Democratic and Popular Republic of Algeria

Ministry of Higher Education and Scientific Research

Abbes Laghrour University -Khenchela-

Faculty of Sciences and Technologies

Department of Mathematics and Computer Science

Order number:
Serial number:

Contribution to the Specification and Formal Analysis of Cyber-Physical Systems: Application to Industry 4.0

Thesis presented by

Ayoub BOUHEROUM

To obtain the degree of

DOCTOR IN COMPUTER SCIENCE

(Option: Software Engineering and Distributed Systems)

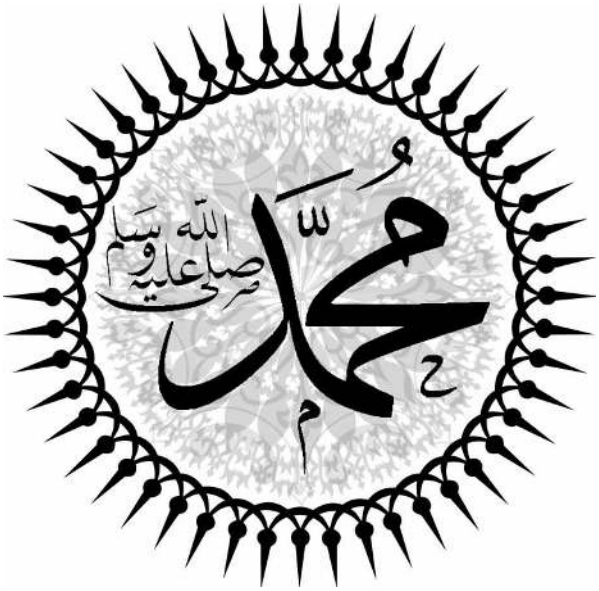
Presented publicly on: 25/05/2025

Composition of the jury:

N°	First & Family name	Grade	Establishment	Quality
01	Zianou AHMED SEGHIR	PROF	Abbes Laghrour University, Khenchela	President
02	Djamel BENMERZOUG	PROF	University Constantine 2 - Abdelhamid Mehri	Supervisor
03	Sofiane Mounine HEMAM	PROF	Abbes Laghrour University, Khenchela	Co-Supervisor
04	Rohallah BENABOUD	MCA	Larbi Ben M'hidi University, Oum El Bouaghi	Examiner
05	Toufik MARIR	MCA	Larbi Ben M'hidi University, Oum El Bouaghi	Examiner
06	Ouided HIOUAL	MCA	Abbes Laghrour University, Khenchela	Examiner

ICOSI (Knowledge engineering and computer security) Laboratory


بسم الله الرحمن الرحيم


محمد
صلى الله عليه وسلم

Abstract

Abstract:

Cyber-Physical Systems (CPS) operate across different spatial and temporal scales and exhibit complex, context-dependent behaviors. The intricate nature of CPS poses significant conceptual and technical challenges, as understanding and managing such systems often exceed individual capacities. To address these challenges, advancements in modeling languages, formal methods, and tools are essential, alongside interdisciplinary collaboration among domain experts, formal methods researchers, and tool developers.

This thesis proposes a multi-phase and iterative approach for designing, defining, and analyzing the dynamic and secure behavior of CPS, addressing the gap between theoretical formal methods and their practical application in CPS development.

First, adhering to the principles of ISO/IEC/IEEE 42010:2021 for architectural descriptions, we develop a metamodel that facilitates effective communication among stakeholders by employing architectural views and viewpoints. This approach ensures consistency and fosters a shared understanding of the system architecture.

Second, to address the limitations of existing formalisms, we introduce CA-BRS, a novel model that combines Bigraphical Reactive Systems (BRS) and Control Agents. CA-BRS distinguishes between the virtual, physical, and cyber levels of CPS, using abstract agents and bigraphs to specify these dimensions. It also incorporates Controlled Reaction Rules to represent both physical and cyber evolutions while considering material constraints. To analyze CPS behavior, we define a formal computational model, the Guided Transition System (GTS), which captures and evaluates emergent properties such as security and safety.

Third, in the design phase of CPS, we establish mapping rules to define

the behavioral semantics of CA-BRS using BPMN activity diagrams. This enables the detection of functional inconsistencies, such as deadlocks, infinite loops, or multiple terminations, during model execution. Additionally, we extend CA-BRS to address security requirements, ensuring data confidentiality and integrity by preventing unauthorized access and modifications in distributed CPS.

Finally, we demonstrate the practicality of our approach through a case study on Medical-CPS and Industry 4.0 (I4.0-CPS), focusing on network routing (Access Control Lists) and data confidentiality in Electronic Health Records. This highlights the balance between theoretical insights and practical considerations in addressing the physical, cyber, and safety dimensions of CPS.

Keywords: CPS, Medical CPS, Formal models, State space, BRS, Security, BPMN.

Résumé:

Les systèmes cyberphysiques (CPS) fonctionnent à différentes échelles spatiales et temporelles et présentent des comportements complexes et dépendants du contexte. La nature complexe des CPS pose d'importants défis conceptuels et techniques, rendant leur compréhension et leur gestion souvent hors de portée d'un seul individu. Pour relever ces défis, des avancées en langages de modélisation, en méthodes formelles et en outils sont essentielles, ainsi qu'une collaboration interdisciplinaire entre experts métiers, chercheurs en méthodes formelles et développeurs d'outils.

Cette thèse propose une approche itérative et multi-phases pour concevoir, définir et analyser le comportement dynamique et sécurisé des CPS, en comblant le fossé entre les méthodes formelles théoriques et leur application pratique dans le développement des CPS.

Premièrement, conformément aux principes de la norme ISO/IEC/IEEE 42010:2021 pour les descriptions architecturales, nous développons un méta-modèle qui facilite une communication efficace entre les parties prenantes grâce à l'utilisation de vues et de points de vue architecturaux. Cette approche garantit la cohérence et favorise une compréhension partagée de l'architecture du système.

Deuxièmement, pour pallier les limites des formalismes existants, nous introduisons CA-BRS, un modèle innovant qui combine les Systèmes Réactifs Bigraphiques (BRS) et les Agents de Contrôle. CA-BRS distingue les niveaux virtuel, physique et cyber des CPS, en utilisant des agents abstraits et des bigraphes pour spécifier ces dimensions. Il intègre également des Règles de Réaction Contrôlées pour représenter les évolutions physiques et cyber tout en tenant compte des contraintes matérielles. Pour analyser le comportement des CPS, nous définissons un modèle computationnel formel, le Système de Transition Guidé (GTS), qui capture et évalue les propriétés émergentes telles que la sécurité et la sûreté.

Troisièmement, dans la phase de conception des CPS, nous établissons des règles de correspondance permettant de définir la sémantique comportementale de CA-BRS à l'aide de diagrammes d'activité BPMN. Cela permet de détecter les incohérences fonctionnelles, telles que les blocages, les boucles infinies ou les terminaisons multiples, lors de l'exécution du modèle. De plus, nous étendons CA-BRS pour répondre aux exigences de sécurité, en garantissant la confidentialité et l'intégrité

des données en empêchant les accès non autorisés et les modifications non approuvées dans les CPS distribués.
Enfin, nous démontrons la praticité de notre approche à travers une étude de cas sur les Medical-CPS et sur la quatrième révolution industrielle (Industry 4.0 - I4.0-CPS), en mettant l'accent sur le routage réseau (listes de contrôle d'accès) et la confidentialité des données dans les dossiers de santé électroniques. Cela met en évidence l'équilibre entre les aspects théoriques et pratiques nécessaires pour aborder les dimensions physique, cyber et sécuritaire des CPS.

Mots-clés: CPS, Medical CPS, Formal models, State space, BRS, Security, BPMN.

خلاصة:

تعمل الأنظمة السيبرانية الفيزيائية (CPS) عبر مقاييس مكانية وزمانية مختلفة وتظهر سلوكيات معقدة تعتمد على السياق. تفرض الطبيعة المعقدة للأنظمة السيبرانية الفيزيائية تحديات مفاهيمية وفنية كبيرة، حيث غالبا ما يتجاوز فهم وإدارة مثل هذه الأنظمة القدرات الفردية. لمعالجة هذه التحديات، فإن التقدم في لغات النمذجة والأساليب الرسمية والأدوات أمر ضروري، إلى جانب التعاون متعدد التخصصات بين خبراء المجال وباحثي الأساليب الرسمية ومطوري الأدوات.

تقترح هذه الأطروحة نهجاً متعدد المراحل وتكرارياً لتصميم وتحديد وتحليل السلوك الديناميكي والآمن للأنظمة السيبرانية الفيزيائية، ومعالجة الفجوة بين الأساليب الرسمية النظرية وتطبيقها العملي في تطوير الأنظمة السيبرانية الفيزيائية.

أولاً، بالالتزام بمبادئ ISO/IEC/IEEE 42010:2021 للأوصاف المعمارية، نقوم بتطوير نموذج ميتا يسهل التواصل الفعال بين أصحاب المصلحة من خلال استخدام وجهات النظر المعمارية ووجهات النظر. يضمن هذا النهج الاتساق ويعزز الفهم المشترك لهندسة النظام.

ثانياً، لمعالجة القيود المفروضة على الصياغات الحالية، نقدم CA-BRS، وهو نموذج جديد يجمع بين الأنظمة التفاعلية ثنائية السطور (BRS) ووكلاء التحكم. يميز CA-BRS بين المستويات الافتراضية والمادية والسيبرانية لـ CPS، باستخدام وكلاء مجردين ومخططات ثنائية السطور لتحديد هذه الأبعاد. كما أنه يشتمل على قواعد رد الفعل المتحكم فيها لتمثيل التطورات المادية والسيبرانية مع مراعاة القيود المادية. لتحليل سلوك CPS، نقوم بتعريف نموذج حسابي رسمي، نظام الانتقال الموجه (GTS)، والذي يلتقط ويقيم الخصائص الناشئة مثل الأمن والسلامة.

ثالثاً، في مرحلة تصميم CPS، نقوم بإنشاء قواعد رسم الخرائط لتحديد الدلالات السلوكية لـ CA-BRS باستخدام مخططات نشاط BPMN. يتيح هذا اكتشاف التناقضات الوظيفية، مثل الجمود أو الحلقات اللانهائية أو الإنهاءات المتعددة، أثناء تنفيذ النموذج. بالإضافة إلى ذلك، نقوم بتوسيع CA-BRS لمعالجة متطلبات الأمان، وضمان سرية البيانات وسلامتها من خلال منع الوصول غير المصرح به والتعديلات في CPS الموزعة.

أخيرا، نوضح مدى جدوى نهجنا من خلال دراسة حالة حول نظام CPS الطبي وكذلك الثورة الصناعية الرابعة I4.0-CPS، مع التركيز على توجيه الشبكة (قوائم التحكم في الوصول) وسرية البيانات في السجلات الصحية الإلكترونية. وهذا يسلط الضوء على التوازن بين الرؤى النظرية والاعتبارات العملية في معالجة الأبعاد المادية والسيبرانية والسلامة لنظام CPS.

الكلمات المفتاحية: CPS، CPS الطبية، النماذج الرسمية، مساحة الحالة، BRS، الأمن، BPMN.

Contents

List of Figures

List of Tables

Chapter 1

General Introduction

Industry 4.0, also known as the fourth industrial revolution [10], represents a transformative paradigm shift in the manufacturing and industrial sectors [11]. It is characterized by the integration of digital technologies, advanced analytics, the Internet of Things (IoT), and Cyber-Physical Systems (CPS) into the fabric of industrial processes [12]. Industry 4.0 has a broad range of application domains across various industries, such as Smart Factories, Automotive Industry, Healthcare, Energy and Utilities, Agriculture, etc [13]. These application domains illustrate the versatility of Industry 4.0, showcasing how the integration of digital technologies can lead to improvements in efficiency, productivity, and innovation in diverse industries [14].

Although Industry 4.0 offers significant advantages, its adoption comes with challenges such as cyber security concerns, the need for up-skilling of the workforce, and the complexity of integrating diverse technologies [14, 15].

In particular, CPS representing the core of our current civilization embrace the principles of Industry 4.0, resulting in more efficient, agile, and intelligent industrial processes [16, 17]. CPSs are defined as systems that offer the integration of computation, networking, and physical processes [18]. Various examples dominate our daily lives and work, such as driverless cars that will soon master our roads, implanted medical devices that will improve and save many lives, and industrial control systems that control production and infrastructure [16, 19, 20].

Some of the defining characteristics of CPS include [21]:

- cyber capability in every physical component,
- high-degree of automation,

- networking at multiple scales,
- integration at multiple temporal and spatial scales and,
- reorganizing/reconfiguring dynamics

As CPS manipulate the real world, they constitute a danger for many applications. Therefore, their safety and security are essential properties of these indispensable systems. These quality properties of the system depend on the architecture of the underlying system [22].

