type should be limited to either calling or communication in order to maintain control within the security domains. In this case, this means that InterAct can call sensory data from the Zemblian aircrafts but is limited to indirectly controlling their onboard effectors. For loosely coupled CISs, e.g. C&C 4 and InterAct, the relationship type is naturally limited to communication. This means that InterAct can only ask a C&C system to pass sensory data or enact effectors.

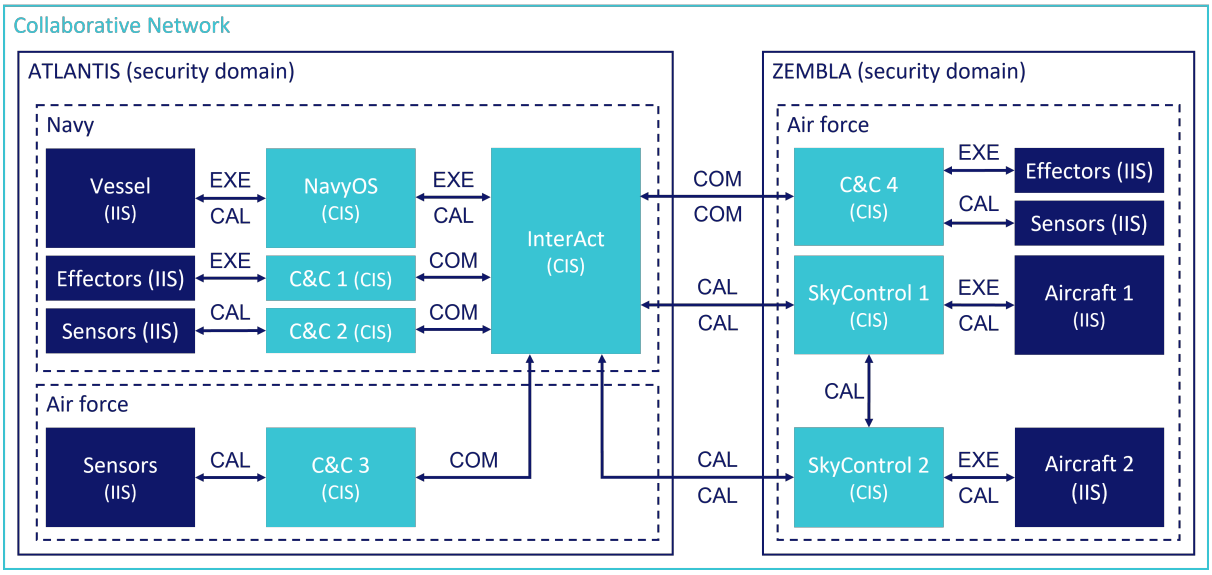


Figure 4: CC Case - Entity Types and Transport Relationships
(COM = Communication relation, CAL = Calling relation, EXE = Execution relation)

Table 3 presents a matrix with all possible relationship pairs and transport relationships. Of the three relationship pairs, only IIS–CIS exist in both collaborative and non-collaborative ecosystems. The relationship pairs CIS–CIS (tight) and CIS–CIS (loose) are introduced because of the collaborative nature of a CN ecosystem, indicated by an asterisk (*). In contrast to the entities, all relationship pairs and their respective relationship types are considered in this study as threats might cascade from entity to entity through their relationships; similar to cascading attacks in a SoS [24].

		TRANSPORT RELATIONSHIPS	
		Intra-domain	**Inter-domain**
RELATIONSHIP PAIR	**CIS–CIS**	Execution Calling Communication	N/A
	CIS–CIS (tight)*		Calling Communication
	CIS–CIS (loose)*	Communication	

Table 3: CN Ecosystems - Relationship Pairs and Transport Relations

In addition, relationships can be classified based on their trust properties. Derived from [10, 23], three types of relations can be distinguished based on which the identity and intentions of mobile and platform entities can be evaluated:

- **Authentication trust:** the belief in the authenticity of the keys held by entities.
- **Execution trust:** the belief that a platform or mobile entity will faithfully and without any tampering execute the given tasks or forward the given message.
- **Competency trust:** the belief that a platform or mobile entity is competent in executing the given tasks.

Figure 5 presents the trust relationships that can be found in the case on CC. Since the relationship pair IIS–CIS is static and fully known, there is no need to embrace trust relationships in such cases. For the dynamic relationship pair CIS–CIS, the required type of trust relationship depends on the type of transport relation and whether the involved entities cross security domains. If all involved entities reside within a security domain, it can be assumed that the entities are trustworthy and competent; i.e. assuming execution and competency trust. However, this is only true if and only if all involved entities are authentic. Hence, for intra-domain CIS–CIS relationships, only authentication trust is required. For inter-domain CIS–CIS relationships, authentication trust is required for the same reason. In addition, execution trust is also required since there is limited to no observability or knowledge on whether entities in another security domain will faithfully and without tempering execute given tasks or forward given messages. Specifically for inter-domain calling relationships, competency trust is also required as it cannot be assumed that ”unknown” entities are competent in executing given tasks. These occurrences of trust relationships are presented in Table 4.

		REQUIRED TRUST RELATIONSHIPS	
		Intra-domain	**Inter-domain**
TRANSPORT RELATIONSHIP (of CIS–CIS)	**Communication**	Authentication	Authentication Execution
	Calling		Authentication Execution Competency
	Execution		N/A

Table 4: CN Ecosystems - Trust Relations of CIS-CIS Transport Relations

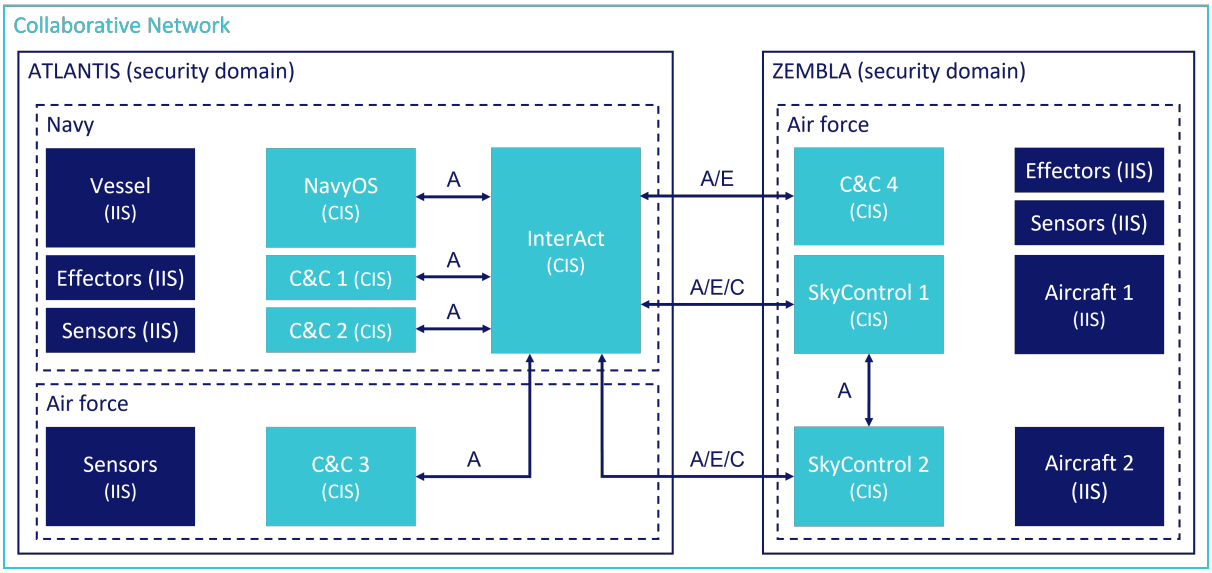


Figure 5: CC Case - Required Trust Relationships
(A = Authentication trust, E = Execution trust and C = Competency trust)

5.1.3 Summary of the Vulnerable Parts of CN Ecosystems

Based on the previous two subsections, it can be concluded that the following parts of a CN ecosystem are vulnerable to attacks: CIS platform entities; CIS mobile entities; IIS–CIS intra-domain communication, calling and execution relations; CIS–CIS intra-domain communication, calling and execution relations with authentication trust; CIS–CIS inter-domain communication relations with authentication and execution trust; and CIS–CIS inter-domain calling relations with authentication, execution and competency trust. This is presented in Table 5.
Note that no distinction has been made between tightly or loosely coupled CIS–CIS relationship pairs. This is because these relationships pairs share similar vulnerabilities depending on the type of transport and required trust relationship(s), rather than on the tightness of the coupling.

VULNERABLE ENTITIES			
CIS platform entities			
CIS mobile entities			
VULNERABLE RELATIONSHIPS			
Rel. pair	**Domain**	**Transport relation(s)**	**Required trust relation(s)**
IIS–CIS	intra	communication, calling or execution	N/A
CIS–CIS	intra	communication, calling or execution	authentication
CIS–CIS	inter	communication	authentication & execution
CIS–CIS	inter	calling	authentication, execution & competency

Table 5: CN Ecosystems - Vulnerable System Parts

5.2 Attacks on and Vulnerabilities of CN Ecosystems

Many different threat models exist to categorise attacks on systems that can be considered a CN. This includes generic threat models for information system security as well as threat models that are specifically created for e.g. mobile agent systems and open grid systems. From these attacks, vulnerabilities can be deduced for the vulnerable system parts as induced in the previous section.

5.2.1 Attacks on CN Ecosystems

Attacks can be classified based on the capabilities they use to exploit a vulnerability and their origin. With regards to the capabilities, there are attacks that use the autonomy of existing system parts, exploiting trust and vulnerabilities of authenticated entities. Other attacks use capabilities beyond (or of) existing system parts, exploiting vulnerabilities of authenticated entities and existing relationships. Analogous to [37], the first type of attacks is referred to as inside attacks (breaching trust) while the second type is referred to as outside attacks (not necessarily breaching trust). With regards to the origin of attacks, inside attacks originate from either mobile or platform entities that are part of the ecosystem, similar to [8, 22, 17], while outside attacks can also originate from other entities (but do not necessarily have to).
For all attacks hold that one or more of the STRIDE [41] elements are enacted; in fact, breaching one or more of the CIA & AAA security risk concerns. Attacks take place in either a critical state in which the attacked entities interact [39] (i.e. communicate, call or execute), a critical state in which an attacked mobile entity is moving within the ecosystem (i.e. migrating between platform entities or in/out of the ecosystem) [22], or a non-critical state [39].
Table 6 presents attacks that are likely to occur in CN ecosystem. These attacks are found in literature and are adapted based on the security and architecture implications W1-W5 and B1-B4 as presented in Chapter 1. (Note that denial-of-execution refers to an entity not wanting to execute a given task while a denial-of-service attack refers to an entity making it impossible for another entity to execute a given task.)
All attacks that are unique to or different for CNs, when compared to atomic systems, are marked with an asterisk (*). For inside attacks (i.e. A1-A7) hold that they are all unique to or different for CNs. This is caused by the fact that the distributed (P1) and heterogeneous (P3) mobile or platform entities (G1) have to collaborate (P2) crossing security domains (G2), causing a strong need for trust when interacting with each other. This cannot be guaranteed as direct trust is assumed while entities act autonomously and asynchronously (P5) in an environment that is adaptive and scalable (P4). For the outside attacks hold that all three marked attacks (i.e. A8-A10) can only enact in a critical system state of entities moving through, in or out of the ecosystem - again related to the distributed (P1), adaptive and scalable (P4) character of CNs. All other outside attacks can also be enacted similarly in atomic ecosystems and are, therefore, not unique to or significantly different for CNs.
Furthermore, it can be argued that collusion or discrediting attacks through the submission of false positive or false negative trust ratings (W5) [37] is also a relevant inside attack. However, as direct trust is assumed between all collaborating entities, there is

no need for recommendations from other entities or a trusted authority. Hence, these trust-related attacks are not considered in this study.

		How (STRIDE)	When (critical state)
INSIDE ATTACKS			
Mobile entities against other entities			
A1*	Circumvention of inter-domain CIS–CIS calling relationships (W1/B1) [14] by mobile entities, accessing unauthorised functions of other entities [17] - breaching execution trust (W5).	E	Interacting
A2*	Denial-of-execution by mobile entities from other domains [14, 37] - breaching execution or competency trust (W5).	D	Interacting
A3*	Denial-of-service attacks by mobile entities from other domains [17, 22, 37, 39] using CIS–CIS communication or calling relations [8, 14] - breaching execution trust (W5).	D	Interacting
A4*	Repudiation of CIS–CIS messages/tasks (W4/B4) sent by mobile entities from other domains [8, 14, 22] - breaching execution trust (W5).	R	Interacting
Platform entities against other entities			
A5*	Misusing resources [14, 17] and information [8, 17] (W1/B1) of other entities by platform entities from other domains - breaching execution trust (W5).	T/I	Interacting
A6*	Denial-of-execution by platform entities from other domains [8, 14, 37] - breaching execution or competency trust (W5).	D	Interacting
A7*	Repudiation of CIS–CIS messages/tasks (W4/B4) executed or forwarded by platform entities from other domains [14, 22] - breaching execution trust (W5).	R	Interacting
OUTSIDE ATTACKS			
A8*	Masquerade platform/mobile entities (W1/B1/B3) [8, 14, 22, 37, 39] by exploiting authentication (inter/intra) CIS–CIS trust relations (W5) [37].	S	Moving entities
A9*	Unauthorised replication [14] or modification [22, 17] of platform/mobile entities (W1/B1/B3).	T	Moving entities
A10*	Theft of rights or delegation misuse (W3/B2) [14, 22] of mobile entities (W1/B1/B3).	E	Interacting Moving entities
A11	Eavesdropping [8, 14, 22, 17, 39] of CIS–CIS or IIS–CIS communication (inter/intra), calling (inter/intra) and execution (intra) relations (W1/B1/B3).	I	Interacting
A12	Unauthorised intervention of CIS–CIS or IIS–CIS communication (inter/intra), calling (inter/intra) and execution (intra) relations [8, 14, 17, 22, 39] (W1/B1/B3).	T	Interacting
A13	Tampering of physical infrastructure (e.g. fire, short-circuits, natural hazards and power interruptions) [39].	T	Non-critical

Table 6: CN Ecosystems - Attacks

5.2.2 Vulnerabilities of CN Ecosystems

The attacks as presented in Table 6 exploit certain vulnerabilities that can be found in vulnerable parts of CN ecosystems (see Table 5). Using the Mitre ATT&CK cybersecurity framework [51] as a reference, Table 7 presents a list of vulnerabilities that can be deduced from their related attacks. Note that this is not a comprehensive list of vulnerabilities. Rather, this list is aimed to illustrate the range of possible vulnerabilities that the identified attacks can exploit.
All vulnerabilities that are unique to or different for CNs are marked with an asterisk (*). This is the case for the first eight vulnerabilities (i.e. V1-V8). Similarly to the attacks, this is because mobile or platform entities (G1) have to collaborate (P2) on inter-domain level (G2) while acting autonomously and asynchronously (P5). This introduces a need to rely on trust relationships, and causes a lack of (security) information while having limited to no mutual or environmental observability.

VULNERABILITY		**ATTACK**
CIS platform and mobile entities through any inter-domain CIS–CIS COM/CAL relation		
V1*	Inadequate identification, authentication, authorisation and accountability of entities from other domains.	A1 / A4 / A7
V2*	Inadequate traffic filtering of attacked cross-domain relationships or attacking entities from other domains.	A3
CIS platform and mobile entities through any intra/inter-domain CIS–CIS trust relation		
V3*	Weak or inadequate authentication.	A8
V4*	Access rights and delegations with insufficient entity access or action restrictions, or with insufficient temporal constraints.	A10
CIS platform or mobile entities		
V5*	Conflicting interests or goals between entities (from other domains).	A1 - A12
V6*	Limited status/capacity information about entities from other domains.	A2 / A3 / A6
V7*	Limited interoperability with entities from other domains.	A2 / A6
V8*	Inconsistent capability definitions with entities from other domains.	A2 / A6
V9	Inadequate encryption of identifiers or security properties of entities.	A8
V10	Inadequate integrity and authenticity protection.	A9
V11	Insufficient data, time or distinct functional redundancy.	A13
V12	Insufficient spatial or geographic redundancy.	A13
Intra-domain IIS-CIS or CIS-CIS COM/CAL/EXE relations **Inter-domain CIS-CIS COM/CAL relations**		
V13	Relationships susceptible for delays.	A3 / A12
V14	Relationships susceptible for redirections.	A12
V15	Inadequate encryption, monitoring or segmentation of relationships.	A11 / A12
V16	Relationships susceptible for replays.	A12

Table 7: CN Ecosystems - Vulnerabilities

6 CREATION OF CN METAMODEL

This chapter presents a metamodel, a suggested modelling pattern and diverse viewpoints that can be used to create security architecture descriptions of CNs. For this, constructs and relationships have to be identified, defined and justified for both the system properties as well as the security properties of CNs.
The constructs and relationships that are necessary for modelling CN system architectures are derived from an existing system architecture language (see Section 2.3) which is adapted based on the architectural implications as presented in Chapter 1. The constructs and relationships that are necessary for modelling security properties on top of CN system architectures are derived from the vulnerabilities that are relevant to CNs (see Chapter 5) while considering the security implications as presented in Chapter 1. By iterating through these activities, research question 2 is answered:

RQ2 How does a metamodel for CN security architectures look like?

RQ2.1 What are relevant constructs and relationships for modelling CN system architectures?

RQ2.2 What are relevant constructs and relationships for modelling security properties within CN system architectures, realising CN security architectures?

RQ2.3 What are relevant modelling patterns and viewpoints of the metamodel for CN security architectures?

6.1 Modelling CN System Architectures

Even though the metamodel for CNs is intended to be independent of any modelling tool or language, the existing modelling elements of Eclipse Capella are adopted for its semantic and syntactic context. This section extends the constructs and relationships of Eclipse Capella, supporting the modelling of CN system architectures.
Eclipse Capella is an open-source solution that is adopted in various industrial domains, including by Thales [29], for modelling i.a. system architectures. The solution is created with a model-based systems engineering approach [28] in mind, meaning that a formal modelling language is used for specifying, designing, analysing and verifying systems. This allows the creation of integrated functional views while also having full traceability from user needs to physical components, reducing inconsistencies in system descriptions and system engineering processes. The modelling language used in Eclipse Capella is inspired by SysML but is positioned as a less expressive but yet a more specialised language [59]. To a large extent, the diagrams are based on existing diagrams from the UML and SysML standards [32].

6.1.1 Existing Constructs and Relationships of Eclipse Capella

For modelling system architectures, Eclipse Capella defined the Physical Architecture Diagram. This diagram presents the allocation of behavioural components into implementation components. The most important constructs and relationships are depicted in Figure 6.
In the figure, four main constructs can be found. At the top, a Physical Actor is depicted, representing an entity that interacts directly with the system under study. At the bottom left, a behavioural physical component (Behaviour Physical Component (PC)) is depicted, representing an entity that can deploy other Behaviour PCs and realises a part of the system functionality. At the bottom right, a node physical component (Node PC) is depicted, representing an entity that can deploy other Node or Behaviour PCs and delivers resources for the execution of Deployed Behaviour PCs. Within a (Deployed) Behaviour PCs and a Physical Actor, one or more Physical Functions reside, representing the operations or services the Actors or PCs fulfil. (Note that *PC* refers to *Physical Component* rather than *Personal Computer*.)
With regards to the relationships, three types can be distinguished. Functional Exchanges are unidirectional relationships, facilitating the data exchange between Physical Functions. Physical Links are bidirectional relationships, representing the communication means between (Deployed) Node PCs or between a (Deployed) Node PC and a Physical Actor. Component Exchanges are uni or bidirectional (delegated) relationships, carrying the Functional Exchanges between (Deployed) Behaviour PCs.