In fact, much attention has been paid during system development and validation efforts. In fact, the architecture of a CPS addresses the unique challenges posed by the integration of software and hardware in real-time, safety-critical environments. It is designed to handle distributed, scalable, and secure systems while considering the specific requirements of the physical processes being controlled or monitored.

1.1 Problem Statement

Software engineering poses specific challenges based on the above characteristics of CPS in today's networked, interconnected world. It can have a profound impact in this domain by defining suitable modeling and specification notations as well as supporting design-time formal verification. In particular, the software architecture of CPS should include robust mechanisms for data encryption, access control, and protection against cyber-physical attacks. Therefore, the application of security-by-design principles during the development of CPS is essential to have an acceptable level of security assurance; Safety and Security modeling should be considered at the early stages of the development life-cycle. It must inform every phase of software development, from requirements engineering to design, implementation, testing, and deployment.

Obviously, software engineers use models early in the life cycle to improve the quality of artifacts such as requirements, but do not typically include security concerns; system requirements and design are done first, and security is added as an afterthought (There are exceptions, of course !).

Our work embraces this idea and investigates in interactions between software engineering and security engineering which give rise to several fascinating research challenges and opportunities. Clearly, there is much to be

gained by developing processes and tools to unify security policy development into the CPS development process, specifically by making use of CPS formal models when designing security property. These are the subject of this thesis that could accrue several advantages, as:

- Formal specification languages and notations to include modeling of security related features such as privacy, integrity, access control, etc.
- Unified formal model of CPS and security property.
- Facilitating communication and common understanding between different and divergent stakeholders (CPS have varied hardware, software, and communication components).
- Leverage of existing standards-based tools and languages for the design and analysis activities of CPS.

The existing modeling approaches of dependable CPS may be classified into two distinctive categories. The design-time modeling approaches define and specify all parts and relationships, assuring the required system quality properties. However, in today's complex CPS, ensuring all quality properties in all operating conditions during design time will never be possible. Therefore, an additional line of defense against safety and security incidents is indispensable: This must be provided by the run-time approaches that offer models to monitor the CPS during operation. They detect anomalies in system behavior or interface functioning and take autonomous mitigation measures, thus attempting to prevent imminent safety accidents or security incidents before they occur. Thus, run-time execution allows CPS to interact with the real world in real-time. This is crucial for applications where immediate response to external stimuli is necessary. However, testing in certain environments or conditions may be impractical, expensive, or risky. Thus, and in most cases, simulation involves creating computer-based models that replicate the behavior of the real-world CPS, it facilitates iterative development and testing, allowing engineers to refine and optimize the system before its deployment in the real world.

1.2 Research Questions

Our solution presented in this thesis investigates and advances CPS modeling and analysis, adopting in a complementary manner the ideas of the two above

approaches while using the most appropriate formal methods. Despite being a well-known practice in software development projects, the adaptation of formal models to secure CPS modeling still requires systematic elaboration. The complex interactions between cyber and physical spaces and their reflection on the security property of a CPS constitute a significant challenge for the development teams. Therefore, we approach the design of a given CPS by first developing its conceptual and formal models while considering a set of scenarios explaining its secure behavior. Then, we proceed to its simulation, validating the CPS in a controlled environment. The refinement of the proposed models, based on the insights gained from the simulation results, allows to iterate through the process of model development, simulation, and analysis until we achieve the desired accuracy and reliability.

The central question that this thesis project asks, is: "How to define suitable formal models for the specification of this new class of engineered systems (CPS) at design time, as well as their secure behavior execution and verification".

Thus, research questions that are consequently asked are:

- RQ1: How to extend software engineering modeling approaches to deal with the CPS design?
- RQ2: How to integrate security considerations in the early life cycle activities of CPS?
- RQ3: What specific language and formal notations may be used to model CPS and their security property?
- RQ4: What are the specificities of the implementation and simulation framework to be considered in order to support modeling of evolving CPS and reasoning about their properties.

1.3 Contributions

Aiming accomplish the general objective and answer the research questions proposed for this thesis, the key contributions are summarized as follow:

1. CT1: We propose a comprehensive methodology for an incremental software development process of secure CPS, recognizing that it is crucial to integrate security practices throughout the development lifecycle. Three complementary phases are identified:

- Analysis and Architecture Design: Identify and prioritize security requirements along with functional requirements during the design of the CPS conceptual models and consider the relevant scenarios.
- Formal Modeling: Transcribe the high-level CPS architecture that outlines the system's components, modules, and their interactions, as well as its secure behavior to some formal models allowing its formal specification and analysis.
- Models Simulation and Implementation: To remove the gap between the obtained formal models and their execution models, we formulate the specifications for CPS in terms of BPMN models. Then, we simulate their behavior execution and communication through possible activities processes. An iterative process between these two last steps is undertaken in order to get more accuracy and guarantee for the formal specification of CPS.

2. CT2: We provide an important reference ISO/IEC/IEEE 42010:2011 standard for software system architects and developers, leading to propose a standardized approach for CPS architecture description that facilitates communication, decision-making, and quality assurance in software development of such systems. Effective software architecture enables teams to manage complexity, facilitate communication among stakeholders, and deliver software that meets both functional and non-functional requirements. Thus, we propose a step-by-step guide on how we use IEEE 42010 in the CPS architecture design process to create a secure architecture.

3. CT3: We define a new formal model called CA-BRS, combining the two formalisms: Bigraphical Reactive Systems (BRS), suggested initially by Robert Milner [23] and the Control Agents (CA), as abstract and intelligent entities. This semantical model allows to specify the structural, the behavior and the security aspects of a CPS. BRS are a unifying formalism that has been used for representing many existing formalisms for concurrency and mobility. Besides, they are suited for reasoning about current systems, essentially formed of spatially-dispersed components involving computation and communication. The BRS extension that we will propose, takes into account specific nodes called "Control Agents" endowed with a certain intelligence, allowing them to observe, analyze and execute actions (in the form of specific

reaction rules) on the bigraphs that host them. Indeed, the extended BRS with the notion of agent as virtual entity will constitute an appropriate formalism to model CPS in general. Especially, the physical layer is specified using bigraphs. The virtual layer is described with abstract algebraic agents. The agents operate on a physical structure and can observe, control, migrate and communicate.

4. CT4: We extend the bigraphical reaction rules of BRS to model the dynamic of CA-BRS. Their formal execution, through a Guided Transition System (GTR), constituted of several Control Reaction Rules (CRR), supports two types of evolution over time; physical, which corresponds to a sequence of bigraphs, and virtual conducted by the control agents that move and migrate (cyber mobility), while considering some material constraints through the "Observations".

5. CT5: Such high-level reasoning with CA-BRS may aid in analysis of CPS behavior. In this step, we use two distinct perspectives to capture two possible implementation of Control Agents (CA). States perspective is used to describe the behavior of a reactive agent, through its possible states and transitions between those states, in response to the events that occur. Activities perspective used to model business processes of a non-reactive agent; it is defined as the flow of activities, including decisions, loops, and concurrent activities. Both perspectives (States and Activities) are used to model the behavior of CA. Each one uses a particular notations to represent the different states and transitions of CA. They are both useful for understanding and communicating this behavior to stakeholders. In this work, we implement the Activities perspective with BPMN notation, as an industry standard developed by the OMG consortium, easily understand by both technical and non-technical stakeholders, and the States perspective with Maude which is a high-performance language and system supporting both equational and rewriting logic computation for a wide range of applications.

6. CT6: We proceed to check some important properties and non-functional requirements in the early stages of design, before implementing the actual CPS. Generally, security properties of evolving CPSs as any distributed system can be roughly classified into: Availability (Information is said to be available if authorized entities can access it at any

time), Integrity (Information is said to be complete if it can only be modified or deleted by authorized entities) and Confidentiality (Information is said to be confidential if it is only accessible by authorized entities). Then, we demonstrate the applicability of our approach in two use cases, from two different application domains, namely Medical-CPS and Access Control rules in communication networks, addressing these three security property.

1.4 Thesis Outline

This thesis monograph is structured as indicated in Figure 1.1.

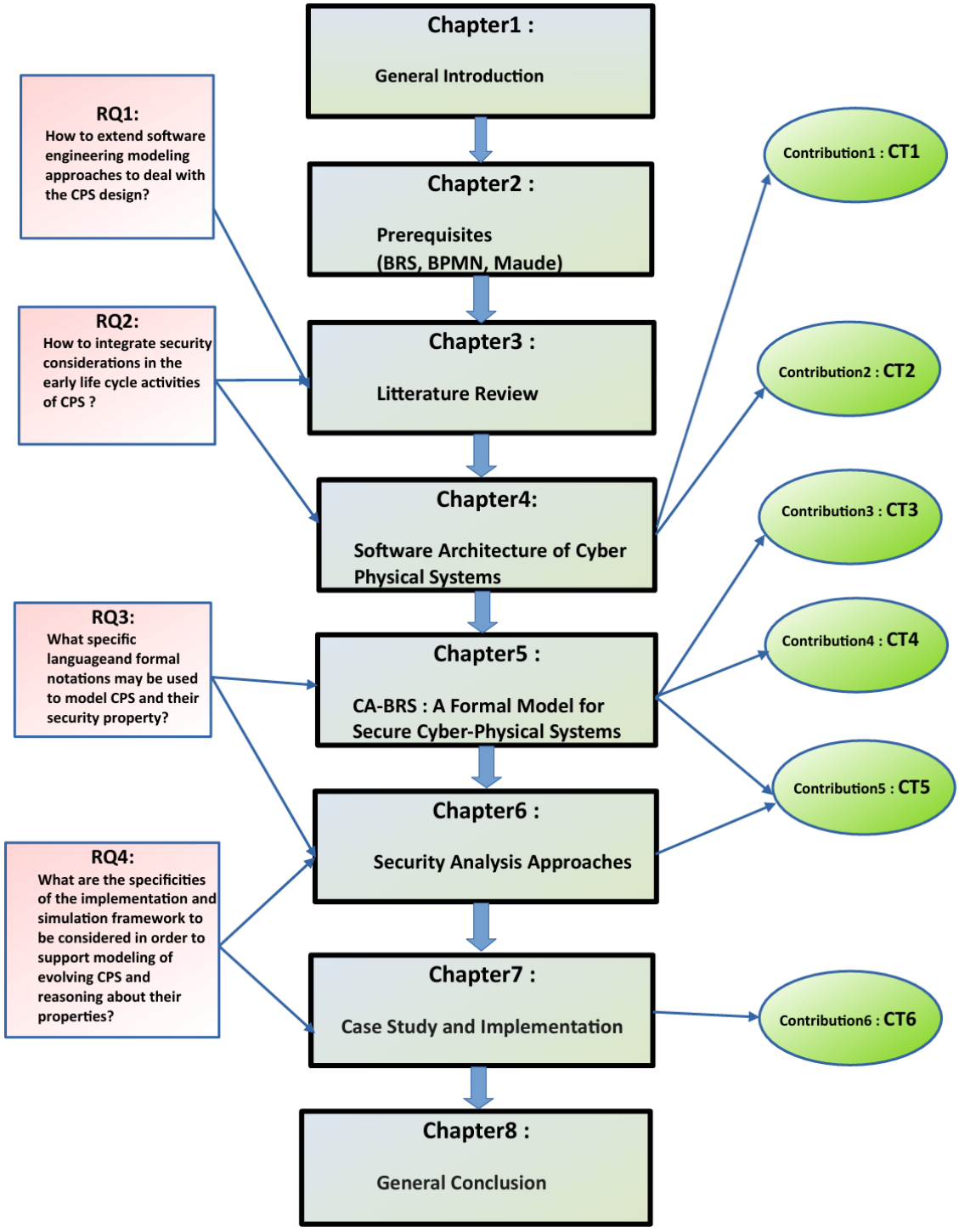


Figure 1.1: Chapters Organization

Chapter 2 serves as a guide to the prerequisites required for a comprehensive understanding of the subject matter covered in this thesis. It outlines the fundamental concepts and theories of Bigraphical Reactive Systems, Business Process Modeling Notations and Maude language that form the building blocks for the subsequent chapters.

Chapter 3 not only delves into the examination of CPS security but also delves into current research and industry trends related to it. Additionally, it identifies the primary research challenges and open problems in the field of CPS security. Lastly, it emphasizes significant works in the literature that enable the formal modeling of CPS and the analysis of their security aspects.

Chapter 4 introduces an iterative design process consisting of three distinct phases: the formal specification phase, which is considered the most crucial, followed by the software architectural design phase, and finally the implementation phase. Furthermore, the chapter provides a comprehensive overview of the IEEE 42010 standard for Architecture Description of Software-Intensive Systems, and offers a step-by-step guide on utilizing the IEEE 42010 standard in the process of designing CPS architectures.

Chapter 5 introduces a novel BRS-based model, called CA-BRS (Control Agents & BRS), which offers a comprehensive representation of all elements of CPS with a specific emphasis on security aspects. The chapter provides a thorough description of the syntax and semantics of CA-BRS, presenting a formal framework that captures the interactions between physical and computational components within CPS. Importantly, CA-BRS takes into account potential security threats and vulnerabilities, enhancing the model's ability to address security concerns effectively.

Chapter 6 and Chapter 7 offer a practical perspective on the subject matter, demonstrating how the theoretical foundation of CA-BRS can be effectively translated into practical solutions. The exploration of the state space in Chapter 6 provides a comprehensive understanding of the dynamic behavior and potential outcomes of the CA-BRS model. By analyzing the state space, we can identify critical states, evaluate system performance, and assess the security aspects of the CPS under examination. Furthermore, chapter 7 presents specific case studies and their corresponding implementation, offering valuable insights into the challenges, strategies, and it outcomes associated with the successful implementation of CPS security measures. This hands-on approach effectively bridges the gap between theoretical concepts and real-world application, allowing for a more holistic understanding of how to apply CPS security principles in practice.

Chapter 8 gives a summary of this thesis and discusses directions for future work.

Chapter 2

Prerequisites: BRS, BPMN, and the Maude System

2.1 Introduction

This chapter serves as a guide to the prerequisites required for a comprehensive understanding of the subject matter covered in this thesis. Describes the fundamental concepts, theories, and tools that form the basis of the following chapters.

The chapter begins by providing readers with a clear understanding of what they should expect to achieve by the end of this chapter. Then it proceeds to introduce the key prerequisites, divided into three sections based on various thematic areas. Section 2.2 provides an in-depth explanation of Bigraphical Reactive Systems (BRS) and their core components. Section 2.3 familiarizes the reader with the basics of Business Process Modeling Notation (BPMN), including its main concepts. Section 2.4 aims to bridge any knowledge gaps in Maude, a high-level formal specification language that supports formal modeling, analysis, and verification of systems.

Each section provides a concise overview of the topics, highlighting their significance and relevance to the subsequent chapters. In addition, where necessary, references to external resources and recommended readings are provided for further study.

2.2 Bigraphical Reactive Systems: Definitions and Notations

Bigraphical reactive systems (BRS) [1] are conceived as a unifying framework for designing models of concurrent and mobile systems. These reactive systems are construed as a set of rewriting rules together with an initial bigraph on which the rules operate. Bigraphs are algebraic terms that may be represented as a combination of two graph kinds allowing the representation of communication among the system constituents as well as their spatial configuration. BRS differ from traditional formalisms such as Z notation, Petri nets, B method and process algebra; they are doted of a specific graphical appearance and have the ability to represent both the locality and connectivity of ubiquitous and distributed computing systems. In general, BRS allow to:

1. sketch the structure and interaction patterns of a system composed of more than one component,
2. specify the dependencies between different components of a system,
3. describe a set of reaction rules (transitions) to model the behavior of a concurrent system, and
4. specify clearly system properties.

2.2.1 Static Structure

A bigraph is so called because its nodes are structured in two ways.

The first structure is placing; the nodes are nested inside one another, giving an ordered set of trees, i.e. a forest. In our generic example of Figure 2.1 there are two trees; each has a root (not itself a node) represented by a dotted rectangle.

The second structure is linking; the ports of the bigraphs are partitioned into links, shown by curved lines. A link may be open or closed; each open link has a distinct name (here x). Names allow bigraphs to be joined via their open links.

The two structures are totally independent and form through their interfaces a given bigraph structure $G =< G^P, G^L >: (i, X) \rightarrow (j, Y)$, where:

- $G^P = (V, ctrl, prnt)$ is the place graph,
- $G^L = (V, E, ctrl, link)$ is the link graph.

We note G^P and G^L share the same set of nodes V and their control $ctrl$.

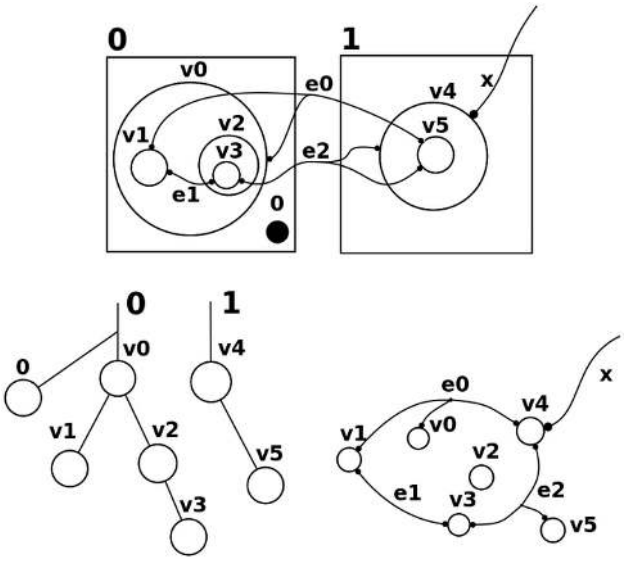


Figure 2.1: A Bigraph generic example

Definition 2.2.1. Interfaces

An interface $(i, x) \rightarrow (j, y)$ of a bigraph is a double pair i and j indicate the number of holes or regions, and x, y are elements of the set of inner or outer names X or Y. $i' \in I = 0, 1, 2, ...i$ and $j' \in J = 0, 1, 2, ...j$

First, each node of a bigraph is assigned a control, which determines how many ports it has. These controls are given by a signature, such as $K = \{v1 : 2, v2 : 0, v0 : 1\}$ in Figure 2.1. Nodes may be nested arbitrarily, and their ports may be linked independently of the nesting. Each bigraph has an inner face and an outer face;

Bigraphs are algebraic terms that may be represented as a particular kind of graphs allowing the representation of communication among nodes, as well as their spatial configuration (nodes may be nested within each other).

The encompassing rectangle represents a region and the grey circles are used to represent holes. Region and hole are the root and leaf node respectively in the placing graph, and enable the composition of placing graphs,

e.g., a hole of a bigraph can be replaced by a region of another bigraph with the aid of composition operator defined in Section 2.2.3 below. Ports are represented as black dots on the node, and are used to connect the edges or names, i.e., the node $v4$ has three ports which connect to the edges $e0, e2$ and the inner name x. The edges $e0, e1, e2$ and inner name x are contained in the linking graph (see Figure 2.1). We can also merge inner names and outer names using the bigraph composition operator. Below we give the definition of a bigraph.

Definition 2.2.2. Bigraph
A bigraph is a 5-tuple: $(V, E, ctrl, prnt, link) : (i, X) \rightarrow (j, Y)$ where:

1. V is the set of nodes,
2. E is the set of hyperedges,
3. $ctrl : V \rightarrow K$ is the control map that assigns controls (and therefore arities) to nodes,
4. $prnt : I \cup V \rightarrow V \cup J$ is the parent map that defines the tree structure in the place graph and,
5. $link : X \cup P \rightarrow E \cup Y$ is a link map that defines the link structure (P is the ports set).
6. The inner interface (i, X) indicates that the bigraph has i holes, and a set of inner names X.
7. The outer interface (j, Y) indicates that the bigraph has j regions and a set of outer names Y.

Example: We illustrate bigraph notations through the following example that specify an organizational structure of intelligent health center. In fact, the way an intelligent health center is structured, establishes important conditions for both the process of care and its outcomes. A comprehensive and holistic organizational structures can help hospital employees understand their day-to-day responsibilities, facilitate decision-making, and revitalize employee performance and productivity. In this example, we are only interested by the design dimension of such structure. We note various nodes representing possible several divisions, units or services. Links relying

nodes specify interactions and control functions between health center constituents. For instance, node Medicine drone1 belonging to the Responders node may have an interaction with Fog node to ...

The bigraph in Figure 2.2 describes a given organizational structure of an intelligent health center. Before presenting the system itself, we proceed with the description of the different node controls that we will use in our example, summarized in Table 2.1. Controls *Monitoring camera, AMB, Nurse'PDA*, etc, denote the atomic nodes of the system, meaning that no node may be nested inside them. The *AMB* and *RES* nodes have arity 1 and *AMB_S* have no ports. The non atomic controls (the compartments) are *AMB_S*, with arity 0, and *Internal Unit*, with arity 0

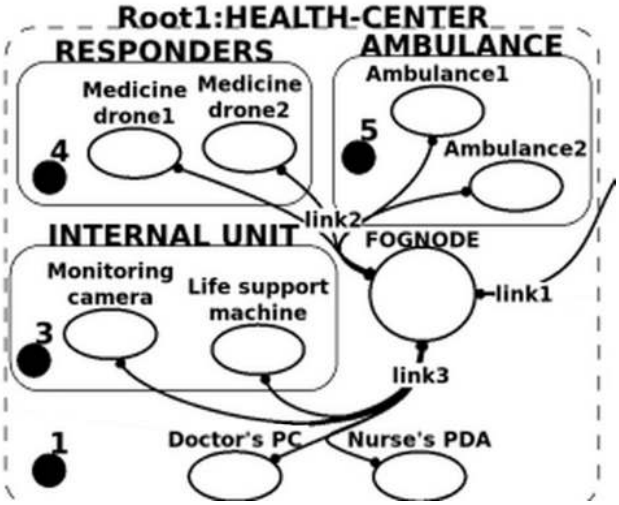


Figure 2.2: B_{HC}: A Bigraph example for Intelligent Health Center architecture

Terms Language: We have introduced far seen two ways of representing a bigraph: as a 5-tuple with interfaces, or using a graphical notation, we give in this section a third way to describe a bigraph, textual term representation.

The language primary elements and operations for composing elementary bigraphs are listed in Table 2.2 For instance, the parallel product $U||V$ term may be used to compose bigraphs by juxtaposing their roots and merging their common names. The merge product term $U|V$ allows juxtaposing two bigraphs and merging their regions into a single one. As an additional example is the nesting operation $U\dot{V}$ which denotes a bigraphical term placed

Table 2.1: Controls and sorts used in a bigraph B_{HC}.

CPS Constituents	**CA-BRS Concepts**
Architectural part of a CPS	B_{CPS}
Geographical location	Root (number of roots is n)
Cyber Constituents(Cloud service, Application, etc.)	V_{Cy}
Physical Constituents(Computer, Processing Unit, Power Unit, Communication Unit, Sensor Nodes)	V_{Ph}
Nesting of a component in its composite	$prnt_{CPS}$
Possible interaction between components	$link_{CPS}$
Control part of a CPS	CA_{CPS}
Abstract agent for physical entities monitoring	A_{Ph}
Abstract agent for cyber entities monitoring	A_{Cy}
A possible set of the Ai agent states	$St_{CPS}(Ai)$
Relationship Architecture/Control in CPS	$Host_{CPS}$
Possible location of Agents controlling Cyber entities	$Host_{Cy}$
Possible location of Agents controlling Physical entities	$Host_{Ph}$
Dynamic action in CPS	
Possible changes in CPS architecture	AC_{CPS}
Possible state change of Agents	TR_{CPS}

inside another one.

Sorting: A bigraph can be further associated with a sorting discipline allowing to impose structural constraints on the different bigraphical elements such as nodes, links and sites. This sorting logic allows constraining on one hand, the children of a node to have only certain controls, and on the other hand, the linkage allowed for possible ports of a node.

A sorting discipline Σ is a triple (Θ, K, Φ) where Θ is a non empty set of sorts. K is the bigraph signature, and Φ is a set of formation rules associated with a bigraph. The latter consists of a set of structural constraints a bigraph has to satisfy.

Table 2.2: Terms language for bigraphs.

Term	**Signification**
$U \| V$	Parallel product
$U \mid V$	Merge product
$U.V$	Nesting (U contains V)
K_x	Node of type (*control*) K having a name or link x
id	Identity or elementary bigraph
d_i	Site numbered i
1	The barren (*empty*) region

Note that disjunctive sorts are written as $\widehat{xy}$ which means a node can either be of sort x or y. This definition ensures that only well formed bigraphs in a given application domain are specified.

For instance we define a sorting discipline for the class of bigraphs used to describe the intelligent health center architecture; it imposes the necessary structural constraints via formation rules Φ_{HC} as indicated in Table 2.3 to ensure that these bigraphs are designed in a meaningful way.