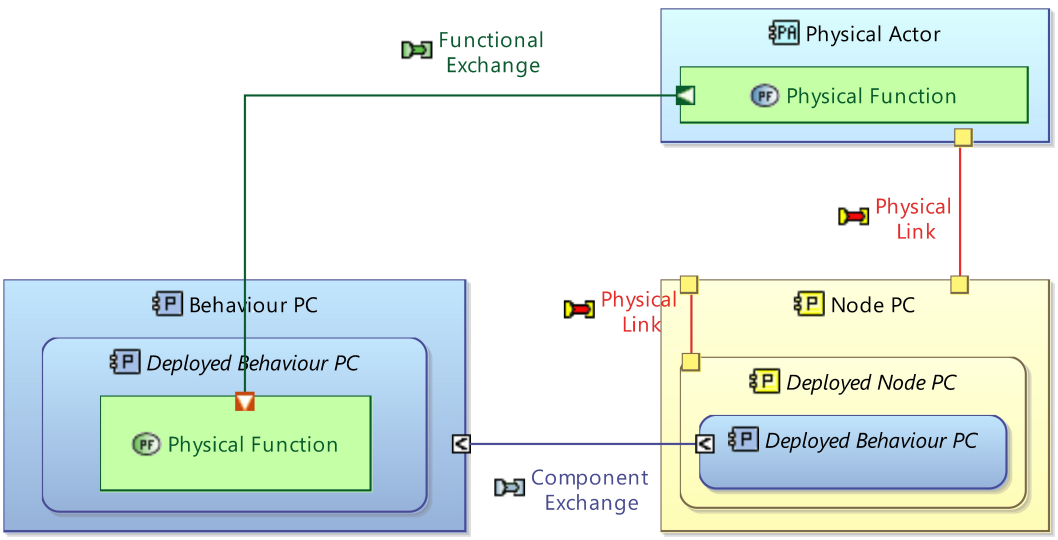


Figure 6: Capella - Physical System Architecture Diagram

6.1.2 Additional Modelling Elements to Create CN System Architectures

To identify which additional constructs and relationships are necessary for modelling CN system architectures, Table 8 is created. This table maps all architectural implications as presented in Chapter 1 to either existing modelling elements of Eclipse Capella or suggested new modelling elements.

Firstly, mobile and platform entities (G1) can be modelled using (Deployed) Behaviour PCs and (Deployed) Node PCs while their relationships can be modelled using Component Exchanges and Physical Links. For the creation and the security of the relationships, a secure association service can be adopted in line with the security services as presented in existing SOA-metamodels (see Table 1 of the related work). This service can be modelled using Physical Functions. Similarly, authorisation (W1), accountability (W4) and authentication (B3) services can also be modelled using Physical Functions
Secondly, information exchanges (B1) can be represented by Functional Exchanges. Again, Physical Functions can be adopted to cover the cryptographic needs of these data exchanges (B1) and for the maintenance of changing security policies (W2/B1). Similarly, privilege delegations (W3/B2) can be modelled using Physical Functions that represent intra-domain delegation services while Component Exchanges with Delegations can be used to represent the intra/inter-domain delegation relationships. The temporal characteristics of a delegation can be included in the description of the exchange.
Thirdly, no constructs are required for modelling the trust facilities (W5) or 3rd-party accountability (B4). In respective order, this his is caused a lack of need for trust authorities as direct trust is assumed and a lack of need for 3rd-party accountability as trust-based accountability is assumed.
Finally, to support the concept of security domains (G2), there is a need to introduce a new construct that groups Physical Actors, (Deployed) Node PCs and (Deployed) Behaviour PCs into Security Domains. The same concept can also be used to model sandboxing environments. For the related interdomain service, including support for credential conversion and identity mapping, Physical Functions can be used.
Considering the above, Figure 7 illustrates the modelling elements of a Capella Physical Architecture Diagram where the suggested additional modelling elements are included. In this case, the Security Domain construct is modelled as an orange dotted rectangular.

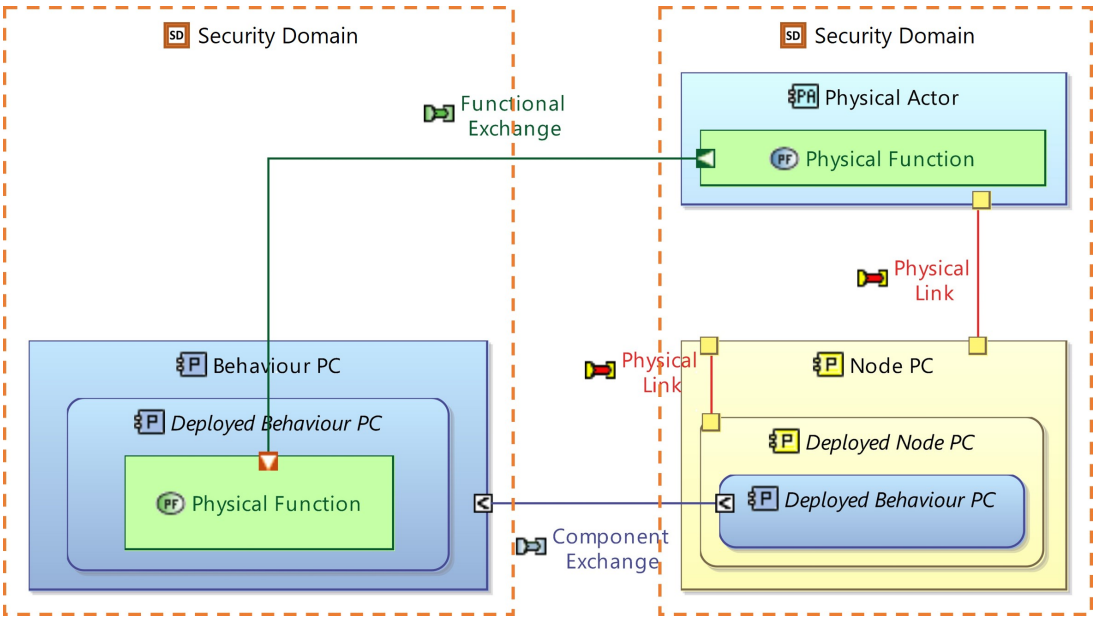


Figure 7: Capella - Extended Physical Architecture Diagram

IMPLICATION		TYPE	MODELLING ELEMENT	COMMENT
G1	Mobile and platform entities	Existing	(Deployed) Behaviour PC (Deployed) Node PC	
		Existing	Component Exchange Physical Link	
		Existing	Physical Functional	Secure Association function
G2	Security domains	New	Security Domain/ Sandbox	New construct to group Physical Actors, Node PCs and Behaviour PCs into Security Domains or Sandboxes
		Existing	Physical Function	Interdomain function (including credential conversion and identity mapping)
W1	Integrated centralised authorisation	Existing	Physical Function	Authorisation and Access Control function
W2	Adaptive policies	Existing	Physical Function	Security Policy function
W3	Privilege delegation	Existing	Physical Function	Privilege Management function
		Existing	Component Exchange with Delegations	Temporal characteristics can be included in the description
W4	Integrated centralised accountability	Existing	Physical Function	Accountability function
W5	Trust facilities	N/A	N/A	No trust authority as direct trust is assumed
B1	Information exchange	Existing	Component Exchange Functional Exchange	
		Existing	Physical Function	Security Policy function Cryptographic Support function
B2	Privilege delegation	Existing	Component Exchange with Delegations	Temporal characteristics can be included in the description
B3	Federated decentralised authentication	Existing	Physical Function	Authentication and Security Attribution function
B4	3rd-party or trust-based accountability	N/A	N/A	No 3rd-party as trust-based accountability is assumed

Table 8: CN Ecosystems - Modelling Elements of System Architectures

6.2 Modelling Security Properties within CN System Architectures

With the system architectures in place, the security properties have to be added, allowing security to be asserted and promoting security by design. As an add-on to Eclipse Capella, the Capella Cybersecurity viewpoint [27] exist that already provides limited means for modelling security concerns, mechanisms, policies and threats on top of system architecture descriptions. This section extends the security constructs and relationships of the Capella Cybersecurity viewpoint, supporting the modelling of CN security architectures.

6.2.1 Existing Constructs and Relationships of the Capella Cybersecurity Viewpoint

For modelling the security properties of systems, the Capella Cybersecurity viewpoint defined a Threat Diagram and extended the Physical Architecture Diagram with security properties.

The Threat Diagram is used to model system threats, threatened assets and involved actors. The most important constructs and relationships are depicted in Figure 8.
Central in this diagram is the concept of a Threat, representing a situation that is unwanted by stakeholders and that has to be avoided. Each threat has a certain Threat Kind to categorise threats and a priority level from 0 to 5 (in the right upper corner).
Involved in Threats are Physical Actors that are distinguished into *threat source*, *not trusted* or *trusted* - as indicated by an icon. The involvement relationship is depicted with a unidirectional arrow labelled «i».
Threats are affecting Primary Assets that can be categorised as Functional or Informational, indicated by a colour and an icon. Functional Primary Assets are related to Physical Functions and represent activities, processes or functionality that are valuable for stakeholders and that need to be protected. Information Primary Assets are related to Exchange Items and represent information that is valuable for stakeholders and that needs to be protected. The affection relationship is depicted with a unidirectional dotted arrow labelled «a».

For the Physical Architecture Diagram holds that the Capella Cybersecurity viewpoint introduced icons and layers to depict the security properties within existing Physical Architecture Diagrams; i.e. creating security architectures from system architectures. The most important additions are presented in Figure 9.
In both Figure 9a and Figure 9b, different icons are introduced that are providing additional security information on top of existing constructs. Firstly, as in the Threat Diagram, Physical Actors are distinguished into *threat source*, *not trusted* or *trusted*. Similarly, (Deployed) Behaviour PCs and Node PCs are categorised as either *trusted* or *untrusted*. Finally, it is possible to indicate whether a Physical Function has a data storage functionality, a data remanence functionality or both. (Data remanence refers to a function that manipulates data in such a way that data remains even after attempts have been made to remove or erase it.)
In Figure 9a, the primary asset layer is enabled along with the trust boundaries layer. The first layer highlights the threatened Physical Functions and Component/Functional

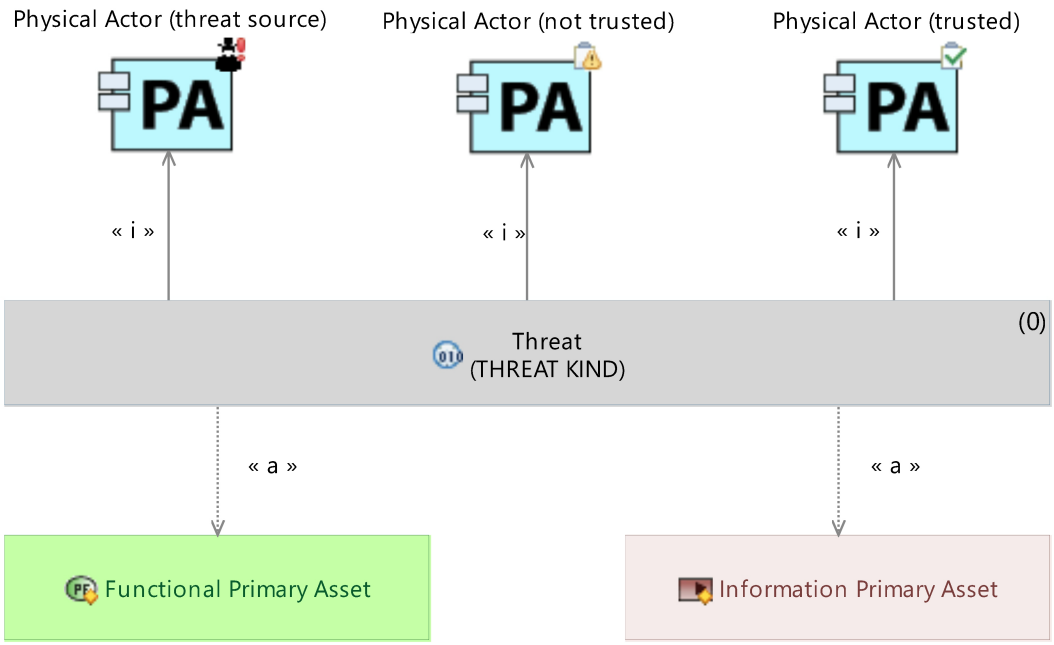
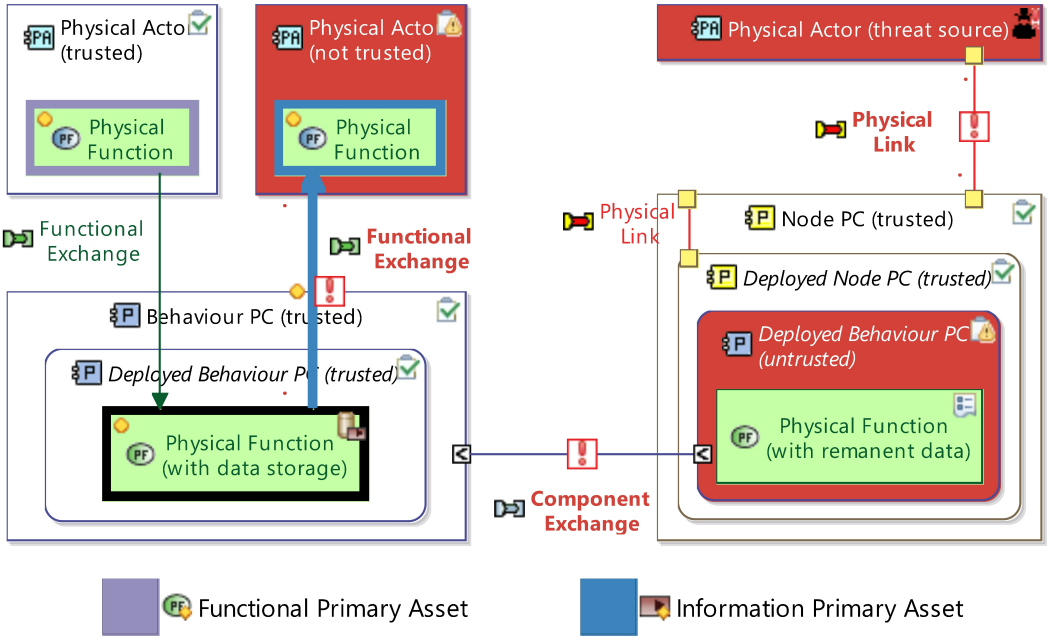


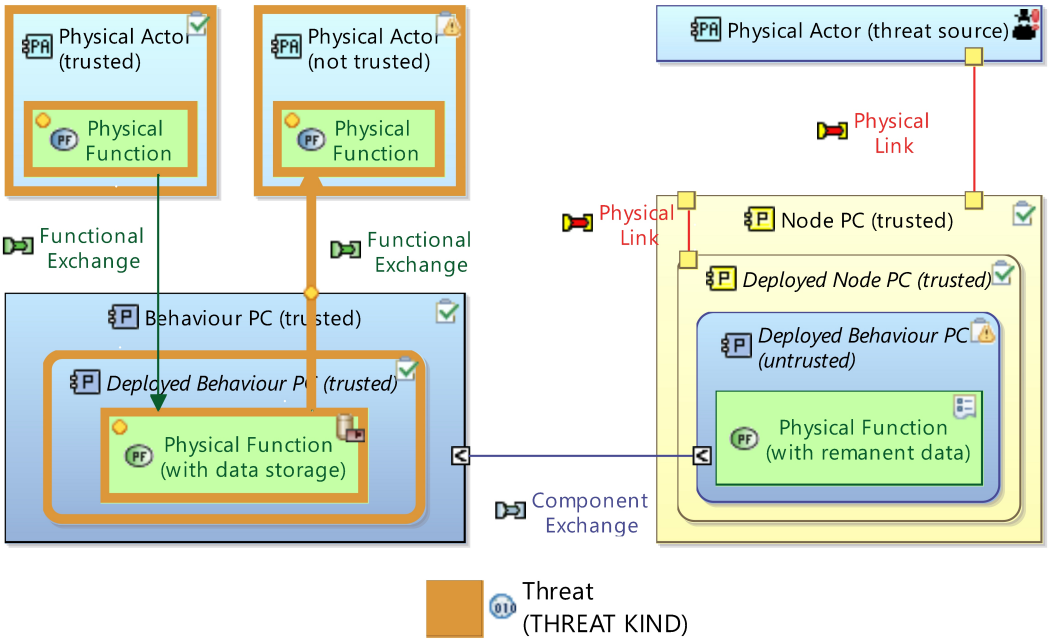
Figure 8: Capella - Threats Diagram

Exchanges in the same colour the selected Primary Assets or Threats, and adds an orange-yellow diamond icon. In this case, the selected Functional and Information Primary Assets are highlighted in the lilac and blue colour, respectively. If a threatened Physical Function or Component/Functional Exchange is related to more than one selected Primary Asset or Threat, the construct will be highlighted black. The second layer changes the background of trusted constructs to white while the background of untrusted/threatening constructs is changed to red. Furthermore, the relationships between trusted and untrusted/threatening constructs are marked with a red exclamation icon while their names are changed to red.
In Figure 9b, both the primary and the supporting asset layers are enabled. Apart from highlighting the threatened Physical Functions and Functional Exchanges in the same colour as the selected Primary Assets or Threats, the Physical Actors and first (Deployed) Behaviour PCs of threatened Physical Functions are highlighted in the same colours as well. In this case, the Threat ”Threat” is selected that has an orange colour. Again, the colour black is used in case a threatened construct is related to more than one selected Primary Assets or Threats (not illustrated). While primary assets are indicated by an orange-yellow diamond, supporting assets do not have such an icon.

Note that Exchange Items are not depicted in these figures. They represent sets of data that are semantically coherent with regards to their usage in a given context and can be marked as Information Primary Assets. Moreover, it should be noted that the Capella Cybersecurity viewpoint also introduces the constructs Primary Assets and Threats to the Exchange Scenario Diagrams, Class Diagrams and Function Dataflow Diagrams. Furthermore, the viewpoint introduces options to value Primary Assets on their level of confidentiality, integrity, availability and traceability. Since these diagrams and properties are not relevant for this study, they are not presented in this subsection.



(a) Primary Asset Layer & Trust Boundaries Layer Enabled



(b) Primary and Supporting Asset Layer Enabled

Figure 9: Capella - Physical Security Architecture Diagram

6.2.2 Additional Modelling Elements to Create CN Security Architectures

For this study, it is important to distinguish Primary Assets from Supporting Assets. Primary Assets represent information, activities, processes or functionality that are valuable to stakeholders. While Functional Primary Assets are always Physical Functions, Information Primary Assets can reside in Physical Functions or be transported through Functional Exchanges. Supporting Assets represent the Physical Actors or first (Deployed) Behaviour PCs in which threatened Physical Functions reside. Hence, while attacks are aimed at undermining valuable Primary Assets, they do this by targeting Supporting Assets. From a modelling perspective, attacks are exploiting vulnerabilities of Supporting Assets, and by doing this, threatening Primary Assets. This current set of modelling elements is graphically depicted in the light blue parts of Figure 10.