Table 2.3: Formation rules for B_{HC}

Φ_{HC}	**Rule description**
Φ_1	All $\widehat{a\,m\,c\,s\,d\,n}$ f-nodes are atomic
Φ_2	All children of rs-node have sort m
Φ_3	All children of as-node have sort a
Φ_4	A $\widehat{a\,m}$-nodes are always linked to one f-node

2.2.2 Dynamics

Dynamic behavior in terms of system evolution is defined in BRS via reaction rules. Two types of reaction rules are possible on a bigraph. The operation of mobility in the system, it represents the arrival or departure of an entity (represented by a node). The operation on links that expresses the connection (or disconnection) of a bigraph node, through one of its interfaces. The possible system configurations, as well as the reaction rules specify how these configurations can evolve.

Definition 2.2.3. Reaction Rule

A reaction rule R, noted (B, B', η), is a couple of bigraphs (B, B') such as: The Redex B specifies the bigraph to transform; The Reactum B' specifies the bigraph after the transformation and the η is a transformation of ordinals.

Example: We now explain how bigraphs can reconfigure themselves, using our example as illustration. We note that the dynamics of our system in this case, concerns only its structural aspect. This will be defined by means of reaction rules, each consisting of a redex (the pattern to be changed) and a reactum (the changed pattern). These patterns are both bigraphs, so they may involve both placing and linking. Here are two possible rules $R1, R2$, for our example, the intelligent health center (in state B_{HC}) shown above.

Rule $R1$ is the simplest: a $Doctor'PC$ can be disconnected. The redex (the left hand pattern) can match any $Doctor'PC$; the out-pointing links mean that his ports may at first be linked to one or more other ports (relying to *Fog node, Monitoring camera*, etc), in the same place or elsewhere. the reaction by $R1$ will unlink him; any link to a $doctor'PC$ is omitted. This rule change only the linking (not the placing) in a bigraph.

Rule $R2$ by contrast, changes the placing; an $Ambulance1$ leaves a garage. Again, the rule requires the $Ambulance1$ to be in the garage node ($Ambulance$). The site (shaded) represents a parameter of the rule; it allows the garage to contain other ambulances.

The Figure 2.2 represents a state which may change because of the execution of Rule1 and Rule2, and perhaps other constituents movement. The resulting system state ($B'1$ and $B'2$) are shown in Figure 2.3 after executing the two reaction rules $R1$ and $R2$.

Bigraphical reactive systems (BRS) augment bigraphs with a rewriting theory that allows models to evolve over time. The rewrite theory consists

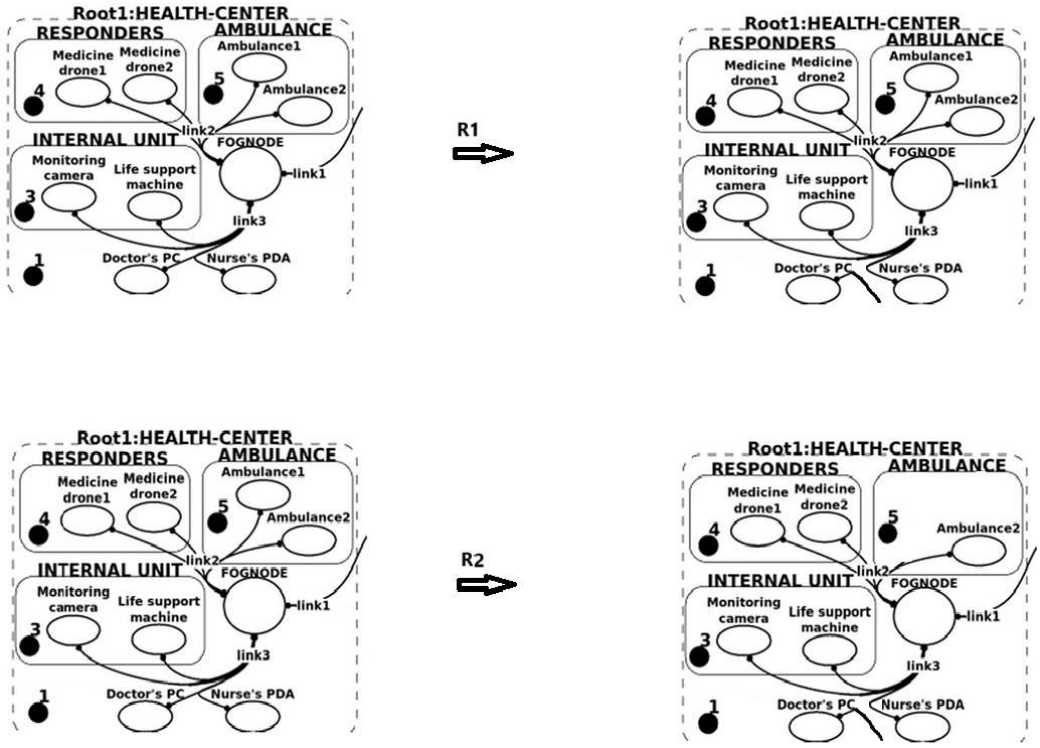


Figure 2.3: Example of Two Reaction Rules R1 and R2

of a set of reaction rules (B, B', η) that finds an occurrence of B in a larger bigraph C and replaces it with B'.

Definition 2.2.4. BRS A bigraphical reactive system $BG(K, R)$, is a category with a signature K (the set of constituent controls), and a set of reaction rules R, where the objects are interfaces and the arrows are bigraphs, equipped with a set of reaction rules R.

For instance, let BRS_{HC} be the bigraphical reactive system formed by the reaction rules $R1$ and $R2$ defined previously and the set of bigraphs B_{HC}, specifying the dynamics of some physical constituents of the intelligent health center.

2.2.3 Bigraphs Composition

As we have seen previously, the bigraph interfaces may be used to conceive larger bigraphs from smaller ones; this construction is defined in terms of the constituent place and link graphs. In this section, we show how complex bigraphs can be designed from elementary ones, by applying composition and tensor product operations ($\circ$ and $\otimes$ operators) [1].

Definition 2.2.5. Composition Given two bigraphs $B : (i, X) \to (j, Y)$ and $A : (j, Y) \to (n, Z)$ whose nodes and edges are disjoint. Then the composition $G = A \circ B : (i, X) \to (n, Z)$ is defined by:

- the roots (regions) of B replace the sites (holes) of A;
- the names of the inner interfaces in A are linked to the names of the outer interfaces in B.
- the number of roots in B must be equal to the number of sites in A.

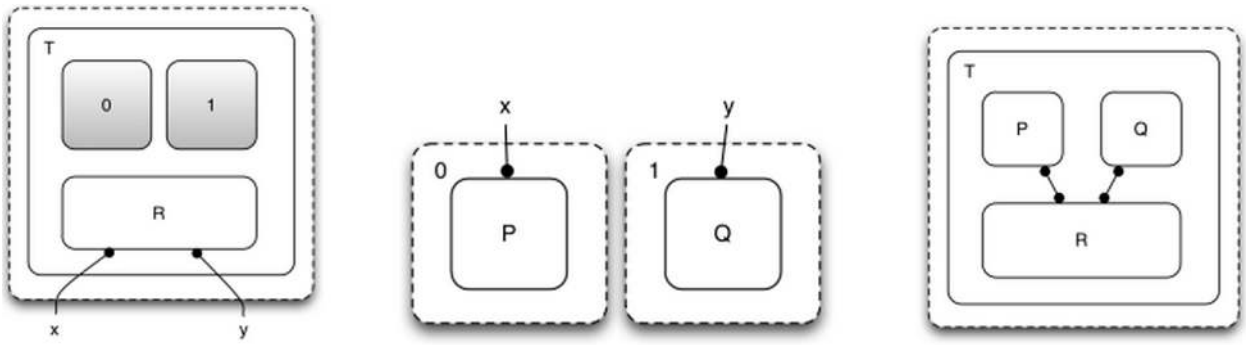
Bigraphs are composed separately in the place and the link graphs. The interfaces of the bigraphs must be compatible in order for composition to be defined.

Figure 2.4 illustrates the composition $A \circ B$ of bigraphs A and B. In the place graph, we insert contents of the left-most region of B into hole 0 of A, and the contents of the right-most region of B into hole 1 of A. Regions are numbered left-to-right: we insert the contents of region 0 into hole 0 etc. In the link graph, links are spliced together where there is name agreement between the inner and outer names of the bigraphs being composed. We may refer to A in this case, as a context into which B is inserted.

There exists an additional way to combine bigraphs, namely the tensor product $A \otimes B$, where A and B are bigraphs. We note that A and B do not share any inner or outer names; this just involves juxtaposing their place graphs, taking the union of their names, and increasing the indices of holes in B to make them unique with respect to A, see Figure 2.5 for illustration.

Definition 2.2.6. Tensor Product

Given two bigraphs $A0 : (i0, X0) \to (j0, Y0)$ and $A1 : (i1, X1) \to (j1, Y1)$ where $X0, X1$ are disjoint and $Y0, Y1$ are disjoint. Then the tensor product $A0 \otimes A1 : (i0 + i1, X0 \cup X1) \to (j0 + j1, Y0 \cup Y1)$. Thus, the tensor product $A0 \otimes A1$ of $A0, A1$ is formed only by placing them side by side.



(a) $A : \langle 2, \{x, y\}\rangle \to \langle 1, \emptyset\rangle$ (b) $B : \langle 0, \emptyset\rangle \to \langle 2, \{x, y\}\rangle$ (c) $A \circ B : \langle 0, \emptyset\rangle \to \langle 1, \emptyset\rangle$

Figure 2.4: The composition $A \circ B$ of two bigraphs A and B with their respective interfaces [1].

2.2.4 Extended BRS

With the development of modern distributed and mobile systems, modeling and analyzing their increasing complexity is of great significance for software engineers. Milner et al. [1] have first introduced a bigraph behaviour model where it is possible to design and analyze reactive systems that include spatial and non-spatial relationships of physical or virtual entities. Bigraphs evolve over time using a rewriting framework BRS that finds instances of a (sub)-bigraph, and substitutes a new bigraph.

Pereira et al. [24] have extended Bigraph by introducing the agent concept. In their BiAgent model, computational or virtual entities are agents and the physical entities or world is a Bigraph. Entities such as vehicles, persons, computers or smartphones are nodes related by the bigraph placing or linking graphs. Agents model the disembodied or virtual entities such as computations or minds. They interact with the bigraph by observing it, being hosted in it, migrating in it, and controlling it [24].

In addition, Pereira et al. [25] seek at leveraging the use of the widely accepted model of computation known as Actor [26] for specifying the agents. An actor system is composed of autonomous objects called actors. Actors communicate using asynchronous message passing. An actor encapsulates a state and a thread and has a mail-address used by other actors to send it messages [26].

Thus, we note that adding reliable inter-agent messaging in BiActors model constitutes a specialization of BiAgents. This may allow the imple-

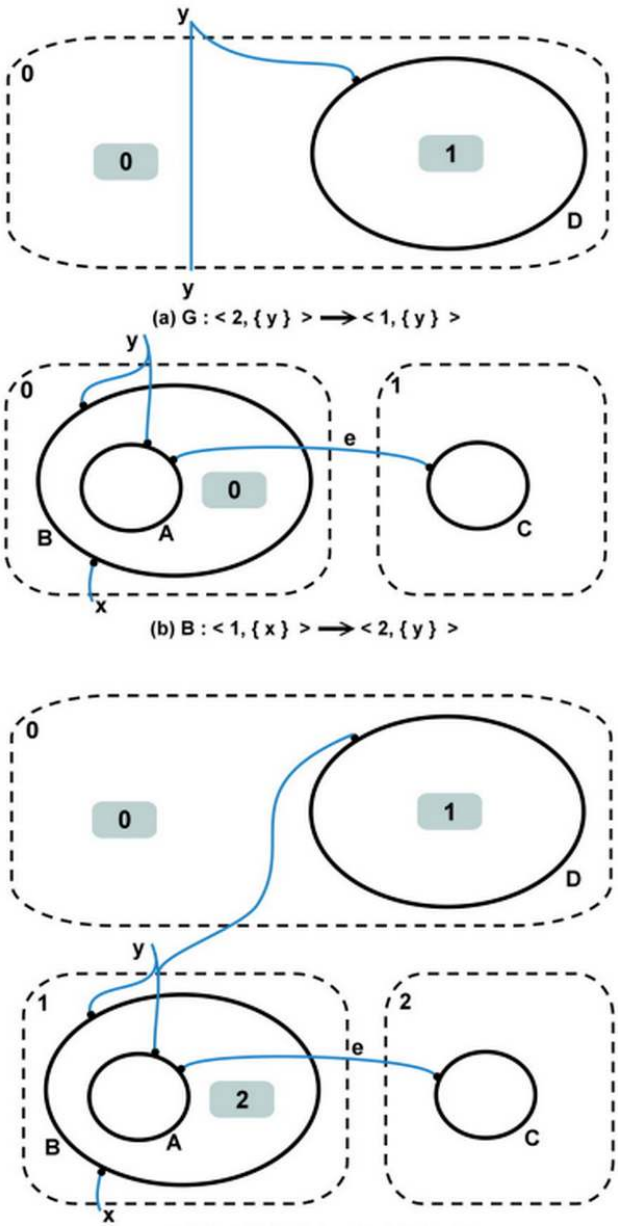


Figure 2.5: The Tensor product $A = G \otimes B$ [1].

mentation of the agent entities in Actor languages such as Erlang, Scala, and Cloud Haskell [26]. Morever, adding a higher-level synchronous semantics to just the agent time, might specialize BiAgents to a BiLustre or BiGiotto,

allowing the implementation of the agent in these langages [26].

In parallel to the cited class of bigraph extensions, other works have been interested in redefining the constraints on the locality of edge (in link graphs) by adding a probability to edges giving rise to stochastic bigraphs [27]. Bigraphs with node sharing have also been defined [28], introducing the possibility that a node can inherit from several parent nodes.

In the same thought, we will give another extension of BRS that aims to formal specify and analyze distributed systems dealing with physical and cyber components.

2.3 Business Process Modeling Notations

BPMN, or Business Process Model and Notation, is a standardized visual language used for modeling business processes. It provides a graphical representation of processes, activities, and the flow of information within an organization. Here is a brief reminder about BPMN:

1. Purpose: BPMN serves as a communication and documentation tool to represent business processes in a clear and standardized manner. It enables stakeholders to understand, analyze, and improve processes within an organization.

2. Symbols and Elements: BPMN uses a set of symbols and elements to represent different aspects of a business process. These include events (start, intermediate, and end), activities (tasks and sub-processes), gateways (decision points), and flows (sequence flows and message flows), among others (see Figure 2.6).

3. Process Modeling: BPMN allows you to model the sequence of activities, decisions, and events that make up a business process. It captures the logical flow, dependencies, and conditions within the process, providing a visual representation that facilitates analysis and optimization.

4. Collaboration and Choreography: BPMN supports modeling collaborative processes involving multiple participants or organizations. It allows for the depiction of message flows and interactions between different entities, ensuring a clear understanding of how they collaborate and exchange information.

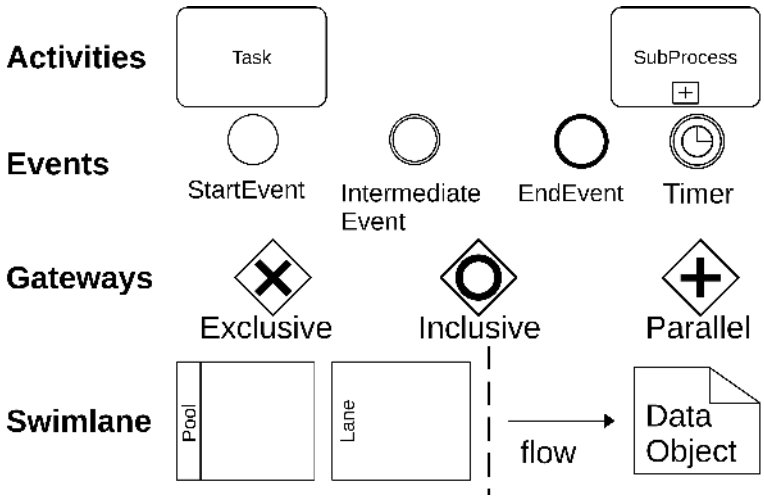


Figure 2.6: Basic elements of BPMN

5. Extensibility and Integration: BPMN can be extended and integrated with other modeling languages and tools, enabling the incorporation of additional details and integration with other systems or methodologies.

6. Process Analysis and Optimization: BPMN diagrams can be used to analyze and optimize business processes. By visualizing the flow and identifying bottlenecks or inefficiencies, organizations can make informed decisions to improve process performance and achieve better outcomes.

7. Industry Standard: BPMN is an industry standard maintained by the Object Management Group (OMG). It is widely adopted and supported by various modeling tools, making it a common language for process modeling and documentation.

Thus, BPMN specification provides a clear and standardized way to represent complex business processes graphically. The fundamental elements of BPMN include:

1. Flow Objects: comprising Events and Activities that represent respectively, occurrences triggering or resulting from activities within a process and work or tasks that are performed as part of a process.

2. Connecting Objects: in these elements category, two possible flows exist: Sequence Flow or Message Flow, they represent respectively the order in which activities are performed. and the flow of messages between different pools or participants in a collaboration diagram.

3. Swimlanes: the element Lanes within a Pool is used to organize and categorize activities. Lanes can represent different roles, departments, or responsibilities within a pool. Thus, Pools represent the major participants or stakeholders in a process. Each pool typically contains its own set of flow objects and can represent a specific department or organizational unit.

Besides, additional elements may be defined to arrange the process modeling as Data, Artifacts, Gateways, etc. Thus, BPMN creates a bridge for the gap between business process design and process implementation [29].

It is important to note that BPMN is extensible, and additional elements or attributes can be added for specific modeling needs. Particularly, the BPMN4CPS extension [2] is designed to model in a convenience way the processes dedicated to CPS. This standard extension is based on BPMN2.0 elements [30]. It introduces the three-part process logic as shown in Figure 2.7: the cyber part, the controller part, and the physical part. Each part has its own type of activities that can be performed. In addition, we also find the modeling of the roles of the CPS devices, the properties of the real world environment and the possible physical entities. Hence, a cyber-physical process must be composed of at least three lanes: a physical, controller and cyber lane. This model is pretty common in cases where the designer wants to be able to present the CPS as a set of processes that interact together, where each individual process represents a physical, control or a cyber part. This idea is taken up in our CPS modeling approach, but we are more particularly interested in the BPMN-based specification of the different agents behavior, controlling the physical and cyber entities of the CPS.

2.4 Maude System

Maude is a high-level language and formal specification language that supports formal modeling, analysis, and verification of systems [31]. It is equipped with powerful rewriting logic and an extensive toolset for exploring and analyzing system behaviors. The Maude system is widely used in academia and

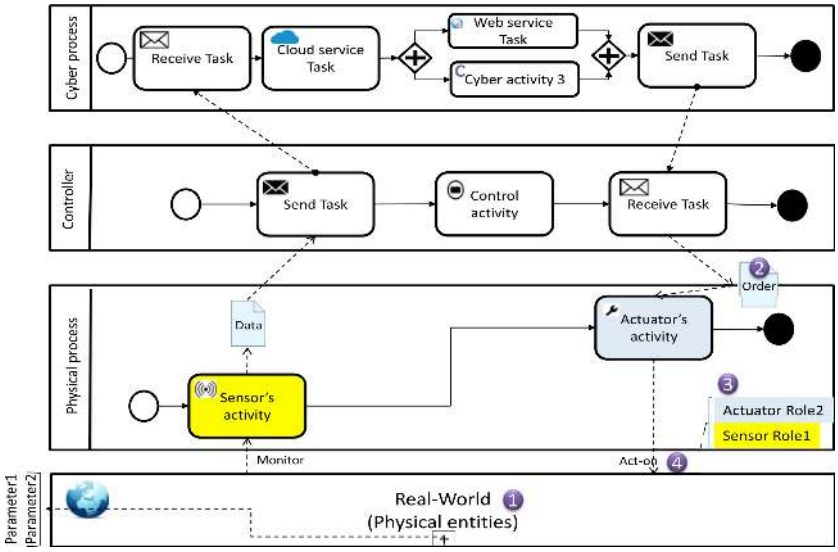


Figure 2.7: BPMN4CPS Example [2]

industry for a range of applications, including software engineering, formal methods, and protocol verification. With Maude, we can formally specify systems using algebraic data types, equations, and rewrite rules. Specifications in Maude capture the structure, behavior, and properties of systems, facilitating rigorous analysis and verification.

The tools set around Maude system includes model checking tools, theorem proving capabilities, simulation tools, and debugging utilities [32]. These tools aid in the verification and analysis of system properties and behaviors.

2.4.1 Maude and Rewriting Logic

Maude is built on rewriting logic, which is a computational model that allows the specification and execution of rewriting rules. Rewriting logic provides a foundation for describing system transitions and behaviors, making it suitable for formal specification and analysis [33]. Indeed, it is a logic to reason about change in a concurrent system, not about equality. Each rewrite rule is a general pattern for a basic action that can occur concurrently with other actions in a concurrent system [33].

Given a signature (Σ, E), with Σ a ranked alphabet of function symbols and E a set of Σ-*equations*. $T_{\Sigma,E}(X)$ denotes the Σ-*algebra* of equivalence classes of Σ-*terms* with variables in X modulo the equations E. In this way,

we free rewriting from the syntactic constraints of a term representation and gain a much greater flexibility, thanks to the "structural axioms" E, in deciding what counts as a data structure (Rewriting in equivalence classes).

The sentences considered in rewriting logic are sequents of the form $[t]_E \rightarrow [t']_E$ with t, t' Σ-*terms*, where t and t' may possibly involve some variables from the countably infinite set X.

Definition 2.4.1. A labelled rewrite theory R is a 4-tuple $R = (\Sigma, E, L, R)$ where:

- Σ is a ranked alphabet of function symbols,
- E is a set of Σ-*equations*,
- L is a set called the set of labels, and
- R is a set of pairs $R \in L \times [T_{\Sigma,E}(X)]^2$ whose first component is a label and whose second component is a nonempty sequence of pairs of Σ-*equivalence* classes of terms, with $X = \{x1, x2, \ldots, xn, \ldots.\}$ a countably infinite set of variables.

Elements of R are called rewrite rules, noted as:
$r : [t(x1, ..., xn)] \rightarrow [t'(x1, ..., xn)]$ if $C(x1, ..., xn)$, $C(x1, ..., xn)$ is the conditional part of the rule wich is optional. Rewrite rule with an empty condition is said unconditional rewrite rule.

Rewriting logic then allows us to reason about what complex changes are possible in a system, given some local changes corresponding to the basic actions axiomatized by these rewriting rules. Thus, we can reason about concurrent programs in a logic intrinsic to their computations. The models of rewriting logic are precisely concurrent systems in the intuitive sense of the word, i.e., they are machine-like entities whose state is distributed and can change by actions taking place simultaneously.

Maude employs term rewriting as a fundamental mechanism for computation and system evolution. Terms in Maude are constructed using constructors and can be rewritten according to defined rules. This feature enables the specification of system behaviors and transformations.

2.4.2 Modules in Maude

Maude is a specification language [32] whose programs are a straight translation to ASCII of the previous mathematical notation for equations, axioms, and rewriting rules. Specifications are organized in modules that can include or extend other modules. Some useful modules are included in the Maude prelude specifying natural and floating-point numbers, lists, strings, sets, reflective operations, and so on.

Pure equational theories are specified in ***Functional modules***, introduced by the *fmod* keyword.

System modules specify rewrite theories by adding rules. In this case, the keyword *mod* is used instead (see the following example).