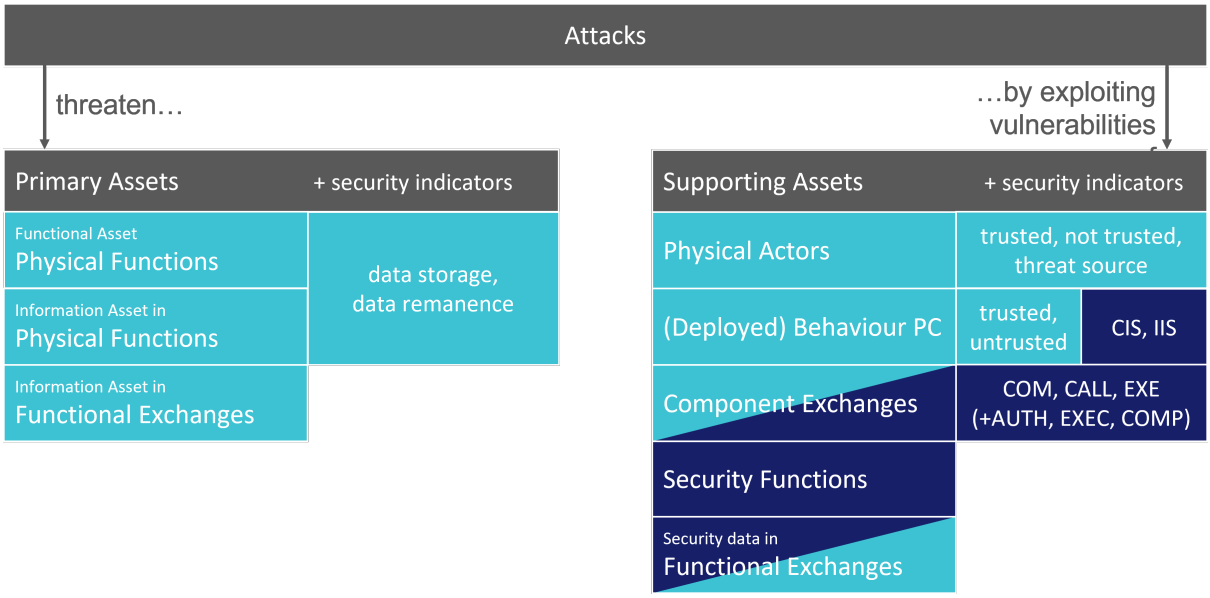
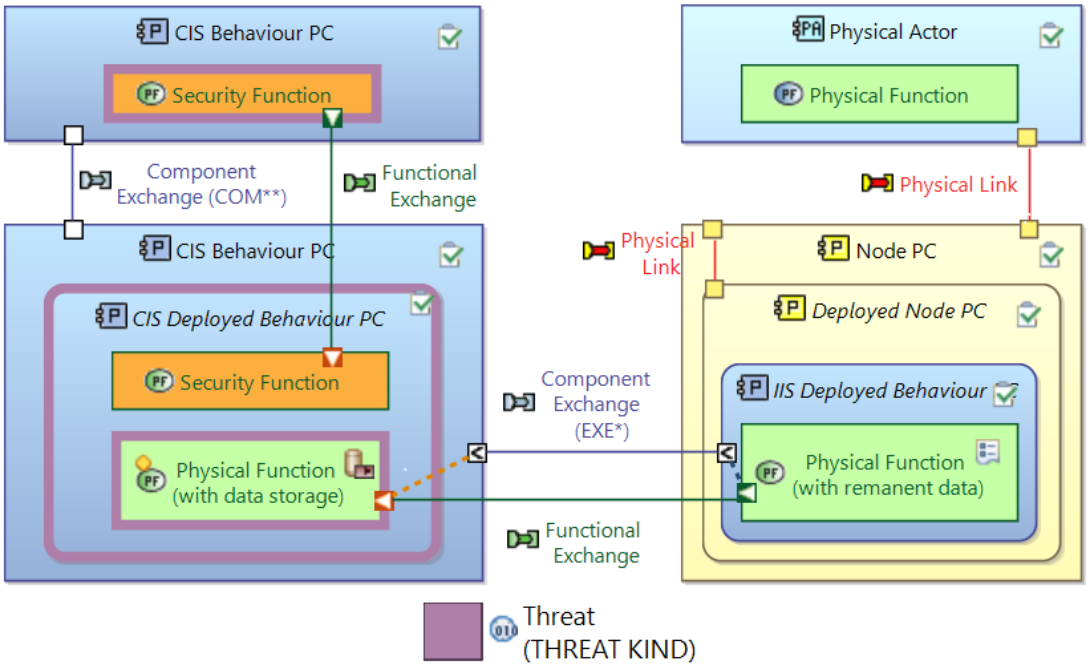


Figure 10: CN Ecosystems - Modelling Elements of Security Architectures

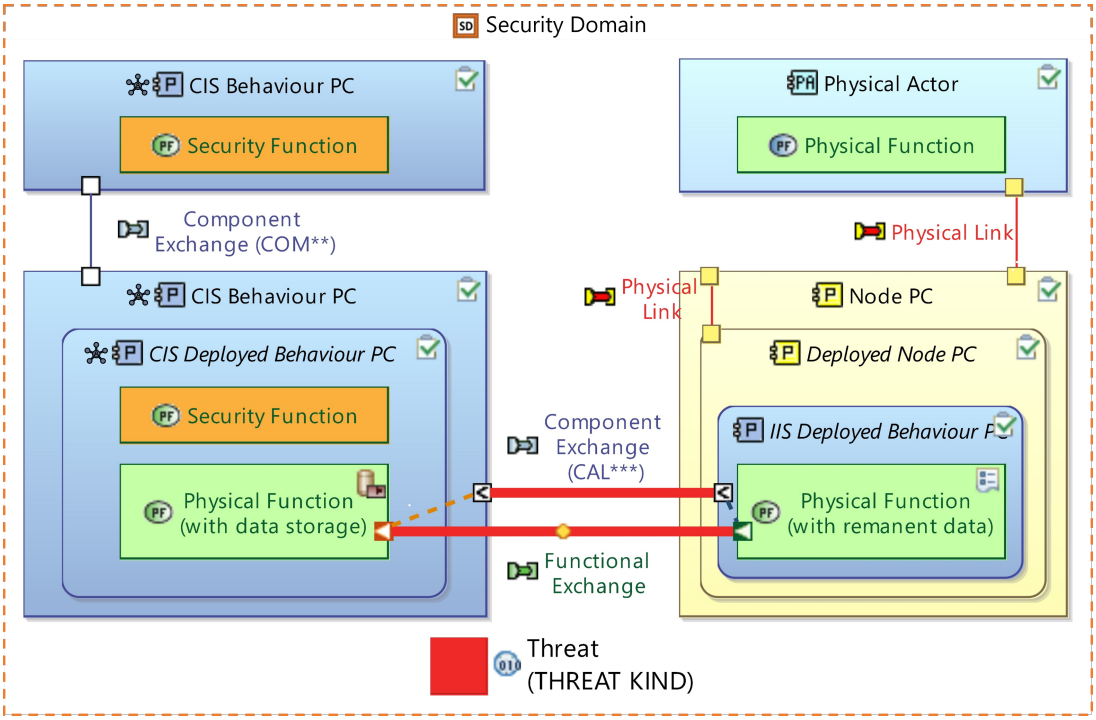
As the purpose of a security architecture is to model CN ecosystems such that their vulnerabilities can be uncovered and attacks assessed, several additional or changes in existing modelling elements are suggested. These are depicted in Figure 10 in the dark blue and dark/light blue parts, respectively.
Firstly, Component Exchanges that carry Functional Exchanges between (Deployed) Behaviour PCs should be considered Supporting Assets. This is because both platform/mobile entities and their relationships are vulnerable to attacks (see Section 5.1). While the former are represented in Capella by (Deployed) Behaviour PCs, the latter are represented by Component Exchanges. Furthermore, as different types of relationships are associated with different kinds of vulnerabilities and attacks (see Subsection 5.2.2), additional security icons need to be introduced. This eases users to uncover vulnerabilities and assess attacks that are specific to communication, calling or execution transport relationships. The necessary authentication, execution and competency trust relationships can be automatically inferred and indicated based on the type of transport relationship, the type of relationship pair (i.e. IIS–CIS or CIS–CIS) and the location of the relation (i.e. intra or inter-domain) - as presented in Table 5.

Secondly, a Security Function construct is suggested as a specialisation of a Physical Function. This distinguishes regular services from security services, allowing models to be validated regarding the presence and complete usage of the latter. Based on Table 1 and Table 8, the following security operations are at least necessary for CNs to be secure: *Authorisation and Access Control*; *Security Policy Management*; *Accountability*; *Privilege Management*; *Secure Association*; *Interdomain*; *Authentication and Security Attribution*; and *Cryptographic Support*. As the first three functions should be deployed on the security domain level (see the security implications of Chapter 1), they reside in separate (Deployed) Behaviour PC(s) that is/are connected to all other CIS-entities within the same security domain. Other security services should be deployed within the local (Deployed) Behaviour PCs. Since Security Functions are not valuable assets but are targeted by attacks to threaten Primary Assets, they are Supporting Asset. Furthermore, as with data from regular Physical Functions, the exchange of security data can be represented by Functional Exchanges. Hence, Functional Exchanges with security data are Supporting Assets as well.
Thirdly, as Supporting Assets are no longer limited to first (Deployed) Behaviour PCs or Physical Actors, automatically highlighting Supporting Assets is not possible anymore. Instead, the modeller has to choose what Supporting Asset(s) a specific attack is aimed at. This requires a non-visual change in the Capella Threats Diagram and the condition that Security Functions are predefined. Note that attacks can be modelled as threats.
Finally, to facilitate the automated determination of the required trust relationships and to assess whether all CIS-entities are connected to exactly one instance of all required security functions, there is a need to indicate whether a (Deployed) Behaviour PC is a CIS or a IIS. Naturally, Deployed Behaviour PCs should inherit the entity type from their Behaviour PCs. Again, the definitions of Section 5.1 can be adopted.

Considering the above, there are no visual changes needed that are specific to the Threats Diagram or the Primary Layer of the Physical Architecture Diagram. Hence, Figure 11 illustrates the modelling elements of a Capella Physical Architecture Diagram where the suggested additional modelling elements are included. Only the Primary and the adapted Supporting Assets Layers are enabled. The Trust Boundaries Layer is disabled as it is assumed that all collaborating entities can be considered trusted.
In both Figure 11a and Figure 11b, the names of all Component Exchanges have suffices between brackets, indicating the type of transport and trust relationship. The letters refer to the type of transport relationship: communication (COM), calling (CAL) or execution (EXE). The number of stars refers to the required type of trust relationship(s): authentication (*), AND execution (**), AND competency (***). Also illustrated in both figures are the Security Functions. Visually, they are similar to Physical Functions but with the difference that they have an orange background. Moreover, to distinguish CIS from IIS (Deployed) Behaviour PCs, a network icon is added in case of the former. This icon can be found on the left of the existing icon for (Deployed) Behaviour PCs.
In Figure 11a, a scenario is illustrated in which a Functional Primary Asset within a Physical Function is threatened as its Deployed Behaviour PC as well as its domain-wide Security Function is vulnerable to attacks. In Figure 11b, a scenario is illustrated in which an Information Primary Asset within a Functional Exchange is threatened because of vulnerabilities in the Component Exchange.



(a) Vulnerable Deployed Behaviour PC and domain-wide Security Function



(b) Vulnerable Component Exchange

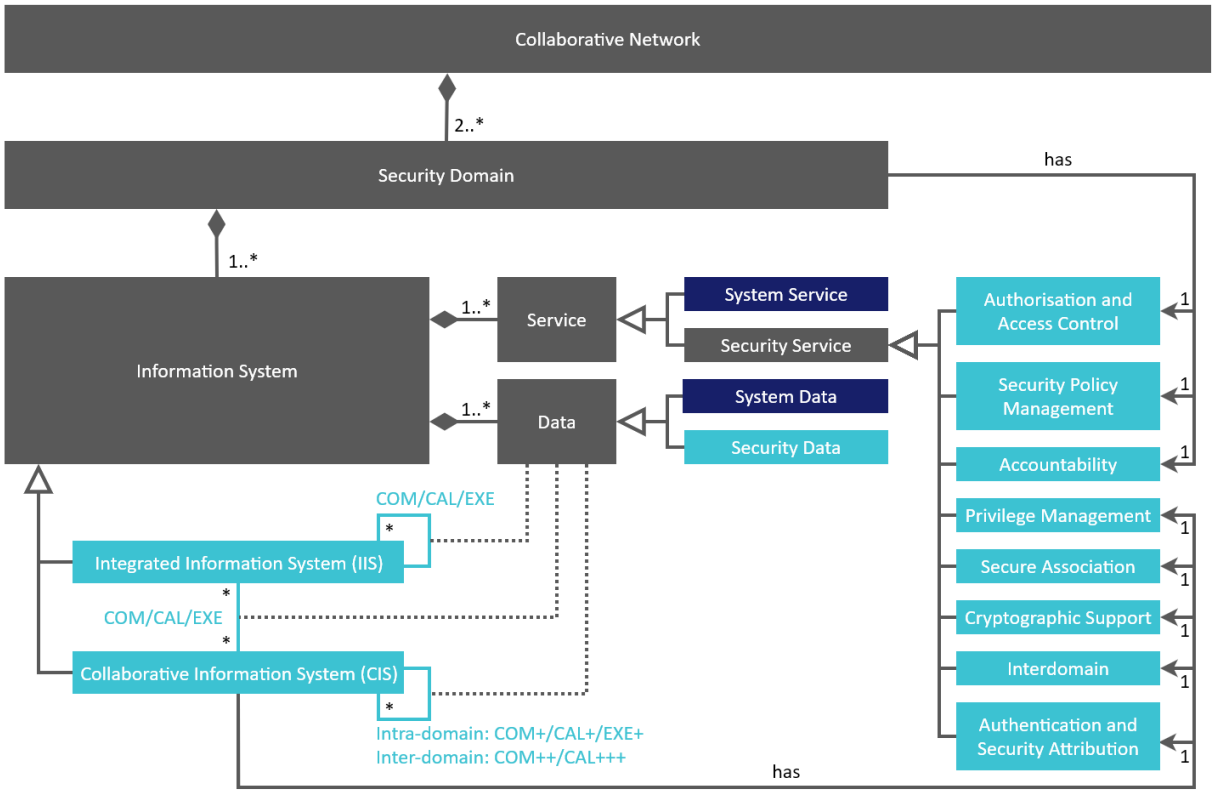
Figure 11: Capella - Extended Physical Security Architecture Diagram

6.3 Metamodel, Pattern and Viewpoints for CN Security Architectures

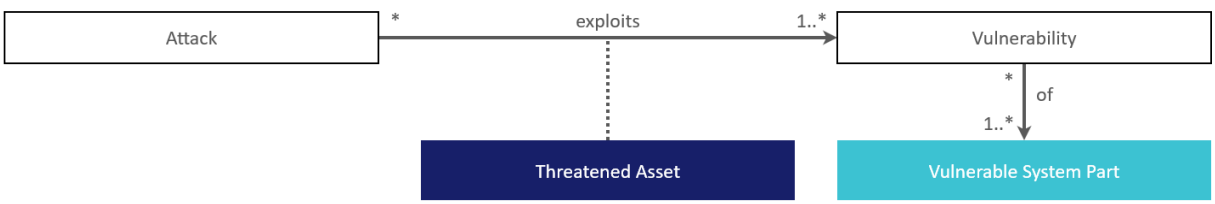
This section presents and describes a generic metamodel for CN security architectures along with its validation rules. In addition, a modelling pattern is suggested that illustrates the intended use of the metamodel and viewpoints are presented for modelling specific security aspects.

6.3.1 Generic Metamodel and Validation Rules

Figure 12 presents two UML Class Diagrams that, together, form a generic metamodel for CN security architectures. This SOA-metamodel is created based on the findings in the previous chapters and on the design decisions made in the previous sections. While Figure 12a presents the constructs and relationships that constitute a CN, Figure 12b places the constructs in their security context. A description of all the modelling elements in this metamodel can be found in Appendix A.



(a) Constructs and Relationships



(b) Constructs in Their Security Context

Figure 12: CN Ecosystems - Metamodel of Security Architectures

Figure 12a shows that each Collaborative Network (CN) is a collaboration between at least two Security Domains. A Security Domain consists of at least one Information System that could be a Integrated Information System (IIS) or a Collaborative Information System (CIS). Both types of Information Systems fulfil certain Services that read or write some Data. A distinction can be made between Services and Data that are specifically related to security and those that are not. For the Security Services, the following validation rules apply:

- Each Security Domain has domain-wide Security Services for *Authorisation and Access Control*; *Security Policy Management*; and *Accountability* operations.
- Each CIS has local Security Services for *Privilege Management*; *Secure Association*; *Interdomain*; *Authentication and Security Attribution*; and *Cryptographic Support* operations.
- Each CIS uses (i.e. is related to) all domain-wide Security Services.
- Within a Security Domain, domain-wide Security Service operations are not overlapping with each other.
- Within a CIS, local Security service operations are not overlapping with each other.
- Local Security Service operations are not overlapping with domain-wide Security Service operations.

IISs and CISs can be related by means of communication (COM), calling (CAL) or execution (EXE) transport relationships. For these relationships, the following validation rules apply:

- Intra-domain CIS–CIS COM, CAL and EXE transport relations require authentication trust (+).
- Inter-domain CIS–CIS relations are either COM or CAL transport relations.
- Inter-domain CIS–CIS COM transport relations require authentication and execution trust (++).
- Inter-domain CIS–CIS CAL transport relations require authentication, execution and competency trust (+++).

In Figure 12b, the constructs are presented in their context from a higher-level security perspective. The light blue constructs in Figure 12a are the Vulnerable System Parts that very likely have certain vulnerabilities. This includes all IISs, CISs, Security Services, Security Data and COM/CAL/EXE relationships. Attacks are exploiting one or more Vulnerabilities, and by doing this, threatening some asset. The Threatened Assets are the dark blue constructs in Figure 12a and include all System Services and System Data that reside in IISs or CISs. System Data can also be found in COM/CAL/EXE the relationships.

6.3.2 Modelling Pattern

Figure 13 presents a modelling pattern that illustrates the intended use of the metamodel. In this pattern, a scenario is sketched with two Security Domains that are collaborating in a CN. Each Security Domain has several IISs and CISs.

For IISs hold that they facilitate Security and System Services as well as their associated Data exchanges. These constructs are labelled with grey and *italic* names because they can be found in CN ecosystems but are outside the scope of this study because of their solely atomic characteristics. For CISs hold that they facilitate Local Security Services, System Services, and their associated Data exchanges. CIS (Security) is a special type of CIS that only facilitates domain-wide Security Services and their associated Security Data exchanges. Since this latter entity is a CIS, it needs a set of Local Security Services as well. Note that the CIS (Security) do not have relationships that cross Security Domains. Hence, in this particular pattern, the Interdomain Security Service can theoretically be omitted for this type of entity given that no additional cross-domain Security Services are defined.
With regards to the relationships, the following relationships can be found: IIS–IIS COM/CAL/EXE, IIS–CIS COM/CAL/EXE, intra-domain CIS–CIS COM+/CAL+/EXE+, and inter-domain CIS–CIS COM++/CAL+++. For all these types of relationships hold that both System and Security Data can be involved. Note that not all data associations are presented in the figure because of formatting considerations. The relationships involving CIS (security) are either COM+ or CAL+ because of the nature of Security Services. Evidently, these relationships might only involve Security Data.
As in Figure 12b, the Vulnerable System Parts are coloured in light blue while the Threatened Assets are coloured in dark blue. Again, an Attack exploits one or more Vulnerabilities of Vulnerable System Parts, and by doing this, they threaten one or more assets. Note that human entities are not modelled as this study assumes that all collaborating human entities are behaving benevolently (intentionally and unintentionally).

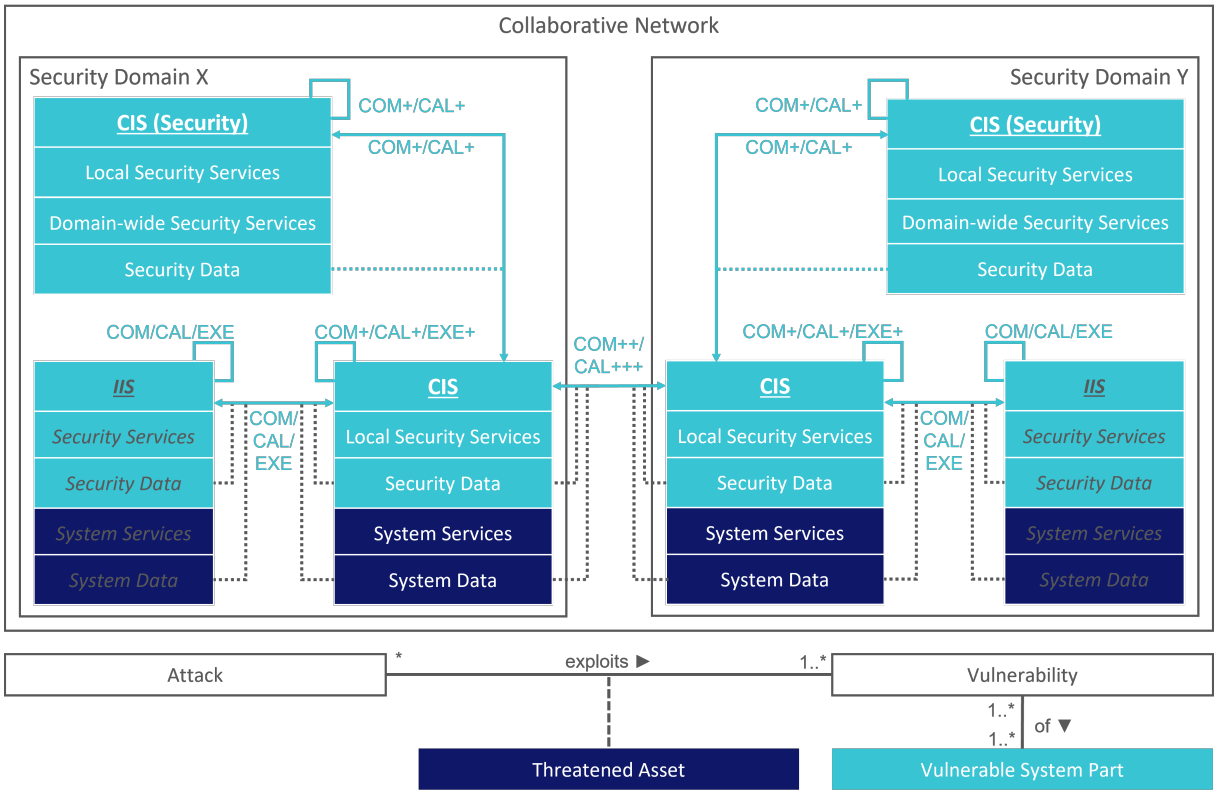


Figure 13: CN Ecosystems - Modelling Pattern of Security Architectures

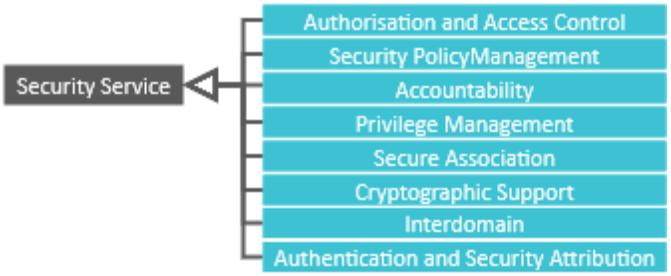
6.3.3 Viewpoints

For modelling specific security aspects, four viewpoints are defined from the generic metamodel for CN security architectures. These viewpoints are created based on the SABSA Matrix for security architecture [63], providing conventions for the construction, interpretation and use of architecture views to frame specific security concerns. Figure 15 presents the defined viewpoints, corresponding to the viewpoints that are marked light blue in the SABSA Matrix for security architecture as shown in Figure 14.