Example 2.4.1. The following functional module specifies the structural part of a graph. The graph has been represented as a set of edges. Edges are represented as pairs of nodes. The graph is represented as a set of edges. Nodes are represented as elements of a set. Declarations and statements can be annotated with attributes between brackets. The ctor attribute for operator declarations states that the operator is intended to be a data constructor of the range sort. Structural axioms of commutativity, associativity, and identity element $\langle\!\langle nil \rangle\!\rangle$ are respectively written as operator attributes comm, assoc, and id: nil. The syntax for membership axioms follows directly from the mathematical notation of the previous section, all variables in the right-hand side of an equation should appear in the lefthand side of that equation. Maude uses equations as left-to-right simplification rules, which are applied exhaustively to obtain canonical forms of the terms modulo the equations and axioms

```
fmod GRAPH is
pr QID .
pr BOOL  .
***including BOOL .
*** Declaration of sorts
sorts Node Edge Graph EdgeSet .
subsort Node < Qid .
subsort Edge < EdgeSet .
subsort EdgeSet < Graph .

*** Edges are pairs of Nodes
op _->_ : Node Node -> Edge [ctor] .

*** A Graph is a set of Edges
```

```
15
16 op empty : -> EdgeSet [ctor] .
17 op __ : EdgeSet EdgeSet -> EdgeSet [assoc comm id: empty] .
18
19 *** Graph Example
20
21 ops A B C D :  -> Node .
22 ops E1 E2 E3 : -> Edge .
23 op Gr : -> Graph .
24
25 eq E1 = A -> B  .
26 eq E2 = C -> D  .
27 eq E3 = A -> C  .
28 eq Gr = E1 E2 E3  .
29
30 *** Operations on Graphs
31 op Node1 : Edge -> Node .
32 op Node2 : Edge -> Node .
33 op addEdge : Graph Edge -> Graph .
34 op hasNode : Graph Node -> Bool .
35 op hasEdge : Graph Edge -> Bool .
36
37 *** Equations for operations
38 var G : Graph .
39 vars E E' : Edge .
40 vars N N' : Node .
41 eq Node1(N -> N') = N  .
42 eq Node2(N -> N') = N'  .
43 eq addEdge(G, E) = G E .
44
45 eq hasNode(empty, N) = false .
46 eq hasNode(G E, N) = (N == Node1(E) or N == Node2(E)) or
      hasNode(G, N) [owise] .
47 eq hasEdge(empty, E) = false .
48 eq hasEdge(G E, E') = (E == E') or hasEdge(G, E') [owise] .
49 endfm
```

For example, rules can be added to the functional module $GRAPH$ above to obtain a system module as $GRAPHDYN$ in this example. The rules $R1$, $R2$, and $R3$ change the graph structure representing possible reaction rules. These rules define how the graph evolves, such as adding or removing edges/nodes dynamically. Define the state of the system: The state can include a graph along with any other information needed for the system.

```
1        mod GRAPHDYN is
```

```
2  including GRAPH .
3  *** Define the state as a graph
4  sort State .
5  ***subsort Graph < State .
6  ****subsort Node < Edge . ***************
7  op removeEdge : Graph Edge -> Graph .
8  op addEdge : Graph Edge -> Graph .
9
10 *** Example rules for graph transformations
11 op G : -> Graph .
12
13 ops N N' : -> Node .
14 ops  E1 E2 : -> Edge .
15
16 *** Rule to dynamically add an edge
17  crl [add-edge] : addEdge(G,N -> N') => G N -> N' if (hasEdge
      (G, N -> N') == false) .
18
19 *** Rule to dynamically remove an edge
20 ***rl [remove-edge] : removeEdge(G E1, E2) => G if (E1 == E2)
       or removeEdge(G, E2) E1 [owise] .
21
22 endm
23
24
```

Rules may be named by means of an optional label between brackets. This will be very helpful for the specification of strategies in the strategy language. Conditions for rules are equational conditions plus possible rewriting condition fragments $l \Rightarrow r$, which may contain free variables in its righthand side r to be instantiated by matching.

The Maude interpreter provides various commands to execute its programs [32].

- The reduce command simplifies a given term to its normal form with the equations and memberships E modulo the structural axioms B.
- The *rewrite* and *frewrite* commands rewrite a term using all the rewriting rules in the module (as well as the equations and memberships E, all applied modulo the axioms B, as explained above). An optional bound on the number of rewriting steps can be given between brackets.

Figure2.8 gives an example of the rewrite command execution.

```
Maude> rew removeEdge(Gr, E1) .
result Graph: E2 E3
```

Figure 2.8: Execution in Maude of Module Example

2.4.3 The Maude Strategy Language

The Maude specification language provides an implementation of rewriting logic that allows executing, verifying, and analyzing the represented systems. These specifications declare their objects by means of terms and equations, and provide rewriting rules to represent potentially non-deterministic local transformations on the state. Sometimes a controlled application of these rules is required to reduce non-determinism, to capture global, goal-oriented or efficiency concerns, or to select specific executions for their analysis. That is what we call a strategy [34].

In order to express them, respecting the separation of concerns principle, a Maude strategy language was proposed and developed [34]. It is implemented in C++ as an integral part of the Maude system.

Strategy modules in the Maude strategy language are a third type of Maude modules that represent the control of rewriting systems using the strategy language. They are extensions of system modules and can include any declaration or statement allowed in lower-level modules. However, it is encouraged to include only strategy-related statements in strategy modules to maintain a clean separation between the rewriting theory specification and its control [35], [34].

Strategy modules can import other strategy modules and modules of any kind using importation modes such as including, extending, generated-by, and protecting.

In Maude, it is possible to associate multiple Maude strategy modules with the same Maude system module. Each strategy module provides a unique definition of an execution path, thereby enhancing flexibility when dealing with various executed scenarios. The behavior of the system is contingent upon the strategies governing the rewrite rules.

The following example illustrates the general syntax of a Strategy module, building upon the previous example of the $GRAPHDYN$ module.

```
1         smod STRATEGIC-DYNAMIC-GRAPH is
2 including GRAPHDYN .
```

```
3
4
5  *** Strategy specification
6  strat AddThenRemoveEdge : Graph  Edge @ GRAPHDYN  .
7
8  var G : Graph .
9  var E : Edge .
10 sd  AddThenRemoveEdge(G, E) := add-edge(G, E) ; remove-edge(G
       ,E) .
11
12 endsm
13
```

smod and *endsd* are the keywords indicating the start and end of the Strategy module. The declared strategy *AddThenRemoveEdge* is applied to ensure that $add-edge$ is applied first, followed by $remove-edge$.

The strategies are defined using the keyword *Sd* and the corresponding expression. The formation of an expression in Maude involves the combination of multiple strategies or rewrite rules, which can be achieved using various predefined operators. The following Table (2.4) illustrates these operators and their usage in Maude.

Table 2.4: Strategy operators in Maude

Operator	Syntax	functionality
;	$sd\ ST := S1, S2$	Concatenation
—	$sd\ ST := S1 \mid S2$	Union
*	$sd\ ST := (.....)*$	At least 0 iteration
+	$sd\ SST := (.....)+$	At least 1 iteration
!	$sd\ ST := (.....)!$	Iteration until execution end
If...Else	$sd\ ST := S1?S2 : S3$	If strategy S1 is executed, then S2 is executed, otherwise S3 will be executed instead of S1 and S2

Strategies are evaluated in the interpreter using the *srewrite* and *dsrewrite*

commands, which look for solutions of the strategy and show all those they find. In our case, the application of these commands will give the following results:

The *srewrite* command explores the rewriting tree using a fair policy that ensures that all solutions are eventually found if there is enough memory. Not being completely a breadth-first search, it may explore multiple execution paths in parallel. On the contrary, the *dsrewrite* command performs a depth-first exploration of the tree. It is usually faster and uses less memory, but some solutions may not be reached because of *nonterminating* execution branches.

The search conducted by the *srewrite* and *dsrewrite* commands theoretically explores the subtree of the rewriting tree pruned by the restrictions of the strategy, but their search space is actually a graph. The execution engine is able to detect already visited execution states, thus preventing the redundant evaluation of the same strategy on the same term. Consequently, the strategy evaluation may finish in situations where, operationally, *nonterminating* executions are involved [34].

2.5 Conclusion

This chapter lays the groundwork for readers, ensuring they are equipped with the necessary prerequisites to comprehend the content discussed in the following chapters. In this regard, we provided a concise overview of the bigraph model, highlighting its key features. Additionally, we explored the graphical representation of business processes, activities, and information flow within organizations using BPMN. It is important to note that BPMN serves as a standardized notation for modeling business processes, promoting effective communication, analysis, and improvement endeavors within organizations.

Finally, we have showed that the Maude system offers a comprehensive and expressive language for formal specification and analysis. Its features, such as term rewriting, modularization, and robust tool support, make it a highly effective tool for modeling and verifying intricate systems.

In the next chapters, we will discuss the limits of the existing bigraph extentions and motivate our proposed extension dealing with the logical structure and dynamic aspects of a complex system in general, whose physical structure is given by the underlying bigraph.

Chapter 3

Literature Review on CPS: Foundations, Security Challenges, and Research Directions

3.1 Introduction

A Cyber-Physical System (CPS) is a computing system in which physical and software components are deeply intertwined, capable of operating at different spatial and temporal scales, exhibiting multiple and distinct modalities of behavior, and interacting with each other in a way that changes with context [36].

This chapter provides a definition of CPS covering their fundamental concepts while identifying key components and their interactions, then discusses the various types of security threats targeting CPS. In addition, security requirements such as confidentiality, integrity, availability, and resilience are specifically addressed [37, 38].

The chapter also explores current research and industry trends in CPS security and identifies the key research challenges and open problems in CPS security.

Finally, it highlights the most relevant works in the literature, allowing the formal modeling of CPS and their security aspect analysis.

3.2 Cyber Physical Systems History

In 2006, the American National Science Foundation introduced the concept of Cyber-Physical Systems (CPS), marking the inception of a transformative era in science [16]. This initiative garnered significant interest from governmental bodies, academic institutions, and industries alike. Subsequently, in 2008, the United States established the CPS Steering Group, aiming to implement CPS across various sectors including energy, transportation, healthcare, and agriculture [39]. Concurrently, Germany unveiled "Industry 4.0", with CPS emerging as its fundamental technology. Projections indicate that by 2025, the integration of CPS into Industry 4.0 could contribute significantly to Germany's economy, with an estimated added value of 267 billion euros [40]. In the context of China's "Made in China 2025" initiative, CPS are recognized as pivotal in fostering synergy between manufacturing and the Internet [41, 42]. The global emergence of CPS has sparked widespread interest, particularly among nations seeking to secure leading positions in key industries. In particular, the United States has prioritized CPS as a strategic imperative in maintaining its industrial dominance [43].

In 2013, Germany's "Industry 4.0 Implementation Recommendations" underscored CPS as the cornerstone of this industrial transformation [44].

3.3 Fundamentals of Cyber-Physical Systems

With regard to the rapid development of computer technology and computer networks, more and more equipment integrates technologies allowing them to interact with their environment with remarkable autonomy. This technology announced as today's great great revolution in all industrial sectors brings large potential for building systems that improve human life and address societal, technical and environmental challenges, e.g. energy consumption, ambient assisted living, crisis coordination. CPS, which are preferred as a generalization of the concept of networked controlled systems, are engineered systems, built from the seamless integration of computational algorithms and physical components.

3.3.1 Definition

CPS are defined as a set of heterogeneous physical units, e.g., sensors, control modules, communicating via networks and interacting with applications deployed on cloud infrastructures and/or humans to achieve a common goal. CPS as represented in Figure 3.1, integrate an IT system with mechanical and electronic components connected to online networks that allow communication between machines in a similar way to social networks.

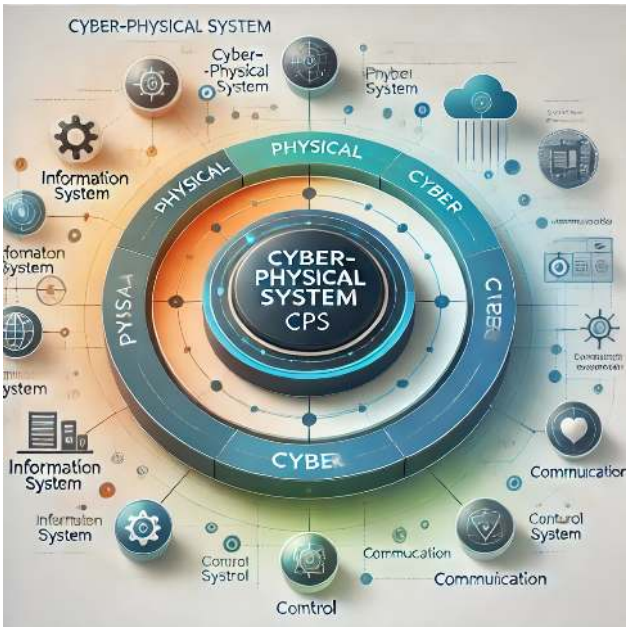


Figure 3.1: CPS Key Elements

Typically, CPS integrate three main parts, a cyber part as a computing core, a physical part as a controlled object, and a network for establishing the communication between the cyber and physical part [45]. What distinguishes CPS is that the integration between the cyber and physical is considered tight. That is, these subsystems must intelligently work together and coordinate seamlessly. This tight integration is aimed to facilitate attractive system properties including adaptability, autonomy, efficiency, functionality, reliability, safety and usability.

However, it should be noted here that there is no standard definition of a CPS. Indeed, some reference definitions presented below, are adopted, all aligning with the fact that a CPS interacts with its environment in which it

collects data, processes it and through a feedback loop, controls or influences the process with which it is associated.

- The NIST formed a group of experts to define these systems. The group defined CPS in more general terms as smart systems that are co-engineered connected networks of physical and computational elements aiming to improve the quality of life in many domains [46].
- Informally, ISO defines CPS as systems that integrate computational components (information processing) with physical processes, which interact through a network. Technological advances in the Internet of Things, Robotics, and Autonomous vehicles are the foundation for making CPS possible, and today there are examples of successful CPS everywhere [47].
- The National Science Foundation (NSF) provides a base definition for CPS, widely accepted today: A Cyber Physical System is a mechanism that is controlled or monitored by computer-based algorithms and tightly integrated with the Internet and its users. CPS systems tightly intertwine the physical and software components, each operates on different spatial and temporal scales, while exhibiting multiple and distinct behavioral modalities [48].