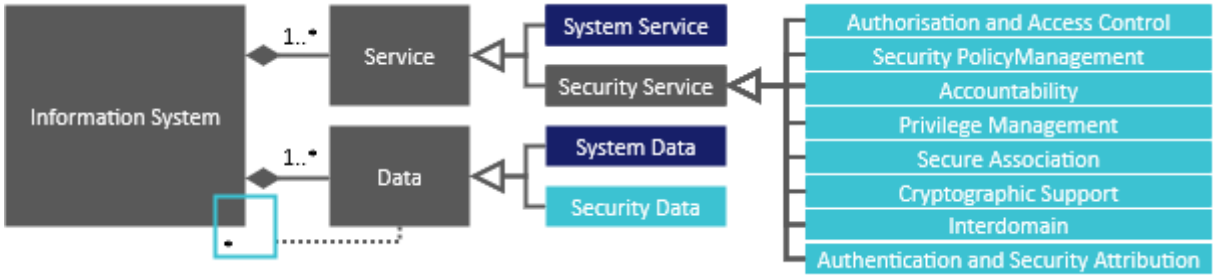
LAYERS	ASSETS (What)	MOTIVATION (Why)	PROCESS (How)	PEOPLE (Who)	LOCATION (Where)	TIME (When)	VIEWS
Contextual	The Business	Business Risk Model	Business Process Model	Business Organization & Relationships	Business Geography	Business Time Dependencies	Business
Conceptual	Business Attributes Profile	Control Objectives	Security Strategies & Architectural Layering	Security Entity Model & Trust Framework	Security Domain Model	Security-Related Lifetimes & Deadlines	Architect
Logical	Business Information Model	Security Policies	Security Services	Entity Schema & Privilege Profiles	Security Domain Definitions & Associations	Security Processing Cycle	Designer
Physical	Business Data Model	Security Rules, Practices, & Procedures	Security Mechanisms	Users, Applications, & the User Interface	Platform & Network Infrastructure	Control Structure Execution	Builder
Component	Detailed Data Structures	Security Standards	Security Products & Tools	Identities, Functions, Actions, & ACLs	Processes, Nodes, Addresses, & Protocols	Security Step Timing & Sequencing	Tradesman
Operational	Assurance of Operational Continuity	Operational Risk Management	Security Service Management & Support	Application, User Management, & Support	Security of Sites, Networks, & Platforms	Security Operations Schedule	Service Manager

Figure 14: SABSA Matrix for Security Architecture

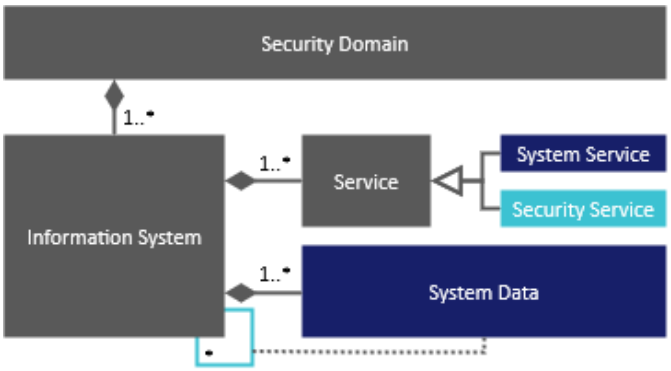
Figure 15a presents the Security Services Viewpoint which focuses on the security services and abstractions thereof. On a more detailed level, Figure 15b presents the Security Mechanisms Viewpoint which focuses on the interactions between and with security services within the context of system services. This includes the relationships between information systems, security data and their associations. System data can be modelled along with security data but this is not prioritised.
Figure 15c presents the Security Domain Model Viewpoint which focuses on the interactions of system services between and within security domains. This includes the relationships between information systems and system data. Security services can be modelled along with system services but this is not prioritised. On a lower level, Figure 15d presents the Security Domain Definition and Association Viewpoint. This viewpoint focuses on the interactions between and with security services within the constraining context of their information systems and security domains. This includes system and security data, and their associations.



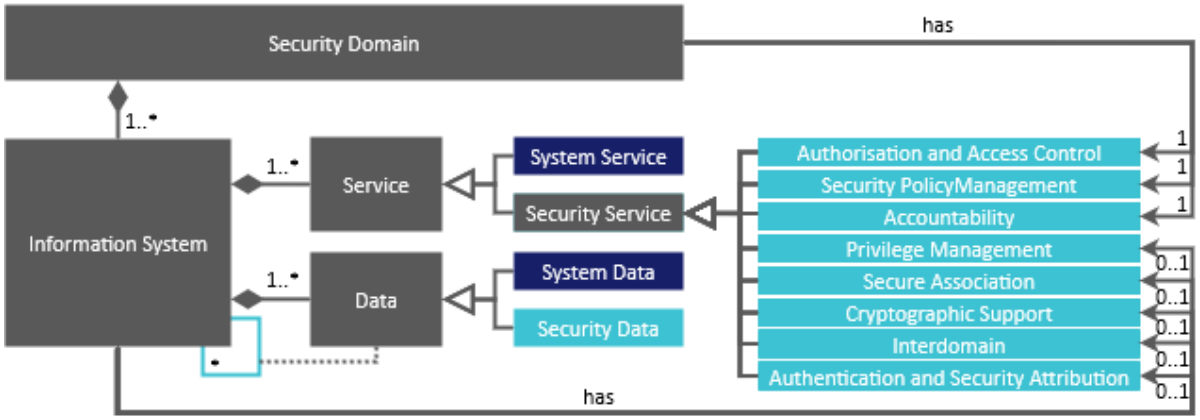
(a) Security Services Viewpoint



(b) Security Mechanisms Viewpoint



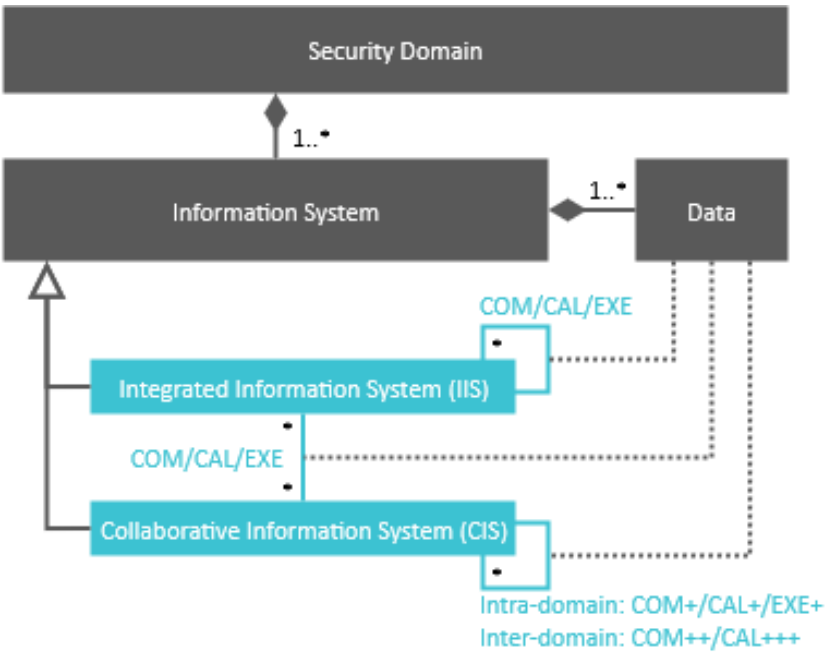
(c) Security Domain Model Viewpoint



(d) Security Domain Definitions and Associations

Figure 15: CN Ecosystems - Viewpoints of CN Metamodel (1)

The Platform & Network Infrastructure Viewpoint is marked dark blue in Figure 14. This is because the metamodel for CN can only partially facilitate the modelling of this viewpoint. As the metamodel distinguishes types of information systems and their corresponding transport relationships, it facilitates the modelling of the network infrastructures. However, since platform entities are intentionally left out for simplicity reasons, the metamodel does not support the modelling of physical platforms. (Note that platform entities, in this context, should not be confused with platform or mobile entities as defined in Chapter 1. Rather, platform entities refer to the entities on which information systems are running, similar to Node PCs as described in Section 6.1.)
Figure 16 presents the Network Infrastructure Viewpoint, a partial physical viewpoint, which focuses on the type of information systems and their transport relationships. This includes the required trust relations. System or security data, and their data associations, can be modelled but this is not prioritised.



(a) Network Infrastructure Viewpoint

Figure 16: CN Ecosystems - Viewpoints of CN Metamodel (2)

7 APPLICATION OF CN METAMODEL

This chapter demonstrates how the CN metamodel (see Chapter 6) can be applied to assert the security of CN security architectures. It illustrates how the CN metamodel can be applied to create security architectures within the context of the Arcadia architecture development methodology and how security analysis techniques (see Subsection 2.2.3) can be applied to assess the level of security in these architectures. The vulnerabilities of CNs, applied within the case on CC, establish a generalisable problem instance for answering research question 3:

RQ3 How can the metamodel for Collaborative Networks be applied to assert the security of CC architectures by design?

RQ3.1 How can the security metamodel for Collaborative Networks be used within the context of the Arcadia architecture development methodology?

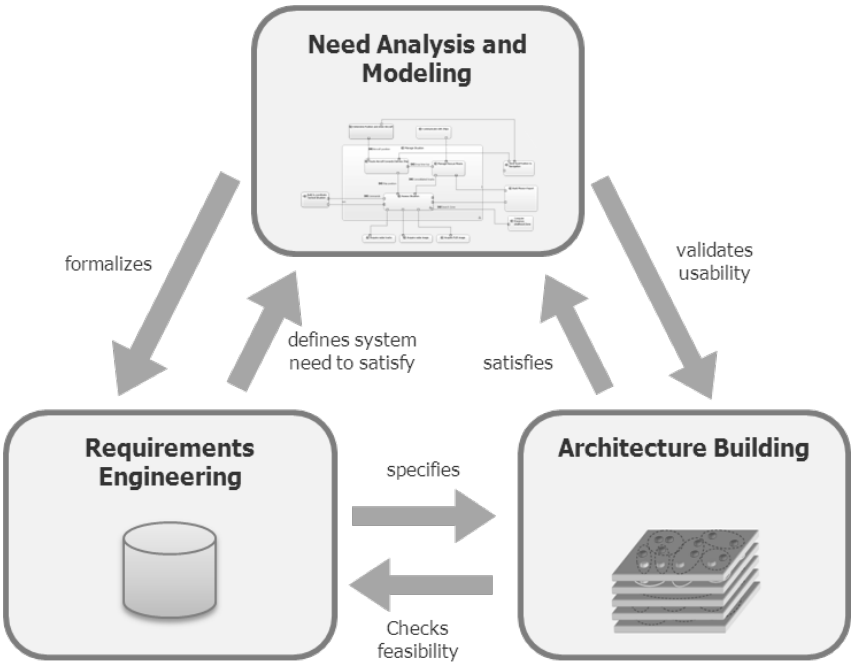
RQ3.2 How can the architectures that are created with the CN metamodel be analysed for their level of security?

7.1 Modelling System Architectures

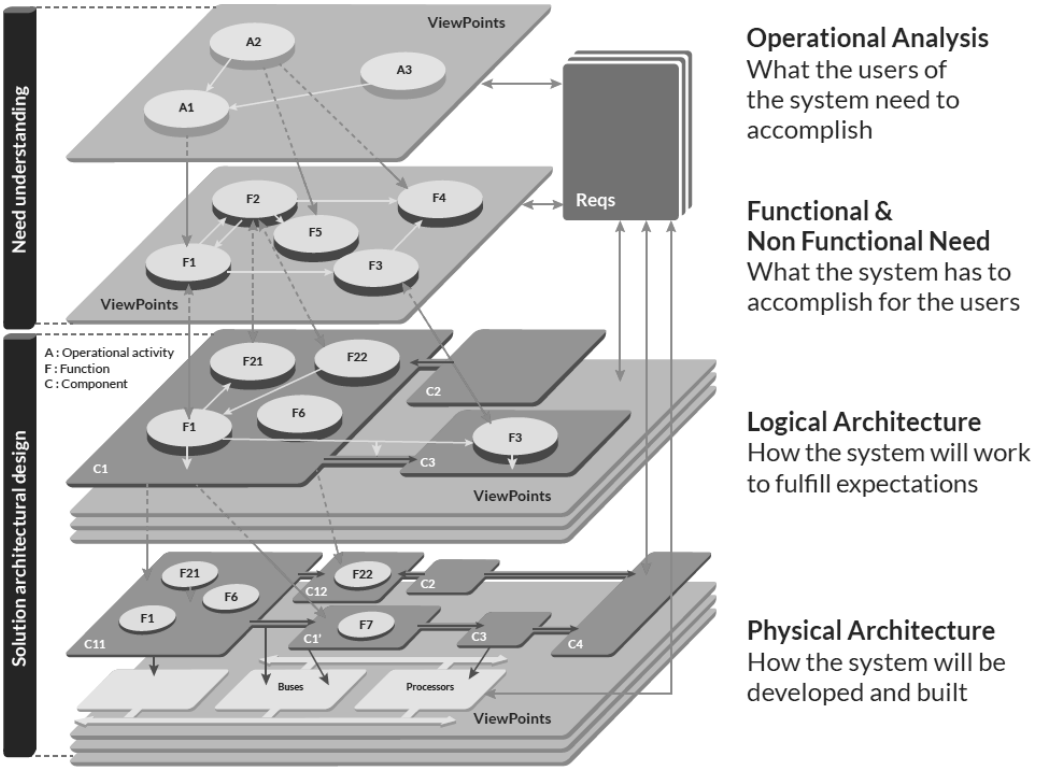
Originally developed by Thales and currently registered at the French Standardization Association, Arcadia [33] is a model-based engineering methodology for designing system, hardware and software architectures. This iterative method promotes a viewpoint-driven approach that distinguishes three related activities as presented in Figure 17a: needs analysis and modelling, architecture building and validation, and requirements engineering. As presented in Figure 17b, the three activities result in a number of architecture products, depicting different viewpoints.

7.1.1 Operational and System Analysis

In the needs analysis and modelling phase, operational and system needs are analysed. While the operational analysis focuses on what users have to accomplish (i.e. defining and allocating operational entities, capabilities and activities), the system analysis focuses on what the system has to accomplish for its users (i.e. defining and allocating actors, missions and functions). Similar to the planning phase of the Secure Mobile Grid Development Process of [19], there is no need to consider security during this early stage prior to architecture development as this is merely about user needs.



(a) Architecture Development Activities



(b) Architecture Development Products

Figure 17: Arcadia Architecture Development Methodology

7.1.2 Logical and Physical Architecture

With help of early design models, logical and physical architectures are built and validated - similar to the development phase of [19]. Based on the needs analysis, a logical architecture is created, showing how the system fulfils the needs of users by presenting viewpoints that cover specific concerns. For this, the system is decomposed into logical components and structured in such a way that a compromise exists between design drivers, (non-functional) constraints and viewpoints. The physical architecture shows how the final system development and IVVQ (Integration, Verification, Validation and Qualification) processes look like. This architecture includes the resource components that embed behavioural components and deals with the technical and developmental issues - favouring the separation of concerns, efficiency and safe component interactions.

Since security concerns should be considered as early as possible during architecture development, promoting security by design, it is advisable to include security services and their interactions in the Logical Architecture development stage. This allows the functional modelling of security facilities in the broader context of system functionality. Analogues to specialising Physical Security Functions from Physical Functions as presented in Subsection 6.2.2, the modelling element Logical Security Function can be defined as a specialisation of the existing Logical Function element.
Figure 18 presents a Security Mechanisms View in which the main interactions with and between the Security Functions are shown. More specifically, a partial Logical Architecture Diagram is created for the case on CC in which the main logical system and security interactions between C&C 1, InterAct and SkyControl 1 are depicted. In the diagram, it is shown that SkyControl 1 has System Functions for collecting and analysing battlefield data, and for controlling own and other effectors. In contrast, C&C 1 can only control their own effectors. The exchange of battlefield data and effector instructions is facilitated by dedicated InterAct System Functions. Since the information systems are geographically distanced from each other, it is assumed that they communicate by means of satellite connections.
The Logical Security Functions are modelled as Logical Functions but with an orange background. While the local Security Functions can be found along with the Logical Functions within the Logical Components, the domain-wide Security Functions are facilitated by a separated Logical Component for each security domain. In summary, the Secure Association Function creates associations with other entities (e.g. through satellite or LAN connections) and applies transport security. Once the entity is connected, the Authentication and Security Attribution function verifies its identity information. The verified information is passed to the Authorisation and Access Control Function which makes access control decisions based on security policies that are collected by the Security Policy Management Function. The access control decision is fed back to the Authentication and Security Attribution Function that returns privileges. These privileges are managed by the Privilege Management Function for delegation purposes and are communicated to the Authentication and Security Attribution Function of the other entity. The access control decision is fed back to the Secure Association Function to be communicated with the Secure Association Function of the other entity. In

case of interdomain exchanges, the Interdomain Functions of the collaborating entities establish sessions between the security domains, and exchanges security policy and domain information. While the security policies of other security domains are passed to the Security Policy Management Function, the policies of the own security domain are shared by this Security Function. Finally, the identity and privilege information is exchanged with the System Functions for further system operations.
Note that the Interdomain Function of C&C 1 is modelled even though it is not in use. This is because CN ecosystems are adaptive, meaning that interdomain relations can be established dynamically, requiring this Security Function to be present when necessary. Furthermore, note that the Functional Exchanges from the Cryptographic Support and Accountability Functions are not modelled. For the Cryptographic Support Function, this is because this Security Function is responsible for information protection, meaning that there are no specific information exchanges that are relevant to model. For the Accountability Function, this is because this Function is responsible for all recording all security operations, meaning that this Function has Functional Exchanges with all Security Functions. For readability reasons, the latter is not modelled. Finally, note that the information systems that facilitate domain-wide Security Functions also have their own local security functions. These Security Functions are excluded from this diagram as they do not directly affect system services.