The common definition that we will adopted throughout this work is the following:

Definition 3.3.1. CPS refer to the integration of physical devices and computational (cyber) components, which are tightly interconnected and integrated under a smart decision system.

- The physical devices provide activities that constitute the physical part of the system. So, computing and controlling the dynamics of the physical behavior are the main activities of embedded systems.
- The cyber part of CPS is a set of computing operations, which depend on embedded, Cloud, and Web-based activities. These cyber activities analyze the collected data and interact with the physical environment in order to make decisions.

Thus, the notion of CPS is then developed and emerged out of the generalization of the concept of embedded systems and by parallel contributions from the fields wireless sensors networks (WSN) and Internet of things (IoT). Compared to WSN, CPS involve many more physical components. Embedded systems focus primarily on computing elements, while the IoT is focused on physical components (things) that are connected to the Internet and to each other [46, 49]. In CPS, physical and computational components are equally important. Such systems have the advantage that smart physical devices and wireless networks are used to create intelligent services based on information and components from the related physical environment.

3.3.2 Key Components and Characteristics

A CPS can be a small and local system such as a building management system or a highly connected, complex, and large system integrated over several domains such as a city-scale autonomous transportation system [50]. Compared to traditional embedded systems, CPS are modular, dynamic, networked, and large-scale [51]. They are also increasingly depending on software, which has become their most intricate and extensive constituent [51]. Some of the defining characteristics of CPS include [21]:

- cyber capability in every physical component,
- high-degree of automation,
- networking at multiple scales,
- integration at multiple temporal and spatial scales and
- reorganizing/reconfiguring dynamics.

Besides, CPS are closely related to other fields including embedded systems, robotics, Internet-of-things, and big data [50].

Embedded Systems: Focused on the integration of cyber elements such as processors and software to purely electrical and mechanical systems to perform a specific task.

Robotics: Focused on the seamless integration of sensors, actuators, processors and control to perform a task autonomously or semi autonomously.

Internet-of-things: Focused on dynamic communication network infrastructure with standard interoperable protocols that autonomously communicate data among entities with well-defined and unique identifiers. These entities include physical equipment, virtual elements, computing devices and human users.

Big data: Focused on the systematic analysis, storage, and visualization of a large volume of data.

3.3.3 Overview of CPS Applications in Various Domains

CPS are used for various applications in different sectors, as indicated by Figure 3.2, including manufacturing, transportation, energy, agriculture, smart buildings/structures, emergency response, defense and healthcare. The disruptive and transformative potential of these applications has led governmental entities and researchers to position CPS as the next technological revolution that will equal and possibly surpass the Internet. This section briefly discusses the application of CPS in the following sectors.

Manufacturing: Through CPS, the development of new business models, new services have emerged, changing many aspects of manufacturing industries, thereby producing cyber-Physical Production Systems (CPPS). CPPS consist of autonomous and cooperative elements and subsystems that are connected based on the context within and across all levels of production, from processes through machines up to production and logistics networks. Nowadays, global market place, it is imperative for manufacturing industries to rely on CPS in order to maintain their economic competitiveness [52].

Transportation: CPS has driven innovation across diverse fields including transportation systems, which it is termed as Transportation Cyber-Physical System (TCPS). Compared to a traditional transportation system, TCPS achieve higher efficiency and reliability by enabling increased feedback-based interactions between the cyber system and the physical system in transportation. TCPS is critical to the safety, security and benefit of society and the environment because they represent

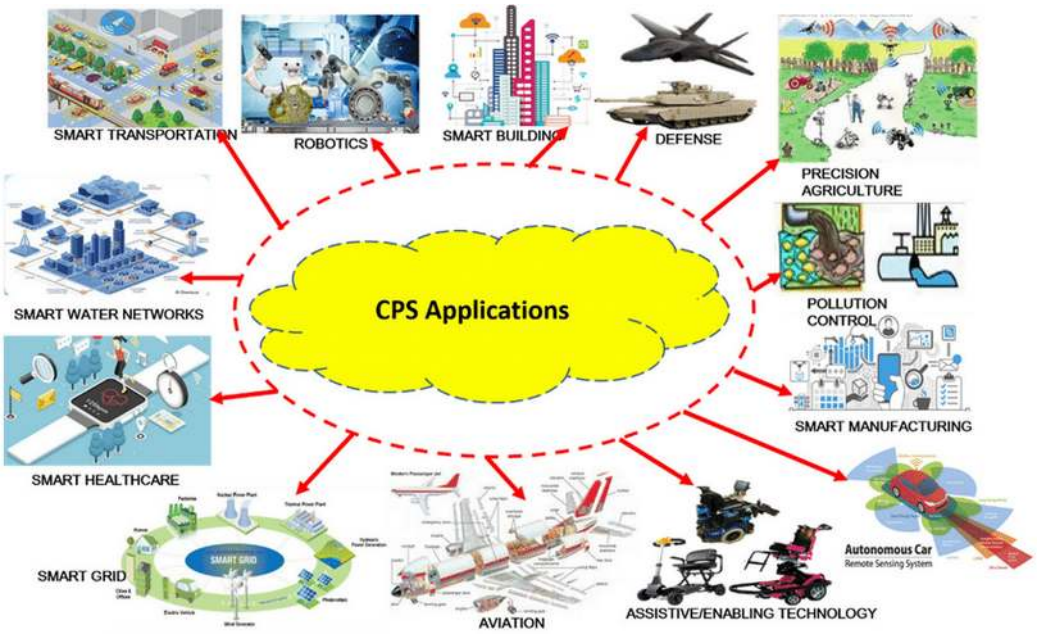


Figure 3.2: Application Domains of CPS [3]

some of the most important infrastructure, such as the systems for aviation, rail, road and marine transportation, and its components for the transportation of both humans and goods [53].

Energy: Cyber Physical Energy System (CPES) are intelligent systems that allow us to add new capabilities to energy systems by integration of communication, computation, information and control. The CPES approach offers some important aspects to address current challenges in energy systems, such as network limitations, computing efficiency, uncertainty prediction, and unified framework of modeling for CPS components of the energy systems [54].

Agriculture: Agriculture Cyber-Physical Systems (ACPS), the CPS designed and applied in agriculture, can collect fundamental and timely information about the climate, the soil, and the crops with high granularity, in order to realize more accurate systems of agricultural management. ACPS can also constantly monitor different resources, such as watering, humidity, plant health and others, through sensors and, thus, maintain the ideal environmental values through actuators and

facilities [55].

Smart buildings/structures: Building management systems are becoming smarter and connected with external infrastructures such as first responders and law enforcement. Moreover, structures like bridges, highways and tunnels use different sensors to improve sustainability and resiliency, as well as safety and reliability. CPS in smart buildings can significantly improve energy efficiency by measuring different quantities, as temperature, occupancy, light intensity and humidity in real-time and adjusting energy consumption. Moreover, CPS can play a vital role in improving the security of building management systems [56].

Emergency Response: Geological disasters can cause extensive destruction of properties, and impact businesses, resulting in physical injuries and deaths. CPS can significantly improve situational awareness and support first responders during natural disasters through their sensor networks, surveillance systems, intelligence, automation and robotics [57].

Military Defense: Defense systems are becoming more reliant on complex, adaptable and autonomous CPS, such as unmanned aerial vehicles, robotics and surveillance systems for meeting the military and national defense needs to reduce the human involvement. Moreover, cyber warfares which rely of CPS has become an important part of offensive and defensive operations.

Healthcare: Medical Cyber-Physical Systems (MCPS) are healthcare critical integration of a network of medical devices. These systems are progressively used in hospitals to achieve a continuous high-quality healthcare. More precisely, they represent a platform through which patient health data are acquired by emergent Internet of Things (IoT) sensors, preprocessed locally, and managed through improved machine intelligence algorithms [58].

In this thesis work, we focus on this later domain for CPS, ideal for studying these complex systems due to its complex requirements for real-time processing, safety, and interoperability. The key benefits of MCPS are: 1) Real-time Patient Monitoring (Wearable health devices collect real-time physiological

data, enabling early detection of health anomalies), 2) Automated Diagnosis and Treatment (AI-driven systems assist in accurate diagnosis and personalized treatment plans), 3) Robotic-Assisted Surgeries (Precision-based robotic surgeries enhance procedural outcomes and minimize recovery times), 4) Telemedicine and Remote Healthcare (CPS facilitates virtual consultations and remote patient management, improving accessibility).

In modern engineering applications, ensuring robustness, flexibility, and adaptability is essential. The design and successful deployment of CPS rely on meeting specific application requirements to optimize performance. In Table 3.1, some common CPS application domains and their specific requirements are illustrated [3].

On the other hand, CPS consist of collaborative, networked, spatially distributed, and tightly intertwined computational (logical) and physical components, each operating at different spatial and temporal scales. Therefore, the spatial and the temporal requirements are fundamentals for their safe and correct execution. In the following section, we will focus on the requirements related to the security of CPS. Indeed, CPS are increasingly applied in critical contexts, where they have to support safe and secure operations, often subject to stringent timing requirements. Typical examples are scenarios involving automated living or working spaces in which humans operate, or human-robot collaborations in modern manufacturing.

3.4 Understanding the Security Challenges in CPS

Security requirements of CPS have been recently recognized as an important challenge to be addressed by the software engineering community. Most of CPS are safety-critical systems whose failure could result in loss of life, significant property damage or damage to the environment. Thus, their safety, reliability and security has higher priority over other objectives such as cost and performance. The safety-critical applications of CPS in sectors like healthcare, transportation, and defense demand novel testing, validation and certification procedures and test-beds from planning to deployment stage including design, assembly, implementation, and delivery stages. In this section, we provide a brief overview of several security attacks/vulnerabilities in the CPS context, while better understanding their security measures. Then,

Table 3.1: Specific Requirements for CPS Applications

Application Domain	**Example**	**Requirements**
Manufacturing	Smart manufacturing, production robotics.	High flexibility, robustness and security, control mechanism, and self-adaptation.
Transportation	Autonomous vehicle, railroad systems, vehicular networks and smart highways, accident signaling, collision avoidance.	High computing power due to complex traffic, robustness, mobility, and self-adaptation.
Energy	Smart grid, smart energy metering, power generation and distribution, energy conservation.	Robustness, self-adaptation, and flexibility.
Agriculture	Precision agriculture, smart irrigation, green house cultivation, disease prediction, and animal localization.	Flexibility, robustness, and scalability.
Smart Building	Green building, asset management.	Precise and reliable control mechanism, robustness, and flexibility.
Emergency response		
Military defense	Firefighter monitoring, soldier supervision, emergency navigation, target tracking, defense jets.	Robustness, flexibility, self-adaptation, military data integrity provision, precise control, high precision sensing, time synchronization, and good coverage.
Healthcare	Smart healthcare, patient's drug administration, robotic microsurgery, health management networks, and epilepsy seizure detection.	Robustness, self-adaptation and flexibility, high precision sensing, data confidentiality, mobility, and quality of service.

we present the attack mechanisms, as well as the defensive measures for each attack.

3.4.1 Security attacks

In CPS, data can be captured by physical objects or sensors and transmitted to a control system over a network. Physical devices are increasingly equipped with bar codes and RFID tags that can be scanned by smart de-

vices, sending identified information over the Internet to monitor and manage the physical environment.

At the same time, computing and processing units can be placed in the cloud, where decisions are generated and sent to physical objects.

The close integration of cyber and the physical world poses significant security challenges on CPS. Consequently, inspired by authors work in [4], we classify the security threats to CPS into three domains: physical, cyber, and cyber-physical domains (see Figure 3.3).