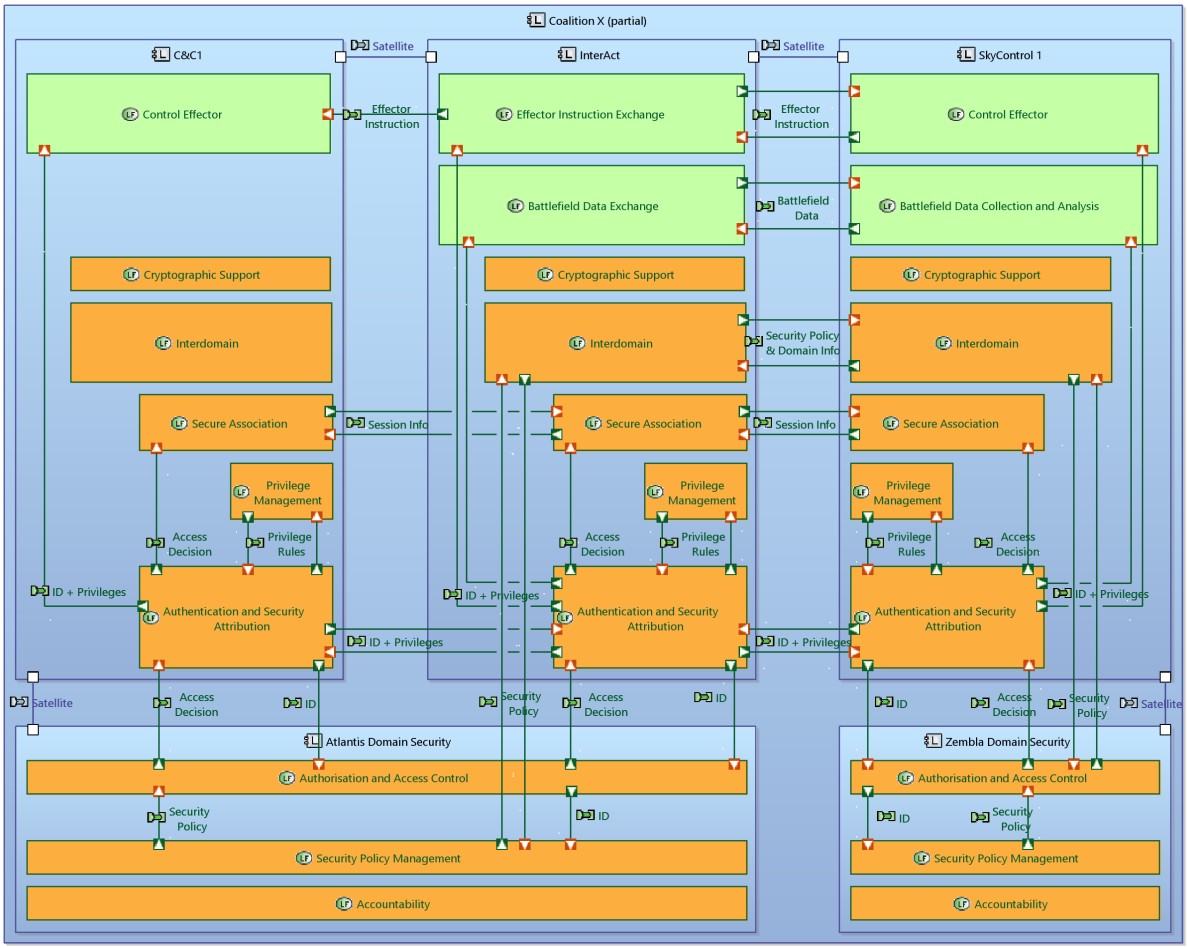
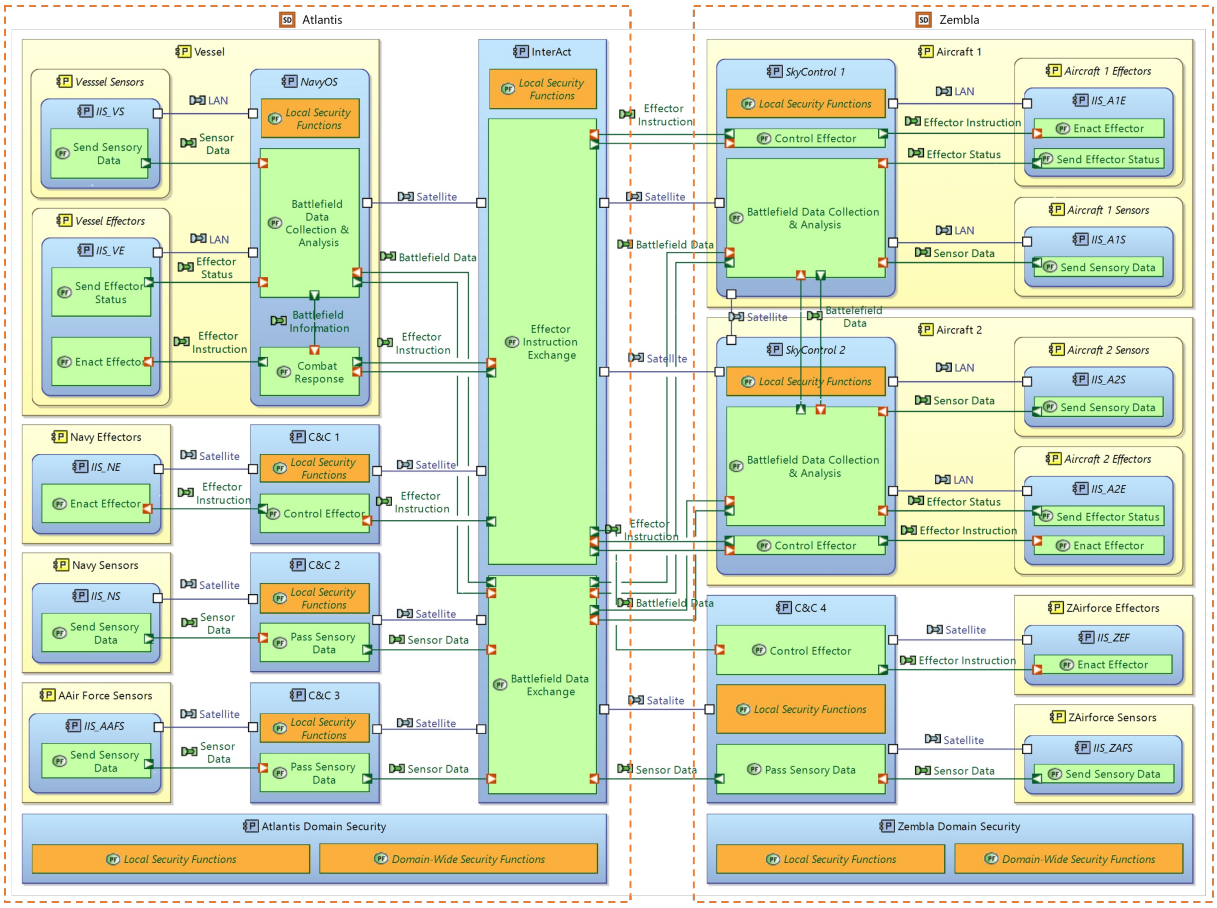


Figure 18: CC Case - Logical System Architecture

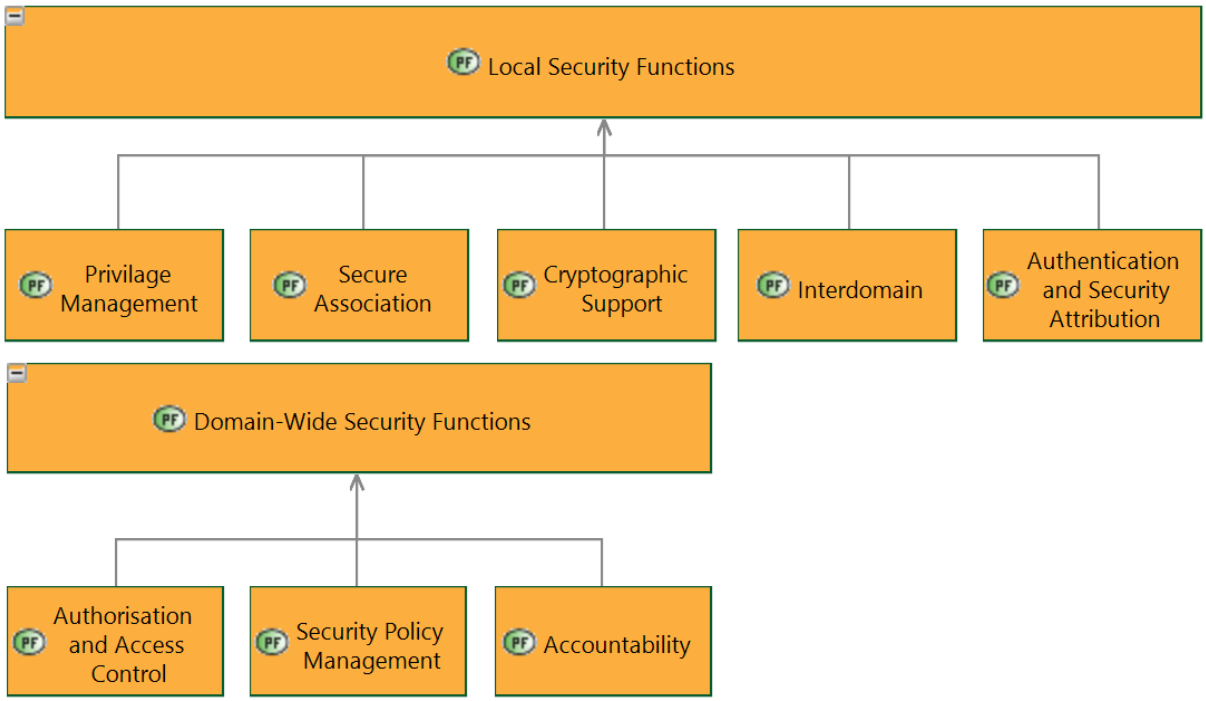
Using the modelling pattern as presented in Figure 13 and the partial Logical Architecture Diagram as presented in Figure 18, two Physical Architecture Diagrams are created. These diagrams are created using the modelling elements as described and extended in Section 6.1, and present different security viewpoints as defined in Subsection 6.3.3. While Figure 19a depicts a high-level system architecture that focuses on system services and their security domains, Figure 20 depicts a lower-level partial system architecture that focuses on security services and their mechanisms.
Figure 19a is a Security Domain Model View and depicts the high-level system architecture of the whole CN ecosystem as described in the case on CC. For both security domains, the diagram presents the physical entities (i.e. Node PCs), the information systems (i.e. Behaviour PCs) with their system services (i.e. Physical System Functions), and the main system interactions within and between these information systems. These system interactions include the data or information that is exchanged (i.e. Physical Exchanges) as well as the medium through which this happens (i.e. Component Exchanges). As this diagram focuses on the system services rather than the security services, the latter are abstracted by means of encapsulations through Physical Security Functions as presented in Figure 19b (using the Security Services Viewpoint). For both local and domain-wide security services, a single encapsulating Physical Security Function is defined for modularity reasons. While local Security Functions can be found in each CIS, domain-wide Security Functions are facilitated by a separate dedicated CIS for each Security Domain. In this diagram, the interactions with and between Security Functions are not depicted for readability reasons. Note that the security facilities from the IIS are not depicted either as this is outside the scope of this study.
In contrast, Figure 20 is a Security Domain Definitions and Associations View, presenting a lower-level system architecture for the Aircraft 1 and the InterAct systems. The Security Functions are modelled similarly as in the Logical Architecture Diagram. The main difference is that the local Security Functions are also modelled for the Atlantis and Zembla Domain Security Behaviour PCs, along with their Functional Exchanges. To distinguish local Security Functions and Functional Exchanges from domain-wide Security Functions and Functional Exchanges, the local Security Functions and Functional Exchanges are marked red within the Domain Security PCs. Note that the local Interdomain Security Function is omitted in these Behaviour PCs. This is because these entities are only interacting with entities within the same security domain, as defined in the modelling pattern. Again, the security facilities from the IIS are not depicted as this is outside the scope of this study.

7.1.3 End Product Breakdown Structure

Based on the activities above, a building strategy is created which formally specifies system components and their integrations. Component integration contracts collect all component properties, and enforce that system and integration requirements are met. Functional and non-functional security requirements are created and consumed throughout all activities. Since this is outside the scope of this study, these activities are not illustrated.
Note that post-integration activities such as architecture maintenance (e.g. [19]), transition (e.g. [34]) and governance (e.g. [35]) are not part of the Arcadia methodology.



(a) Physical Architecture Diagram



(b) Functional Breakdown Diagram

Figure 19: CC Case - Physical System Architecture (1)

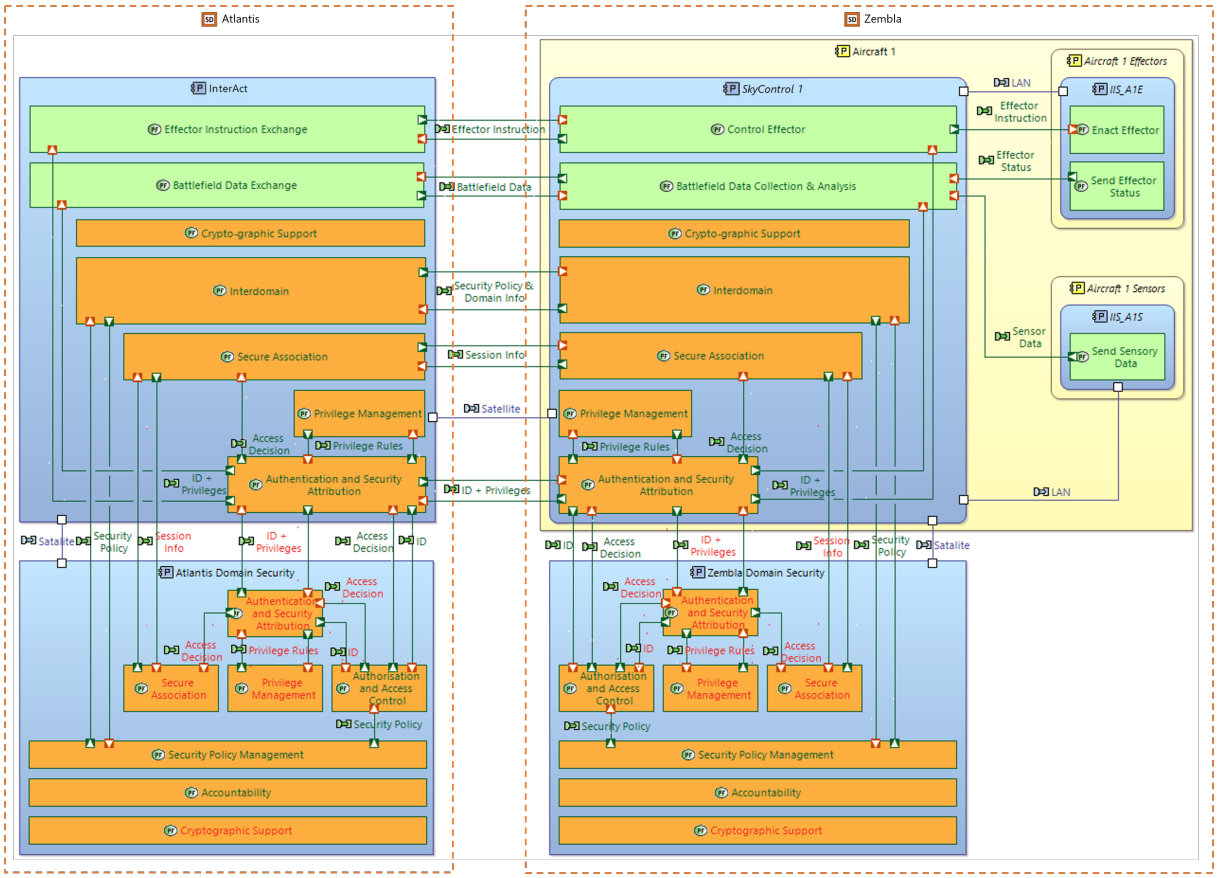


Figure 20: CC Case - Physical System Architecture (2)

7.2 Modelling Security Architectures

Based on the partial Physical System Architecture Diagram as presented in Figure 20, a partial Physical Security Architecture diagram is created. This system security architecture diagram, presented in Figure 21, is the same diagram as the system architecture diagram but enriched with indicators regarding data functionality and communication security. These security indicators are modelled as described and extended in Section 6.2, and in accordance with Figure 4 and Figure 5.

The remainder of this section illustrates how attacks A1, A8 and A12 can be modelled within the partial Physical Security Architecture. These attacks are representative for all types of studied attacks (see Table 6) and the vulnerable system parts they exploit (Table 7). All other attacks can be modelled in a similar way based on which ecosystem vulnerabilities can be uncovered and attacks assessed.

Figure 22 presents a Threat Diagram which depicts the three selected attacks and two Primary Assets they might affect. For illustrative purposes, it shows that the effector-controlling functionality of Coalition X can be affected by circumventing inter-domain CIS–CIS relationships (A1). Similarly, the figure shows that their battlefield data can be affected by an unauthorised intervention of any relationship (A12). Finally, it is shown that both Primary Assets can be affected by masquerading platform or mobile entities (A8).

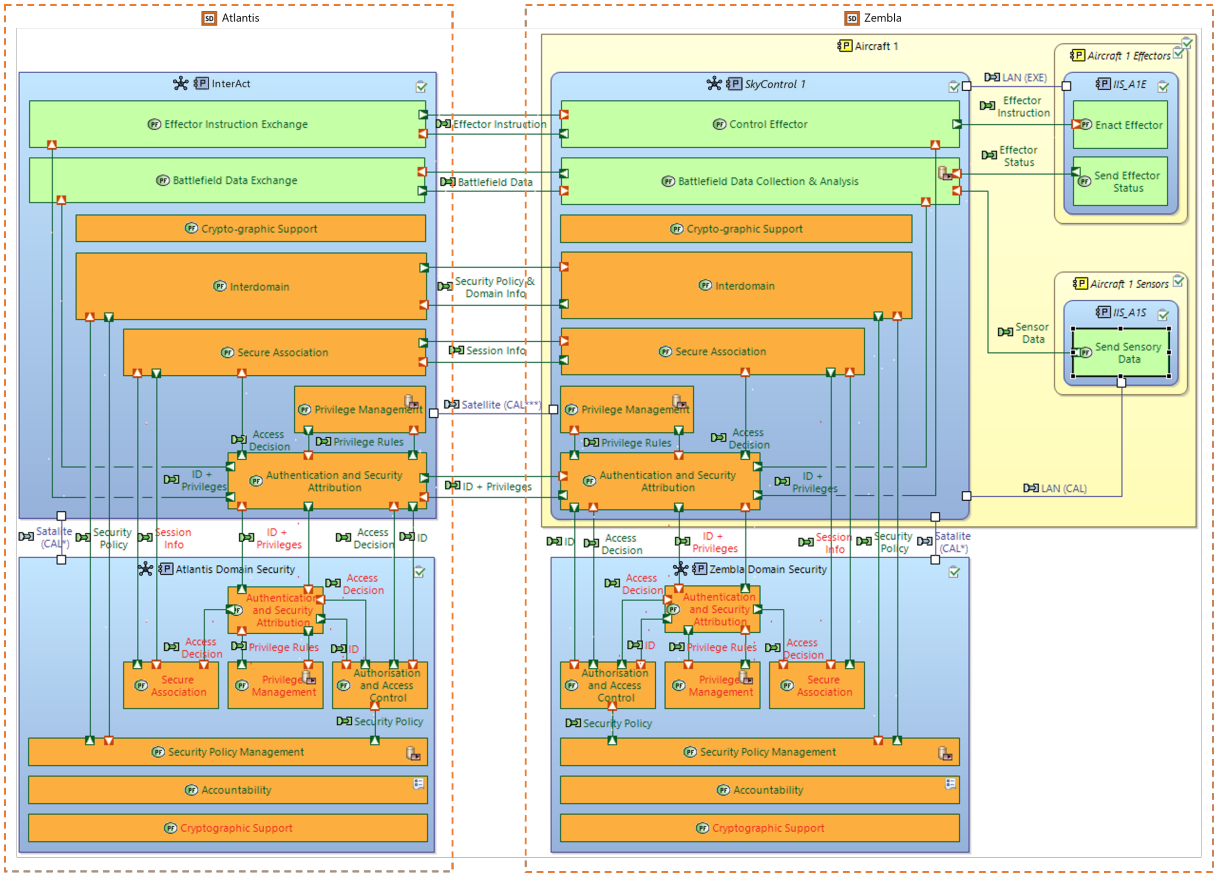


Figure 21: CC Case - Physical Security Architecture

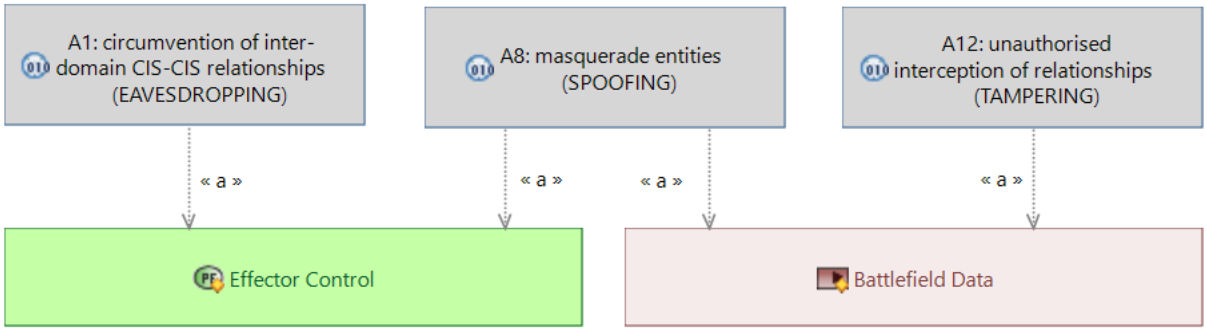


Figure 22: CC Case - Attacks and Primary Assets

Figure 23 presents a Physical Security Architecture Diagram in which it is indicated where the Primary Assets can be found within the architecture description. The Physical Functions that are related to the Functional Primary Asset *Effector Control*, (i.e. *Effector Instruction Exchange*, *Control Effector* and *Enact Effector*) are marked by a neon blue border and an orange-yellow diamond. The modelling elements that are related to the Exchange Items - containing the Information Primary Asset *Battlefield Data* - are marked with a bordeaux red border and an orange-yellow diamond. This includes the Physical Functions *Battlefield Data Exchange*, *Battlefield Data Collection & Analysis*, *Send Effector Status* and *Send Sensory Data*, as well as their Functional Exchanges *Battlefield Data*, *Effector Status* and *Sensor Data*.

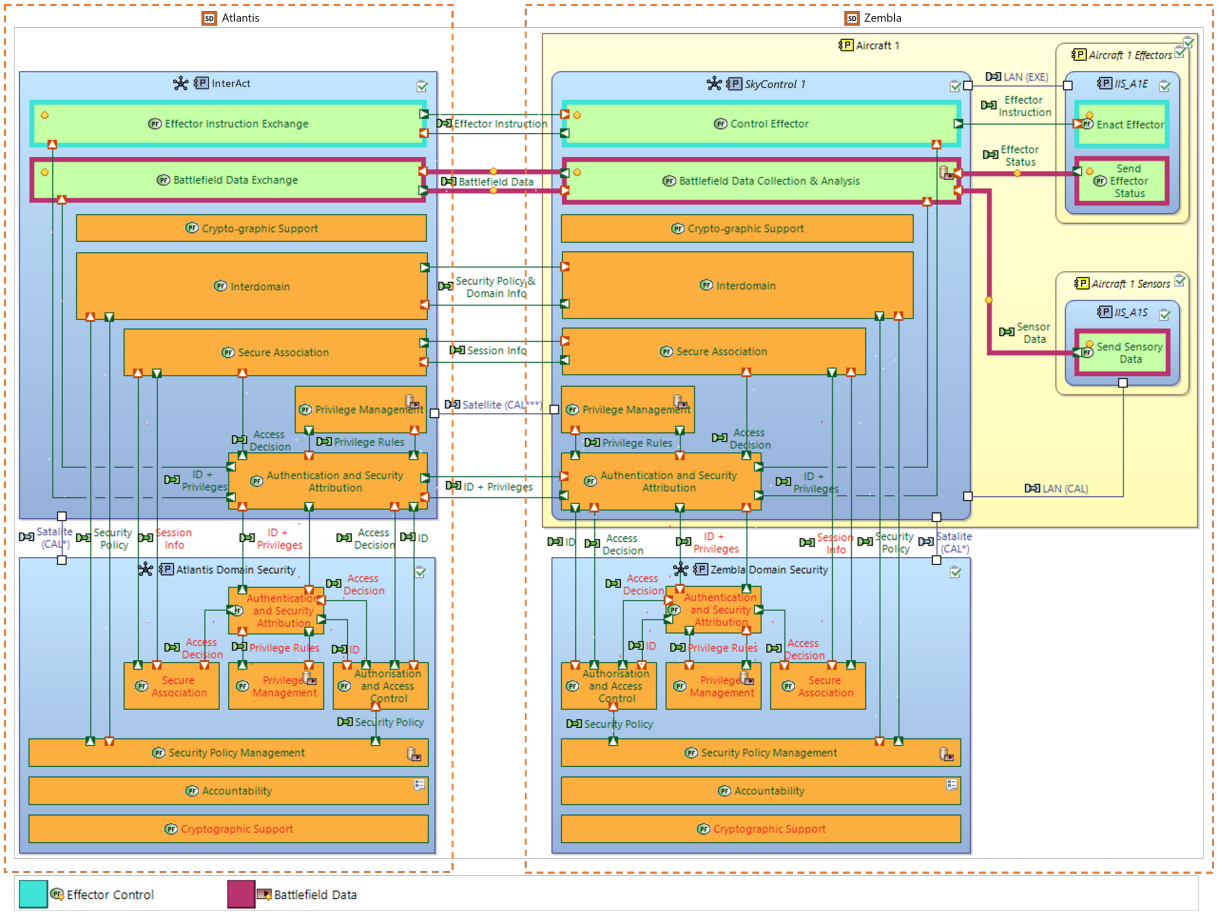
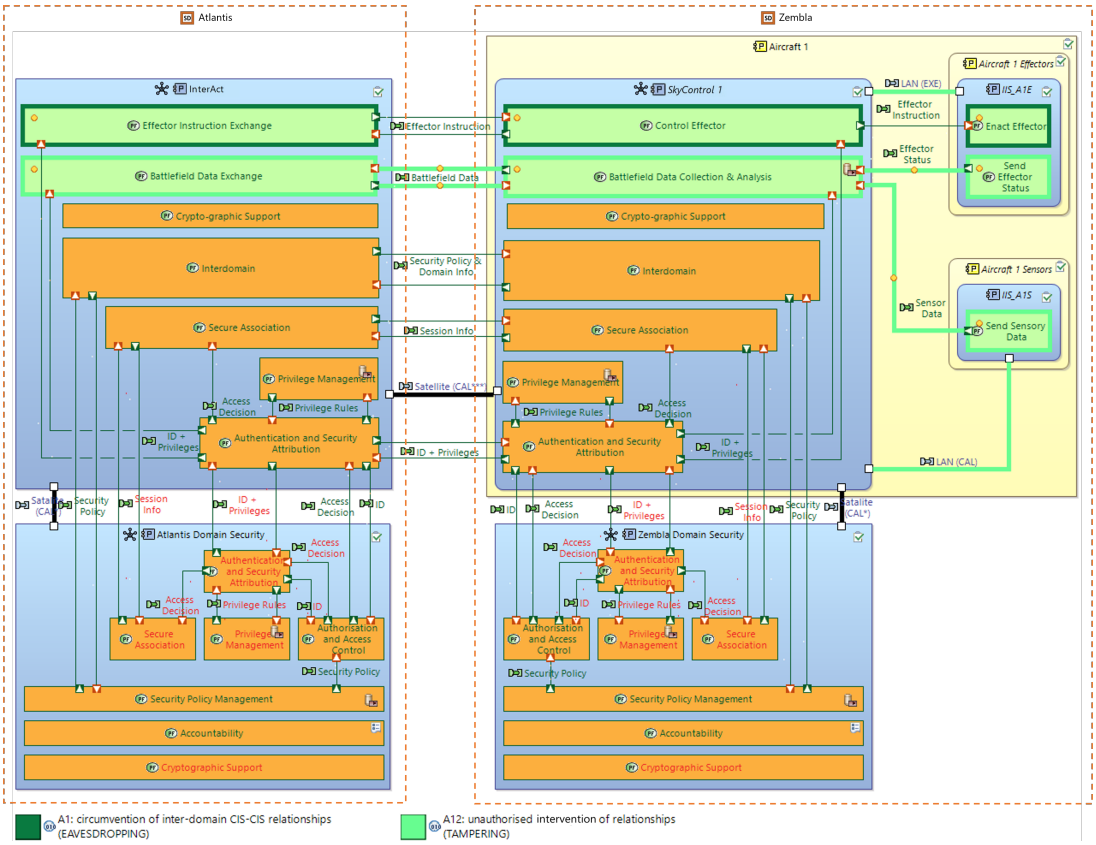
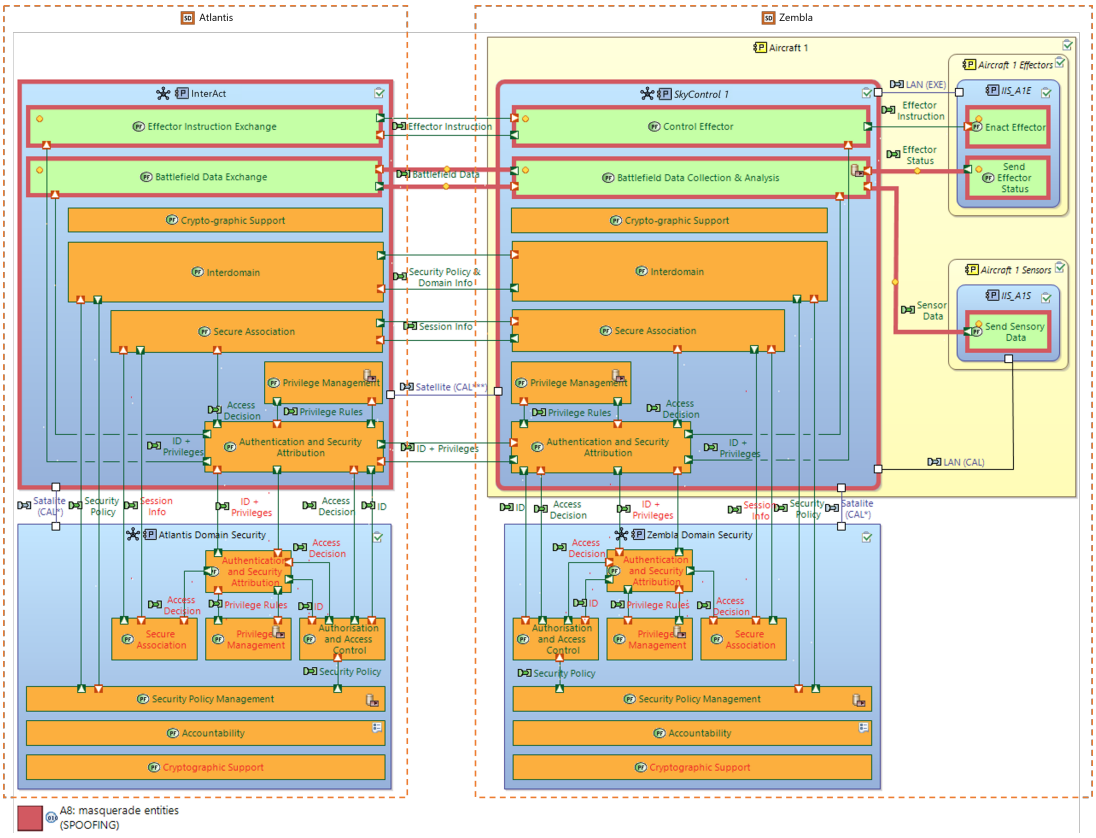


Figure 23: CC Case - Physical Security Architecture (Primary Assets)

Figure 24 presents two Physical Security Architecture Diagrams in which the locations of both the Primary and Supporting Assets are depicted for the selected attacks and their cascading effects. Marked in neon green (and without orange-yellow diamonds), Figure 24a shows that attack A12 can exploit vulnerabilities of the Component Exchanges *LAN (CALL)* or *LAN (EXE)*. The Component Exchanges *Satellite (CALL***)* and *Satellite (CAL*)* are marked in black as both attacks A1 and A12 can exploit vulnerabilities of these exchanges. In Figure 24b, the borders of the CIS entities *InterAct* and *SkyControl 1* are marked red as their vulnerabilities can be exploited by attack A8.



(a) Attacks A1 and A12



(b) Attacks A8

Figure 24: CC Case - Physical Security Architecture (Attacks)

7.3 Analysing Security Architectures

As presented in Subsection 2.2.3, many different frameworks exist that can be used to analyse the security of system (security) architectures. This section illustrates how the security properties of CNs can be analysed in line with Breu, Innerhofer-Oberperfler, *et al.* [55]. For this, the partial security architecture from the case on CC - as presented in Figure 21 - is used as the motivating example.

The security analysis framework as presented in [55] supports the identification of the weakest links in an enterprise security system, the identification of risks that have the highest business impact, and the valuation of security investments. This is achieved by creating layered threat graphs that depict concrete (context-independent) threats, target (context-dependent) threats and the cascading effects thereof. On these graphs, measures are defined that quantify the propagation effect of successful attacks; the ratio of attacks that results in successful breaches of security requirements; and the business losses caused by failure to fulfil security objectives.
Since this study focuses on attacks rather than threats, an attack graph is presented in Figure 25. For illustrative purposes, this layered graph shows how concrete attacks from Table 6 (in light blue) propagate to system-specific target attacks (in dark blue) - violating the business security objective *prevent unauthorised access to battlefield data*. First, the propagation of attacks is presented. Then, the graph measures are estimated and justified. Finally, the system measures are calculated based on the graph measures.

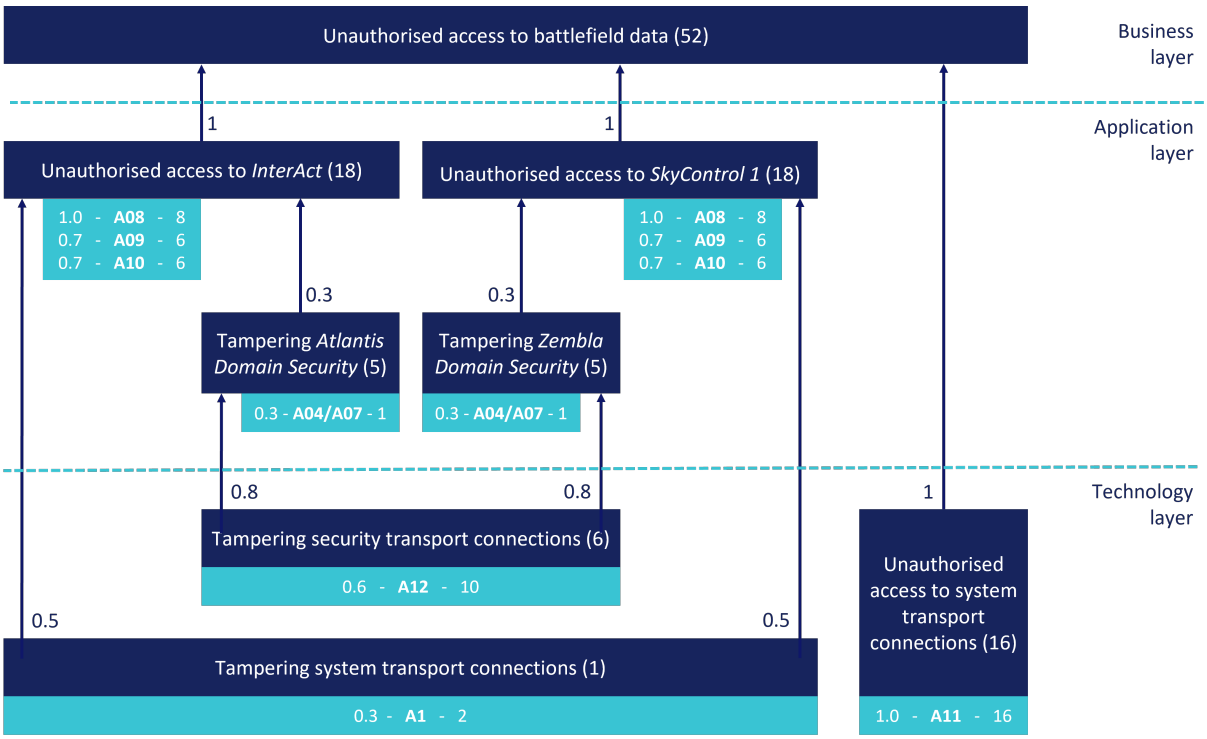


Figure 25: CC Case - Attack Graph and Measures

7.3.1 The Propagation of Attacks

In the graph, attacks are categorised in layers, similar to those of [60]. In the business layer, the business security attack of gaining unauthorised access to battlefield data is presented. As this study assumes that all human entities are secure, no other attacks are presented in this layer. The business security attack is a direct result of the unauthorised access to InterAct or SkyControl 1 in the application layer, or to a system connection that transports battlefield data in the technology layer.
To gain unauthorised access to InterAct or SkyControl 1, attackers can deploy concrete attacks A8 (i.e. masquerading), A9 (i.e. unauthorised replication or modification) or A10 (i.e. theft of rights or delegation misuse). Alternatively, attackers can also get unauthorised access to InterAct or SkyControl 1 by tampering system transport connections in the technology layer; this through concrete attack A1 (i.e. circumvention). As an addition, attackers might also tamper with domain-wide security mechanisms. Directly, this can be achieved through concrete attack A4/A7 (i.e. repudiation). Indirectly, this can be achieved by tampering the security transport connections in the technology layer; this through concrete attack A12 (i.e. unauthorised intervention).
To gain unauthorised access to the system transport connections, concrete attack A11 (i.e. eavesdropping) can be deployed.

7.3.2 Graph Measures

In the graph, two measures are presented of which their values are estimated based on educated guesses: the propagation effect of successful attacks and the ratio of concrete attacks that results in successful breaches of security requirements.

The propagation effect of successful attacks is the probability that a target attack occurs if a source (target or concrete) attack is successful. These weights can be seen as a function of (1) the attractiveness of the destination target attack for an attacker and (2) the impact of the source attack on the destination target attack. Between two target attacks, the weights are presented on the edges. Between a concrete attack and a target attack, the edges are not depicted to reduce the complexity of the graph. In this case, the weights are presented on the left of the concrete attacks. For illustrative purposes, the attractiveness and impact are estimated based on an ordinal scale of *low*, *mid* and *high*. Mapping these categories to the numeric values 0.1, 0.5 and 1.0, respectively, the propagation weight is calculated by multiplying the attractiveness factor with 0.4 and the impact factor with 0.6. More weight is given to the impact as this is the purpose of an attack, irrespective of the target. This gives:

$$PropagationWeight = 0.4 \cdot Attractiveness + 0.6 \cdot Impact$$

where

$$Attractiveness, Impact = \begin{cases} 0.1, & \text{low} \\ 0.5, & \text{mid} \\ 1.0, & \text{high} \end{cases}$$

All edges that lead to the business security attack is weighed with 1.0. This is because it can be argued that battlefield data is always compromised (high) once an attacker has access to InterAct, SkyControl 1 or a system transport connection that transports battlefield data (high). The edges that originate from tampering the domain-wide security systems are weighed with 0.3. This is because the impact is only mediating, and therefore limited (mid), while the attack itself is relatively hard to perform well due to the intra-domain characteristics of the domain-wide security entities (low). The edges that originate from tampering the security and system transport connections are weighted with 0.8 and 0.5, respectively. This is because tampering with security transport connectors will likely cause many domain-wide security mechanisms (mid) to malfunction (high). In contrast, tampering with system transport connections will likely have a very limited contribution (low) to accessing the InterAct or SkyControl 1 systems (high).
Similarly, it can be argued that the concrete attacks A1, A4/A7/A12 and A8/A9/A10/A11 have low, mid and high levels of attractiveness, respectively. With regards to their impact, attacks A4/A7, A1/A9/A10 and A8/A11/A12 have low, mid and high levels, respectively.

The ratio of concrete attacks that results in successful breaches of security requirements is quantified by the estimated number of times a concrete attack occurs. This is presented as an integer on the right of a concrete attack. For this illustration, an arbitrary reference period is chosen such that the expected number of concrete attacks is calculated by multiplying the propagation weight with 16 - with corrections for the type of attack and their spreading over multiple entities. It is assumed that internal attacks are 50% less likely to occur than external attacks and that attacks (with the same amount of impact and attractiveness) are spread evenly over multiple entities. This gives:

$$NrConcreteAttacks =$$

$$16 \cdot PropagationWeight \cdot AttackTypeFactor \cdot AttackSpreadFactor$$

where

$$AttackTypeFactor = \begin{cases} 0.5, & \text{internal} \\ 1.0, & \text{otherwise} \end{cases}$$

and

$$AttackSpreadFactor = \frac{1}{NrSimilarEntities}$$

7.3.3 System Measures

Using the estimations of the graph measures above, the ratio of target attacks that results in successful breaches of security requirements is calculated. From this measure, the losses caused by failure to fulfil the business security objective is calculated. With all measures determined, the weakest links in the CN can be identified as well as the attacks that have the highest business impact. In addition, the value of security investments can be calculated.

The ratio of target attacks that results in successful breaches of security requirements is quantified by the estimated number of times a target attack occurs. This number is presented as integers between brackets for each target attack and can be calculated using a bottom-up approach. This means that the estimated number of a target attack occurrences is determined by the accumulated number of propagated target attacks and the expected number of direct target attacks. The latter is calculated by multiplying the expected number of concrete attacks with their propagation weight. This gives:

$$NrTargetAttacks = NrPropagatedAttacks + NrConcreteAttacks \cdot PropagationWeight$$

From a system perspective, Figure 25 shows that InterAct, SkyControl 1, system transport connections and security transport connections are all weak links of the partial security architecture when it comes to preventing unauthorised access to battlefield data. With an attack propagation weight of 0.3 and an attack occurrence rate of 5, the domain-wide security systems play a less significant role. Looking at the concrete attacks, attacks A08 and A11 are most impactful, followed by attacks A09, A10 and A12. Since the numbers of concrete attack occurrences are unknown, they are calculated based on the propagation weight, meaning that the same attacks are also more likely to occur.
The losses caused by failure to fulfil the business security objective can be calculated by multiplying the expected number of successful attacks (in this case 52) with the average loss per single attack. This last number can be expressed in monetary terms, but given the context of CC, losses are likely expressed in different terms. The losses of all other target attacks can be calculated in the same way, based on which the value of security investments can be determined. This gives:

$$LossExpectancy = NrTargetAttacks \cdot SingleLossExpectancy$$

and

$$SecurityInvestmentValue = -LossExpectancy$$

8 EVALUATION OF CN METAMODEL

This chapter describes the last main design activity of this study in which the utility value of the CN metamodel is assessed. More specifically, this chapter presents the method and results of an evaluation that assesses the extent to which the suggested CN metamodel, modelling pattern and viewpoints can be used to model relevant system and security properties of CNs such that their vulnerabilities can be uncovered and attacks assessed.

8.1 Evaluation Methodology

This activity is partially combined with activity 6 of the DSRM, communication. For both evaluation and initial communication purposes, the outcomes of chapters 1 to 7 of this study are shared with relevant stakeholders of Thales Nederland by means of a draft written report and a live presentation. These stakeholders are asked to evaluate the main design artefacts of this study in relation to their intended utility value. Due to time constraints, the evaluation is conducted by means of written surveys even though interviews or a case study would likely result in more qualitative and comprehensive data. The surveys with the questionnaire are provided on paper to Thales employees.
For the questionnaire, as presented in Appendix B, the Unified Theory of Acceptance and Use of Technology (UTAUT) of Venkatesh, Morris, *et al.* [64] is adopted as a reference framework. This framework is a synthesis of eight models, theorising that individual reactions and intentions towards using information technology can explain the user acceptance thereof. As presented in Figure 26, the UTAUT framework consists of four core determinants and four moderators of key relationships.
Performance Expectancy is defined as ”the degree to which an individual believes using the system will help him or her to attain gains in job performance”. Effort Expectancy is defined as ”the degree of ease associated with the use of the system”. In this case, *the system* refers to the CN metamodel, modelling pattern and viewpoints; *job performance* refers to modelling relevant system and security properties of CNs such that their vulnerabilities can be uncovered and attacks assessed.
The determinant Social Influence is not considered in this evaluation as ”the degree to which an individual perceives that important others believe he or she should use the new system” cannot be properly surveyed since the CN metamodel, modelling pattern and viewpoints are merely proof of concepts - and are not yet realised in practice. For the same reason, the determinant Facilitating Conditions, i.e. ”the degree to which an individual believes that an organisation and technical infrastructure exists to support use of the system”, is also not considered.

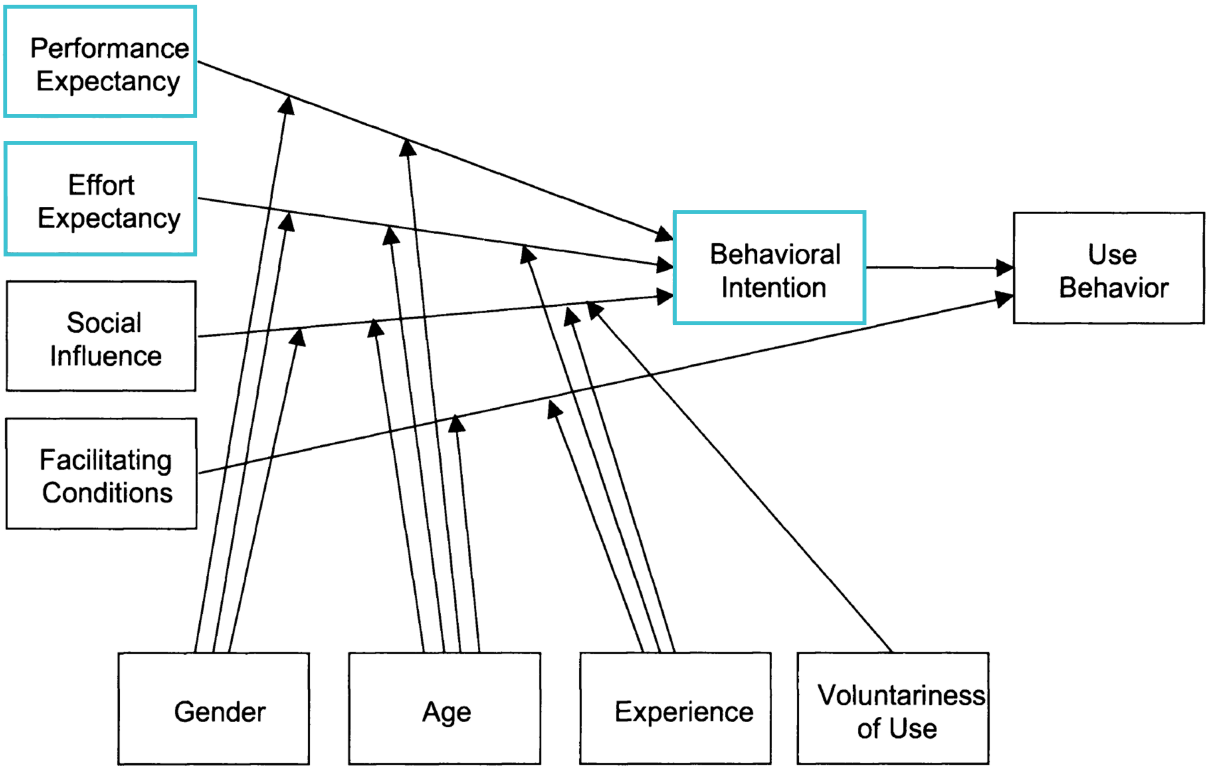


Figure 26: Unified Theory of Acceptance and Use of Technology Model

The core determinants (i.e. independent variables) Performance Expectancy and Effort Expectancy influence the mediating variable Behavioural Intention. The latter can be defined as ”the degree to which a person has formulated conscious plans to perform or not perform some specified future behaviour” [65]. In this case, this refers to using the CN metamodel, modelling pattern and viewpoints.
While questions related to Performance Expectancy and Effort Expectancy are directly used for the evaluation, questions related to Behavioural Intention are used as control. Questions related to the mediating variables are not included as they are not relevant for evaluating the design artefacts of this study. The selected variables are marked light blue in Figure 26.
The independent and mediating variables are operationalised in the questionnaire using the statements [64] suggested as most adequately representing the conceptual underpinnings of the variables. The specific statements that are adapted for this evaluation are presented in Table 9. Whenever possible, for the Performance Expectancy (P1-P3) and Effort Expectancy (E1-E4), a distinction is made between using the design artefacts to model relevant system and security properties (_a), and using the resulting architecture descriptions to uncover vulnerabilities and assess attacks (_b). This is to survey not only the direct practical utility value but also the indirect goal-oriented utility value of the design artefacts. The statement ”If I use the system, I will increase my chances of getting a raise.” is excluded as the author deems this statement inappropriate. For the Behavioural Intention (B1-B3), the distinction between use cases is not made as the use cases only exist as a whole. For all statements, respondents are asked to indicate their level of agreement or disagreement on a symmetric 5-point Likert scale. After each category of statements, they are asked to elaborate on the given scores.

PERFORMANCE EXPECTANCY	
P1a	The CN metamodel, modelling pattern and viewpoints would be *useful to me* when modelling relevant system and security properties of CN.
P1b	The resulting CN architecture descriptions would be *useful to me* when uncovering vulnerabilities and assessing attacks of CN
P2a	The CN metamodel, modelling pattern and viewpoints would *accelerate my job* of modelling relevant system and security properties of CN.
P2b	The resulting CN architecture descriptions would *accelerate my job* of uncovering vulnerabilities and assessing attacks of CN
P3a	The CN metamodel, modelling pattern and viewpoints would *increase my productivity* when modelling relevant system and security properties of CN.
P3b	The resulting CN architecture descriptions would *increase my productivity* when uncovering vulnerabilities and assessing attacks of CN.
EFFORT EXPECTANCY	
E1	The CN metamodel, modelling pattern and viewpoints are *clear and understandable*.
E2a	It would be easy for me to *learn using* the CN metamodel, modelling pattern and viewpoints for modelling relevant system and security properties of CN.
E2b	It would be easy for me to *learn using* the resulting CN architecture descriptions for uncovering vulnerabilities and assessing attacks of CN.
E3a	It would be easy for me to *use* the CN metamodel, modelling pattern and viewpoints for modelling relevant system and security properties of CN.
E3b	It would be easy for me to *use* the resulting CN architecture descriptions for uncovering vulnerabilities and assessing attacks of CN.
E4a	It would be easy for me to *become skillful* with the CN metamodel, modelling pattern and viewpoints for modelling relevant system and security properties of CN.
E4b	It would be easy for me to *become skillful* with the resulting CN architecture descriptions for uncovering vulnerabilities and assessing attacks of CN.
BEHAVIOURAL INTENTION	
B1	Given that all facilitating conditions are met, I *intend* to use the CN metamodel, viewpoints and modelling pattern in the next 12 months.
B2	Given that all facilitating conditions are met, I *predict* I would use the CN metamodel, viewpoints and modelling pattern in the next 12 months.
B3	Given that all facilitating conditions are met, I *plan* to use the CN metamodel, viewpoints and modelling pattern in the next 12 months.

Table 9: CN Metamodel Evaluation - Adapted UTAUT Survey Questions

8.2 Evaluation Analysis

In total, nine stakeholders within Thales Nederland participated in this evaluation, assessing the direct and indirect utility value of the proposed design artefacts. Of the nine participants, one person indicated that his or her daily job does not really involve modelling or using (CN) security architectures. Hence, all answers of this participant are excluded from this evaluation analysis. Of the eight other participants, some indicated that they have little to no experience in either modelling or using (CN) security architectures. These participants are asked to only answer the questions that are relevant to them.

Table 10 provides for each question the number of times a certain agreement level is chosen as well as the total number of responses. Based on this table, Figure 27 is created which provides a graphical summary by means of a diverging stacked bar chart. For the latter hold that the ordinal levels 1, 2 and 3 are considered negative to neutral while levels 4 and 5 are considered positive.
Based on the results of this small-scale evaluation, it seems that the participants generally believe that the CN metamodel, modelling pattern and viewpoints will help them to attain gains in job performance. This holds for both using the design artefacts to model relevant system and security properties as well as using the resulting architecture descriptions to uncover vulnerabilities and assess attacks. The two extreme outliers are given by the same participant, indicating that he or she believes that the design artefacts will likely not accelerate modelling speed (P2a) but will increase modelling productivity (P3a). As a comment, this participant suggested including a step-by-step wizard for analysis purposes. This suggests that this particular participant believes that the design artefacts will likely result in proper architecture descriptions but at the cost of time. Looking at the comments of the other participants, it is noticeable that the focus is mainly on the indirect performance expectancy. They mentioned that the design artefacts are ”a good way to document and reason about security attacks/vulnerabilities”, for ”performing risk analyses” and ”contributing to security and discussions”.
With regards to the effort expectancy, it seems the participants are generally neutral to positive when it comes to learn using, using and becoming skilful with the design artefacts and their resulting architecture descriptions. Even though ”the use of this model requires some level of training”, the ”high abstraction [makes it] easy to perform”. Nevertheless, several participants noted important boundary conditions. From a modelling perspective, this includes ”Capella [that] has the tendency to be more difficult in multi user situations, [while] getting readable output is not if best feasible”. From a model usage perspective, a participant mentioned that the ”output is very dependent on [the] engineer”.
When asking about the intentions of the participants, it is noticeable that only a few of them answered these questions. From those who answered these questions, it seems that the participants are generally not intending, predicting or planning to use the CN metamodel, modelling pattern and viewpoints. This is likely because the artefacts are ”still very theoretical”, lacking ”details [of] how to do this”.

	1	2	3	4	5	#Responses
PERFORMANCE EXPECTANCY						
P1a	0	0	1	6	0	7
P1b	0	0	3	5	0	8
P2a	1	0	2	4	0	7
P2b	0	0	3	5	0	8
P3a	0	0	4	2	1	7
P3b	0	1	2	5	0	8
EFFORT EXPECTANCY						
E1	0	0	3	4	0	7
E2a	0	2	3	0	1	6
E2b	0	0	4	1	1	6
E3a	0	1	3	1	1	6
E3b	0	1	2	4	0	7
E4a	0	0	4	2	0	6
E4b	0	0	4	1	2	7
BEHAVIOURAL INTENTION						
B1	1	2	1	1	0	5
B2	1	2	1	1	0	5
B3	1	1	2	1	0	5

Table 10: CN Metamodel Evaluation - Numeric Summary of Evaluation Results

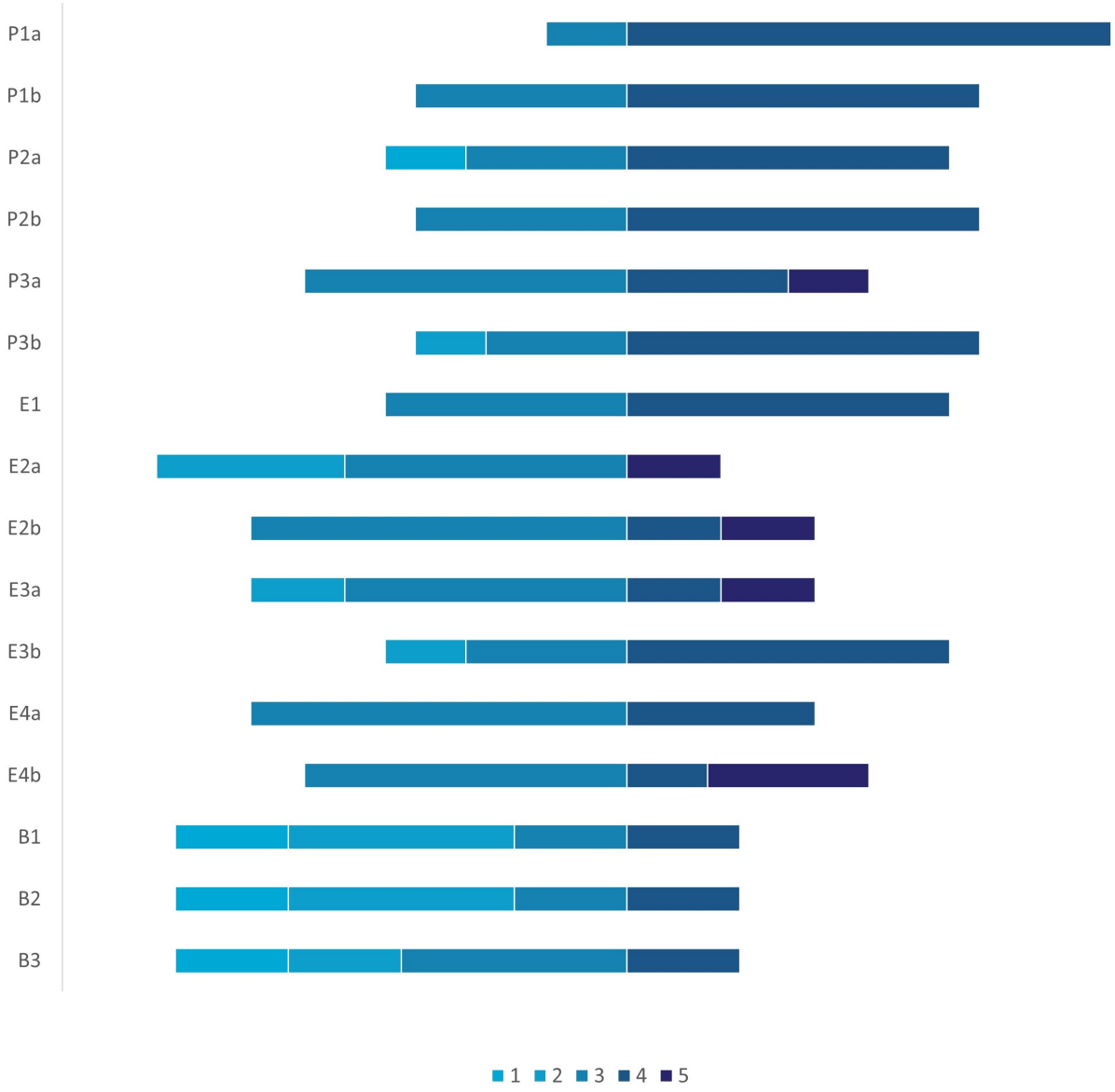


Figure 27: CN Metamodel Evaluation - Graphical Summary of Evaluation Results

9 CONCLUSION AND DISCUSSION

This design science study shows how security properties of Collaborative Networks (CNs) can be integrated within CN system architectures, realising CN security architectures for asserting system security and promoting security by design. More specifically, the Design Science Research Methodology of Peffers, Tuunanen, *et al.* [36] was adopted as a guiding framework for the creation of an initial generic metamodel, modelling pattern and viewpoints that can support the modelling of CN security architectures. Using a fictional but representative case on Collaborative Combat (CC) as a running example, an application of these design artefacts was illustrated.

In contrast to frameworks such as [19] and [62], where security architectures are separated from system architectures, this study attempted to model security properties within existing system architectures such that their specific vulnerabilities can be uncovered and attacks assessed. This means that the CN metamodel must consist of both generic system modelling elements as well as CN-specific security elements.
With regards to the generic system elements, existing constructs and relationships from the Eclipse Capella modelling tool [26] were adopted and generalised. This was based on architectural properties that were extracted from related work on systems that share similar characteristics with CNs (e.g. [9, 14]). This includes generally available constructs representing information systems, services and data. Specifically for CNs, it seems that that there is a need to introduce a more specialised modelling element: the Security Domain element. Even though it is unlikely that comparable constructs can be found in existing modelling tools, languages and architecture descriptions, it is likely that such an element can be relatively easily introduced as a specialisation of any generic grouping element.
Note that this study assumed that human entities are behaving benevolently - intentionally and unintentionally - meaning that such a construct and its associated relationships are not included in the metamodel. In future studies, human entities and their interactions can be incorporated within the metamodel, for instance, by adopting and adapting SecureUML [50]. Furthermore, this study assumed that all involved atomic systems can be considered secure and their security architectures as known. Hence, the suggested generic metamodel should be used in conjunction with existing security metamodels, for example [16, 20].
With regards to the specific security-related elements, these were derived from vulnerable system parts as induced from the case on CC and from a set of security properties as extracted from related work. This includes different types of entities (i.e. integrated or platform/mobile collaborative information systems), transport relationships (i.e. communication, calling and execution) and their required trust relationships (i.e. authenti-

cation, execution and competency) as well as a set of relevant security services (i.a. security policy management, privilege management and cryptographic support).
In this study, a simplified CN scenario was assumed in which direct trust exist between all participating entities (i.e. between all security domains). Even though this is likely the case for most CN scenarios, future studies can mitigate this limitation, for instance, by introducing a trust authority construct similar to the MobileTrust framework [10, 13] which can facilitate recommended or derived trust. Another limitation of the metamodel is the omission of security recovery considerations. This is because only operational security properties were considered in this initial design, in contrast to e.g. [6]. Since proper security response is imperative in time and data sensitive environments, additional research is necessary with regards to including these considerations. Finally, note that paradigms such as Security as a Service[47] were also excluded from this study. This is because such topics are more suitable to consider once the metamodel is more mature and practical experiences exist.

In the application phase of this study, the usage of the CN metamodel, modelling pattern and viewpoints was illustrated within the context of the Arcadia architecture development methodology [33], the Eclipse Capella modelling tool [26], its SysML-inspired modelling language [59], and the security analysis methodology as presented by Breu, Innerhofer-Oberperfler, *et al.* [55]. The CN vulnerabilities that were deduced from attacks as extracted from related work, applied on the case on CC, established a generalisable problem instance.
It is advisable to incorporate security considerations as early as possible during architecture development. From an architecture perspective, this promotes security by design. From a stakeholder perspective, this promotes security awareness and facilitates discussions about security.
In logical system architecture development, architects should consider including at least (the most important) security services and their main interactions, allowing the functional modelling of security facilities in the broader context of system functionality. In physical system architecture development, architects should include both high-level security properties such as security domains as well as lower-level security properties such as security service mechanisms. While the former can serve as input for business stakeholders, the latter is more tailored towards technology stakeholders. The lower-level physical system architectures can be further enriched with security indicators such as those related to data functionality and communication security, realising detailed system security architectures based on which system security can be asserted. Using the system security architectures, both architects as well as security officers should be able to analyse the level of security for the systems under study. This allows not only the identification of weak system links but might also provide a basis for security investments. However, proper quantitative analyses require estimates or measures on the number of security breaches and their associated direct/indirect costs, which can be challenging. Furthermore, while analysing security, it is important to not only consider direct security risks but to also consider the propagating effect of attacks or threats. Especially in CNs, where heterogeneous and distributed entities have to collaborate while having limited environmental observability, it cannot be assumed that all collaborating entities are secure.

The design artefacts, along with this illustration, were presented to and evaluated with possible future end-users. This small-scale evaluation seems to indicate that it might take some effort to learn using, use and become skilful with the CN metamodel, modelling pattern and viewpoint for modelling relevant system security properties. Nevertheless, with further in-depth development, detailed elaboration and the right tooling, it seems that these design artefacts have the potential to attain gains in job performance - especially when it comes to using the resulting architecture descriptions to reason and discuss about security, and for performing security risk analyses.

BIBLIOGRAPHY

[1] U. S. Bititci, V. Martinez, P. Albores, and J. Parung, "Creating and managing value in collaborative networks," *International Journal of Physical Distribution & Logistics Management*, 2004.

[2] L. M. Camarinha-Matos and H. Afsarmanesh, "Collaborative networks: A new scientific discipline," *Journal of intelligent manufacturing*, vol. 16, no. 4, pp. 439–452, 2005.

[3] *Airborne superiority through collaborative, connected combat*, May 2019. [Online]. Available: https://www.thalesgroup.com/en/worldwide-defence/air-forces/magazine/airborne-superiority-through-collaborative-connected-combat.

[4] L. Prusak, "Building a collaborative enterprise," *Harvard Business Review*, vol. 89, no. 7-8, pp. 94–101, 2011.

[5] J. Eloff and M. Eloff, "Information security architecture," *Computer Fraud & Security*, vol. 2005, no. 11, pp. 10–16, 2005.

[6] "Security in open systems - a security framework," European Computer Manufacturers Association, Geneva, CH, Report, Jul. 1988.

[7] J. P. Kruys, "Security of open systems," *Computers and Security*, vol. 8, no. 2, pp. 139–147, 1989.

[8] W. A. Jansen, "Countermeasures for mobile agent security," *Computer communications*, vol. 23, no. 17, pp. 1667–1676, 2000.

[9] Y. Demchenko, "Virtual organisations in computer grids and identity management," *Information Security Technical Report*, vol. 9, no. 1, pp. 59–76, 2004.

[10] C. Lin and V. Varadharajan, "Mobiletrust: A trust enhanced security architecture for mobile agent systems," *International Journal of Information Security*, vol. 9, no. 3, pp. 153–178, 2010.

[11] Y. Demchenko, C. De Laat, O. Koeroo, and D. Groep, "Re-thinking grid security architecture," in *4th IEEE International Conference on eScience, eScience 2008*, pp. 79–86.

[12] P. J. Broadfoot and A. P. Martin, “A critical survey of grid security requirements and technologies,” *Programming Research Group, PRGRR-03-15, Oxford University Computing Laboratory*, 2003.

[13] C. Lin, V. Varadharajan, W. Yan, and V. Pruthi, “Security and trust management in mobile agents: A new perspective,” in *2005 2nd Asia Pacific Conference on Mobile Technology, Applications and Systems*, pp. 1–9.

[14] H. Reiser and G. Vogt, “Security requirements for management systems using mobile agents,” *ISCC 2000 - 5th IEEE Symposium on Computers and Communications*, pp. 160–165, 2000.

[15] R. Cole, “A model for security in distributed systems,” *Computers and Security*, vol. 9, no. 4, pp. 319–330, 1990.

[16] “Security in open systems - data elements and service definitions,” European Computer Manufacturers Association, Geneva, CH, Standard, Dec. 1989.

[17] N. M. Karnik and A. R. Tripathi, “Security in the ajanta mobile agent system,” *Software: Practice and Experience*, vol. 31, no. 4, pp. 301–329, 2001.

[18] J. L. Vivas, J. Lopez, and J. A. Montenegro, “Grid security architecture: Requirements, fundamentals, standards, and models,” *Security in distributed, Grid, mobile, and pervasive computing*, pp. 255–288, 2007.

[19] D. G. Rosado, E. Fernandez-Medina, J. Lopez, and M. Piattini, “Systematic design of secure mobile grid systems,” *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1168–1183, 2011.

[20] I. Foster, H. Kishimoto, A. Savva, D. Berry, A. Djaoui, A. Grimshaw, B. Horn, F. Maciel, F. Siebenlist, and R. Subramaniam, “The open grid services architecture,” in *Global Grid Forum, GFD-I*, vol. 30.

[21] Y. Demchenko, L. Gommans, C. De Laat, B. Oudenaarde, A. Tokmakoff, M. Snijders, and R. Van Buuren, “Security architecture for open collaborative environment,” in *European Grid Conference on Advances in Grid Computing - EGC 2005*, vol. 3470, Springer Verlag, pp. 589–599.

[22] V. Varadharajan, “Security enhanced mobile agents,” in *Proceedings of the 7th ACM conference on Computer and Communications Security*, 2000, pp. 200–209.

[23] H. K. Tan and L. Moreau, “Trust relationships in a mobile agent system,” in *International Conference on Mobile Agents*, Springer, 2001, pp. 15–30.

[24] J. El Hachem, Z. Y. Pang, V. Chiprianov, A. Babar, and P. Aniorte, “Model driven software security architecture of systems-of-systems,” in *23rd Asia-Pacific Software Engineering Conference, APSEC 2016*, A. Potanin, G. C. Murphy, S. Reeves, and J. Dietrich, Eds., vol. 0, IEEE Computer Society, pp. 89–96.

[25] J. Blangenois, G. Guemkam, C. Feltus, and D. Khadraoui, “Organizational security architecture for critical infrastructure,” in *2013 International Conference on Availability, Reliability and Security*, pp. 316–323.

[26] *Model based systems engineering | capella MBSE tool*. [Online]. Available: https://www.eclipse.org/capella/ (visited on 05/11/2021).

[27] *Capella cybersecurity viewpoint*. [Online]. Available: https://github.com/eclipse/capella-cybersecurity/blob/master/plugins/org.polarsys.capella.cybersecurity.doc/html/usermanual.mediawiki (visited on 05/11/2021).

[28] *Capella MBSE tool - what is MBSE*. [Online]. Available: https://www.eclipse.org/capella/what_is_mbse.html (visited on 05/11/2021).

[29] *Capella MBSE tool - adopters*, May 2019. [Online]. Available: https://www.eclipse.org/capella/adopters.html (visited on 05/10/2021).

[30] “OMG system modeling language,” Object Management Group, Standard, Dec. 2019.

[31] “OMG unified modeling language,” Object Management Group, Standard, Dec. 2017.

[32] *Capella MBSE tool - features*. [Online]. Available: https://www.eclipse.org/capella/features.html (visited on 05/11/2021).

[33] *Capella MBSE tool - arcadia*. [Online]. Available: https://www.eclipse.org/capella/arcadia.html (visited on 05/11/2021).

[34] “The TOGAF® standard, version 9.2,” The Open Group, Standard, Mar. 2018.

[35] “NATO architecture framework, version 4,” NATO Consultation, Command and Control Board, Standard, Sep. 2020.

[36] K. Peffers, T. Tuunanen, M. A. Rothenberger, and S. Chatterjee, “A design science research methodology for information systems research,” *Journal of management information systems*, vol. 24, no. 3, pp. 45–77, 2007.

[37] J. Kantert, L. Klejnowski, S. Edenhofer, S. Tomforde, and C. Müller-Schloer, “A threatmodel for trust-based systems consisting of open, heterogeneous and distributed agents.,” in *ICAART (1)*, 2016, pp. 173–180.

[38] A. Iranmehr, A. Iranmehr, and M. Sharifnia, “Message-based security model for grid services,” in *2009 International Conference on Computer and Electrical Engineering, ICCEE 2009*, vol. 2, pp. 511–515.

[39] S. Naqvi and M. Riguidel, “Threat model for grid security services,” in *European Grid Conference on Advances in Grid Computing - EGC 2005*, vol. 3470, Springer Verlag, pp. 1048–1055.

[40] H. Jameel, U. Kalim, A. Sajjad, S. Lee, and T. Jeon, “Mobile-to-grid middleware: Bridging the gap between mobile and grid environments,” in *European Grid Conference*, Springer, 2005, pp. 932–941.

[41] S. Hernan, S. Lambert, T. Ostwald, and A. Shostack, “Uncover security design flaws using the STRIDE approach,” *MSDN Magazine*, Nov. 2006. [Online]. Available: https://docs.microsoft.com/en-us/archive/msdn-magazine/2006/november/uncover-security-design-flaws-using-the-stride-approach.

[42] L. O. Nweke, “Using the CIA and AAA models to explain cybersecurity activities,” *PM World Journal*, vol. 6, 2017.

[43] “Chapter 1 - general security concepts: Access control, authentication, and auditing,” in *How to Cheat at Securing Your Network*, ser. How to Cheat, I. Dubrawsky, Ed., Burlington: Syngress, 2007, pp. 1–33.

[44] “ISO/IEC/IEEE 42010:2011 systems and software engineering — architecture description,” International Organization for Standardization, Geneva, CH, Standard, Dec. 2011.

[45] “Information technology - open systems interconnection - basic reference model: The basic model,” International Organization for Standardization, Geneva, CH, Standard, Nov. 1994.

[46] “Security architecture for open systems interconnection for CCITT applications,” ITU+T (CCITT), Geneva, CH, Recommendation, 1991.

[47] M. Hafner, M. Memon, and R. Breu, “SeAAS - a reference architecture for security services in SOA,” *Journal of Universal Computer Science*, vol. 15, no. 15, pp. 2916–2936, 2009.

[48] “Integrating risk and security within a TOGAF® enterprise architecture,” The Open Group, Guide, Apr. 2019.

[49] M. A. Neri, M. Guarnieri, E. Magri, S. Mutti, and S. Paraboschi, “A model-driven approach for securing software architectures,” in *2013 International Conference on Security and Cryptography (SECRYPT)*, pp. 1–8.

[50] T. Lodderstedt, D. Basin, and J. Doser, "SecureUML: A UML-based modeling language for model-driven security," in *International Conference on the Unified Modeling Language*, Springer, 2002, pp. 426–441.

[51] *MITRE ATT&CK® enterprise matrix*, Apr. 2021. [Online]. Available: https://attack.mitre.org/matrices/enterprise/.

[52] B. Strom, *ATT&CK 101*, Sep. 2018. [Online]. Available: https://medium.com/mitre-attack/att-ck-101-17074d3bc62.

[53] A. Grusho, N. Grusho, M. Levykin, and E. Timonina, "Analysis of information security of distributed information systems," in *9th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops, ICUMT 2017*, vol. 2017-November, IEEE Computer Society, pp. 96–100.

[54] Z. Jiajia, "An object oriented dynamic security reference model for virtual enterprises," in *2016 Chinese Control and Decision Conference (CCDC)*, pp. 6800–6803.

[55] R. Breu, F. Innerhofer-Oberperfler, and A. Yautsiukhin, "Quantitative assessment of enterprise security system," in *2008 Third International Conference on Availability, Reliability and Security*, IEEE, 2008, pp. 921–928.

[56] J. Jürjens, "UMLsec: Extending UML for secure systems development," in *International Conference on The Unified Modeling Language*, Springer, 2002, pp. 412–425.

[57] L. Apvrille and Y. Roudier, "Towards the model-driven engineering of secure yet safe embedded systems," *Electronic Proceedings in Theoretical Computer Science*, vol. 148, pp. 15–30, Apr. 2014.

[58] J. F. Ruiz, R. Harjani, A. Maña, V. Desnitsky, I. Kotenko, and A. Chechulin, "A methodology for the analysis and modeling of security threats and attacks for systems of embedded components," in *20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing, PDP 2012*, pp. 261–268.

[59] *Is capella a sysml tool?* [Online]. Available: https://www.eclipse.org/capella/arcadia_capella_sysml_tool.html (visited on 05/11/2021).

[60] "Archimate® 3.1 specification," The Open Group Standard, Standard, Nov. 2019.

[61] I. Band, W. Engelsman, C. Feltus, S. González Paredes, J. Hietala, H. Jonkers, and S. Massart, "Modeling enterprise risk management and security with the archimate® language," The Open Group, White Paper, Jan. 2015.

[62] J. J. Brennan, M. Rudell, D. Faatz, and C. Zimmerman, “Visualizing enterprise-wide security (VIEWS),” in *20th Annual Computer Security Applications Conference, ACSAC 2004*, pp. 71–79.

[63] J. S. Burkett, “Business security architecture: Weaving information security into your organization’s enterprise architecture through SABSA®,” *Information Security Journal: A Global Perspective*, vol. 21, no. 1, pp. 47–54, 2012.

[64] V. Venkatesh, M. G. Morris, G. B. Davis, and F. D. Davis, “User acceptance of information technology: Toward a unified view,” *MIS quarterly*, pp. 425–478, 2003.

[65] P. R. Warshaw and F. D. Davis, “Disentangling behavioral intention and behavioral expectation,” *Journal of experimental social psychology*, vol. 21, no. 3, pp. 213–228, 1985.

A DEFINITIONS OF METAMODEL ELEMENTS

MODELLING ELEMENT	DEFINITION
Collaborative Network	An adaptive and scalable ecosystem of IISs and CISs that collaborate based on agreed on principles and interoperable infrastructures with the purpose of achieving some common goals.
Attack	An intentional attempt to compromise the confidentiality, integrity or availability of a Threatened Asset.
Vulnerability	A characteristic or weakness that renders a Vulnerable System Part open to exploitation by an Attack.
(Data Association)	A modelling construct which represents the presence of Security or System Data in Communication, Calling or Execution Relationships.
Security Domain	A group of security-controlled entities - administered and controlled by an organisation that has certain security policies and mechanisms in place - propagating trust, privilege and security knowledge.
Vulnerable System Part	A system part of which its vulnerabilities can be exploited by attacks with the purpose of threatening a Threatened Asset.
Integrated Information System (IIS)	An atomic entity that is embedded on or is closely related to a physical system and interacts with intra-domain IISs or CISs.
Collaborative Information System (CIS)	A platform or mobile entity that interacts with intra-domain IISs and other intra/inter-domain CISs. CISs have distributed, heterogeneous, autonomous and asynchronous characteristics.
Communication Relationship	A IIS–CIS or CIS–CIS relationship that transports messages (without directly interfering local resources).
Calling Relationship	A IIS–CIS or CIS–CIS relationship that transports tasks between IISs/CISs without directly interfering local resources.
Execution Relationship	A IIS–CIS relationship that transports tasks between IISs/CISs that directly manipulate local resources.
Authentication Trust	In CIS–CIS relationships, the belief in the authenticity of the keys held by the CISs.
Execution Trust	In inter-domain CIS–CIS relationships, the belief that the CISs will faithfully and without any tampering execute the given tasks or forward the given message.
Competency Trust	In inter-domain CIS–CIS Calling relationships, the belief that the CISs are competent in executing the given tasks.

Security Service	A decoupled and composable software unit which runs on IISs or CISs, implementing one or more security mechanisms.
Authentication and Security Attribution	A local Security Service that verifies credentials of CISs and provides identity information based on which privileges and other security attributes are returned that can be validated by all collaborating Security Domains.
Privilege Management	A local Security Service that manages and enforces CIS privileges as well as the temporary delegations thereof.
Authorisation and Access Control	A domain-wide Security Service that uses security attributes to make access control decisions based on relevant security policies.
Security Policy Management	A domain-wide Security Service that manages changing security policies.
Secure Association	A local Security Service that creates associations between CIS, verifies authorisations and applies transport security.
Interdomain	A local Security Service that establishes sessions between Security Domains, and controls the exchange of security and policy information.
Accountability	A local Security Service that records all security operations over time (i.e. audition), creating a permanent proof of actions (i.e. non-repudiation).
Cryptographic Support	A local Security Service that protects System and Security Data using cryptographic algorithms.
Security Data	Information related to security mechanisms that is produced, used, manipulated or stored by a System or Security Service.
Threatened Asset	A system part that is valuable to stakeholders and that needs to be protected.
System Service	A decoupled and composable software unit which runs on IISs or CISs, implementing core system functionality to fulfil user needs.
System Data	Information related to core system functionality that is produced, used, manipulated or stored by a System or Security Service.

B EVALUATION SURVEY

See next pages.

Modelling Security Architectures of Collaborative Networks

Dear reader,

You have have attended a presentation and/or read a preliminary report about the modelling of CN security architectures. As part of this study, you are invited to participate in a short evaluation survey.

This survey is divided into 3 sections, presenting 16 statements and 3 open questions in total. For each statement, please indicate your level of agreement based on your expectations or intentions. At the end of each section, you are asked to elaborate on the given answers.

No personal data is collected but note that all answers to the open questions will be included in the final study report and made publicly available. If you have any questions, feel free to contact me at t.man@student.utwente.nl.

Kind regards,
Tjo-Kin Man

*Vereist

Performance Expectancy

1. The CN metamodel, modelling pattern and viewpoints would be useful to me when modelling relevant system and security properties of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

2. The resulting CN architecture descriptions would be useful to me when uncovering vulnerabilities and assessing attacks of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

3. The CN metamodel, modelling pattern and viewpoints would accelerate my job of modelling relevant system and security properties of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

4. The resulting CN architecture descriptions would accelerate my job of uncovering vulnerabilities and assessing attacks of CNs *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

5. The CN metamodel, modelling pattern and viewpoints would increase my productivity when modelling relevant system and security properties of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

6. The resulting CN architecture descriptions would increase my productivity when uncovering vulnerabilities and assessing attacks of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

7. Please elaborate on the given answers *

Effort Expectancy

8. The CN metamodel, modelling pattern and viewpoints are clear and understandable. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

9. It would be easy for me to learn using the CN metamodel, modelling pattern and viewpoints for modelling relevant system and security properties of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

10. It would be easy for me to learn using the resulting CN architecture descriptions for uncovering vulnerabilities and assessing attacks of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

11. It would be easy for me to use the CN metamodel, modelling pattern and viewpoints for modelling relevant system and security properties of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

12. It would be easy for me to use the resulting CN architecture descriptions for uncovering vulnerabilities and assessing attacks of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

13. It would be easy for me to become skillful with the CN metamodel, modelling pattern and viewpoints for modelling relevant system and security properties of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

14. It would be easy for me to become skillful with the resulting CN architecture descriptions for uncovering vulnerabilities and assessing attacks of CNs. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

15. Please elaborate on the given answers *

Intentions

16. Given that all facilitating conditions are met, I intend to use the CN metamodel, viewpoints and modelling pattern in the next 12 months. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

17. Given that all facilitating conditions are met, I predict I would use the CN metamodel, viewpoints and modelling pattern in the next 12 months. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

18. Given that all facilitating conditions are met, I plan to use the CN metamodel, viewpoints and modelling pattern in the next 12 months. *

Markeer slechts één ovaal.

	1	2	3	4	5	
Strongly disagree	◯	◯	◯	◯	◯	Strongly agree

19. Please elaborate on the given answers *