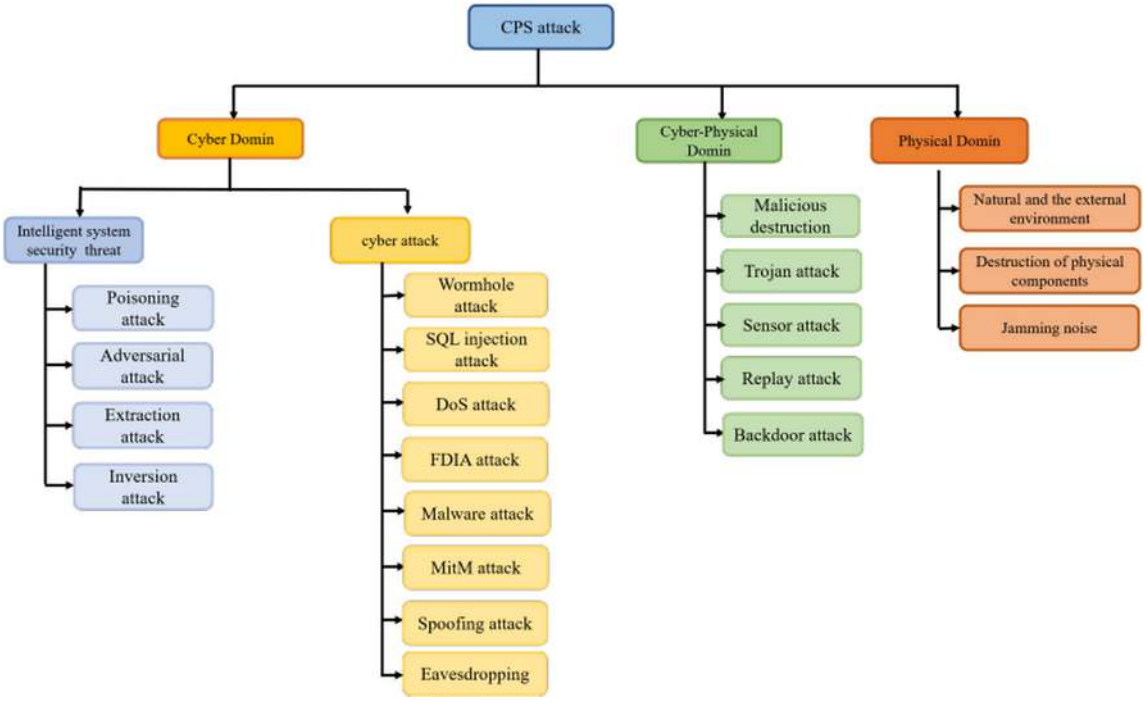


Figure 3.3: Possible CPS Attacks [4]

CPS are becoming more and more vulnerable to various security threats at different layers of the hardware and software CPS stacks, covering both computation and communication layers. Consequently, several security incidents related to physical and hardware attacks on the CPS have been reported in real-world [59]. Based on the CPS layers, treats can be categorized as illustrated in Table 3.2.

In the physical layer, attackers may directly intervene or destroy physical

objects, sensors, and controllers, leading to inaccuracies in measurements, erroneous control decisions, and inappropriate actuator actions.

In the sensor/actuators layer, the confidentiality and sensitivity of data make it highly vulnerable to privacy breaches. Attack vectors include brute force attacks to compromise sensors/actuators for extracting sensitive information (e.g., secret keys, side channel parameters) and manipulating power distribution to either drain energy for denial-of-service attacks or activate malicious payloads.

In the network layer, communication-related concerns arise. Two prominent types of attacks are observed: replay attacks, where messages are misrouted or delayed, and denial-of-service (DoS) attacks, which disrupt network functionality. DoS attacks encompass various techniques, such as jamming, collision, routing misdirection, flooding, wormholes and selective forwarding.

In the control layer, even slight desynchronization among control units or signals can lead to catastrophic consequences as erroneous decisions may result in CPS failures.

In the information layer, attacks typically involve information through eavesdropping or traffic data analysis.

3.4.2 CPS security breaches: Case studies

CPS have to meet stringent design requirements in terms of security, adaptability and dependability. Moreover, the security features need to be intelligent to combat with various attack models, which could even be unforeseen at the design time.

In the following, we cite, on the one hand, two case studies (1-2)that underscore the importance of cybersecurity in protecting CPS from malicious attacks and emphasize the need for continuous vigilance and investment in cybersecurity measures to mitigate such risks. On the other hand, we give two other case studies (3-4) highlighting security breaches related to the physical components of CPS.

1. **Stuxnet Worm Attack [60]:** It is one of the most infamous cyberattacks targeting CPS. It was discovered in 2010 and specifically aimed at

Table 3.2: CPS Threats Categorization

Layers	Attacks	Description
Physical Layer	Direct Intervention and damages	Changes the hardware to damage the system
Sensor/Actuator Layer	Node hacking, Node Destruction, Energy stealing/draining, Cryptography attacks.	Leakages the information directly from Sensor/Actuator via RF signals. Destructs or modifies node physically. Drains out the limited power of Sensor/Actuator. Cracks secret keys with brute force or side channel analysis.
Network Layer	Replay, Communication Jamming, Data Flooding in communication, Spoofing and altering the communication information	Forward message to an incorrect destination. Halts the on-going the communication. Inserts the bogus data into the established communication. Changes routing information illegimately.
Control Layer	Controller hacking, Control signal hacking.	Hacks the controller to perform malicious activities. Interrupts and manipulates the control. signals to perform malicious activities.
Information Layer	Privacy	Steal information from traffic analysis

Iran's nuclear program, particularly the centrifuges used for uranium enrichment. Stuxnet exploited vulnerabilities in Windows operating systems to infiltrate industrial control systems (ICS) used in Iran's Natanz nuclear facility. It propagated via USB drives and exploited zero-day vulnerabilities to gain access to the systems. It targeted the programmable logic controllers (PLCs) responsible for controlling centrifuge speeds. By issuing unauthorized commands, it caused the centrifuges to spin out of control, ultimately damaging around 1,000 of them. This significantly hindered Iran's nuclear enrichment capabilities. It underscored the need for robust cybersecurity measures in critical infrastructure systems.

2. **Ukrainian Power Grid Cyberattack [61]:** In December 2015, Ukraine experienced a widespread power outage affecting multiple regions, which was attributed to a cyberattack. Attackers used sophisticated malware, known as BlackEnergy, to infiltrate the IT networks of Ukrainian power

companies. This malware then targeted the industrial control systems (ICS) used to manage the power grid. The cyberattack resulted in the disruption of electricity supply to hundreds of thousands of customers in Ukraine during the winter season. The attack affected multiple components of the power grid, including substations and distribution systems.

3. **Jeep Cherokee Remote Car Hacking [62]:** In 2015, security researchers Charlie Miller and Chris Valasek demonstrated how they could remotely hack into a Jeep Cherokee's CPS, gaining control over various physical functions of the vehicle. The researchers exploited vulnerabilities in the vehicle's entertainment system, which was connected to the internet via a cellular network. They were able to remotely access the car's CAN bus, which controls critical functions, such as steering, brakes, and acceleration. Through their remote access, the researchers were able to manipulate the vehicle's brakes, steering, and transmission, demonstrating the potential for malicious actors to take control of a vehicle remotely. This raised significant safety concerns regarding the security of modern automobiles. The Jeep Cherokee hacking incident highlighted the importance of securing the physical components of CPS, especially in safety-critical systems like autonomous vehicles.

4. **Shodan IoT Search Engine Exploits [63]:** Shodan is a search engine specifically designed to find internet-connected devices, including those used in CPS, such as industrial control systems (ICS) and IoT devices. Hackers have used Shodan to identify vulnerable CPS devices connected to the internet. By exploiting known vulnerabilities or weak security configurations, attackers gain unauthorized access to the physical components of these systems. Several incidents have been reported where attackers have compromised the physical integrity of CPS devices, such as industrial machinery or critical infrastructure components. For example, attackers have disrupted manufacturing processes, caused equipment malfunctions, or even damaged physical assets. The Shodan exploits highlight the importance of implementing robust security measures to protect the physical components of CPS from unauthorized access and manipulation.

These case studies underscore the significance of securing the CPS components to prevent unauthorized access, manipulation, or damage, therefore,

safeguarding critical infrastructure and ensuring public safety.

3.4.3 Security Measures for CPS

Security measures for CPS are essential to protect these interconnected systems that integrate physical components with computing and communication elements. Indeed, we have to conduct a comprehensive risk assessment to identify potential vulnerabilities and threats to the CPS. This assessment should consider both cyber threats and physical risks. For the above mentioned security threats, this subsection summarizes the corresponding defense measures and detection methods [59].

- **Authentication and Access Control:** Implement strong access control mechanisms to restrict unauthorized access to the CPS. This includes user authentication, authorization, and privilege management. We may use for instance, strong passwords, multi-factor authentication, and role-based access control (RBAC) to enforce access restrictions.

- **Secure Communication Networks:** We have to encrypt communication channels between CPS components to protect data integrity and confidentiality. We can use protocols, such as Transport Layer Security (TLS) or Virtual Private Networks (VPNs) to secure data in transit.

- **Intrusion Detection and Prevention:** To deploy intrusion detection and prevention systems (IDPS), we monitor network traffic, detect anomalies, and block malicious activities. For this, we utilize both signature-based and behavior-based detection techniques to identify potential threats.

- **Network Segmentation:** If we proceed to segment the CPS network into separate zones or subnets, we can isolate critical components and restrict lateral movement of threats. This helps contain potential attacks and limits their impact on the entire system.

- **Patch Management:** Regularly update and patch all software and firmware components of the CPS are used to address known vulnerabilities. Thus, we establish a systematic process for vulnerability management and ensure timely application of patches.

- **Physical Security:** We also have to implement physical security controls to protect the physical components of the CPS. This includes controlling access to physical infrastructure, securing equipment against theft or tampering, and monitoring the physical environment for any anomalies.

We note that securing a CPS is an ongoing process, and it requires a holistic approach that considers, in addition to the measures already cited, other more general maneuvers such as:

- **Incident Response:** To develop an incident response plan to handle security incidents effectively. This will include defined roles and responsibilities, incident detection and reporting procedures, and steps to mitigate and recover from security breaches.
- **Continual Evaluation and Improvement:** Regularly assess the effectiveness of security measures through audits, penetration testing, and security assessments.
- **Security Monitoring and Logging:** Robust monitoring and logging mechanisms are implemented to capture and analyze security events within the CPS.
- **Vendor and Supply Chain Management:** Establish security requirements and conduct due diligence when selecting vendors and suppliers for CPS components.

Moreover, these security measures center around random faults, during design or fabrication stages (before deployment), and not the ones that can be stealthy during the testing stage or even at the earlier stages of runtime operations.

3.5 Future Directions and Emerging Trends

Although CPS have many advantages and are developing fast and are being more widely used, the gap in coordinating engineering innovation with information technology innovation, as the physical meets the digital, can result in immeasurable losses. Therefore, it is important to develop common languages and other commonalities in this plural-disciplinary field.

Particularly, to the CPS engineer, the system of interest includes both cyber and physical elements. Consequently the foundations, methods and tools of CPS engineering should incorporate both the discrete models of computing hardware and software, and the continuous-value formalism of physical (e.g. mechanical, electrical, electronic) engineering.

The development of CPS faces significant technological challenges, which are partially due to the unique characteristics that distinguish CPS from classical systems. In particular, advancements in technology are needed to create distributed, interoperable, autonomous, and reliable systems that can effectively safeguard the safety, privacy, dependability, and cybersecurity of CPS. The industry began exploring the development of interoperable systems with a certain degree of modularity and composability, which could be combined and integrated with existing legacy systems, about a decade ago. However, despite these initial efforts, the deployment, testing, and validation of such systems are still in their early stages of development. In addition, the safety and reliability concerns are still the main barriers in front of the adoption of autonomous systems in different sectors. Addressing the volume of data generated, collected, and processed by CPS presents another technological challenge, necessitating the development of mechanisms to ensure data privacy protection. Additionally, cybersecurity stands out as the most crucial technological challenge that demands prompt attention and effective solutions.

In the following paragraphs, we briefly discuss the research needs of CPS related to this thesis work.

1. **Abstraction and Architectures:** The development of CPS requires innovative approaches to abstraction and architectures, facilitating the seamless integration of control, communication, and computation. These advancements are crucial for the rapid design and deployment of CPS. By enabling heterogeneous systems to be easily composed in a plug-and-play manner, the overall design opens up opportunities for innovation, widespread adoption of technology, and the evolution of the Internet. However, the current science and engineering foundation falls short in supporting routine, efficient, robust, and modular design and development of CPS. Standardized abstractions and architectures are urgently needed to fully support integration and interoperability in CPS (see RQ1 (1.2) in General Introduction Chapter 1).

2. **Distributed Computations and Networked Control:** The design and

implementation of networked control systems present a range of challenges, including time and event-driven computing, software considerations, variable time delays, failures, reconfiguration, and distributed decision support systems. CPS research faces obstacles, such as designing protocols to ensure real-time quality of service guarantees over wireless networks, managing tradeoffs between control law design and real-time implementation complexity, addressing the gap between continuous and discrete-time systems, and ensuring the robustness of large-scale systems. Additionally, there is a need for frameworks, algorithms, methods, and tools to meet the high reliability and security requirements of heterogeneous cooperating components that interact within a complex, interconnected physical environment operating across different spatial and temporal scales (see RQ2 and RQ3 (1.2) in General Introduction Chapter 1).

3. **Verification and Validation:** The hardware and software must be highly dependable, reconfigurable, and, where required, certifiable, from components to fully integrated systems. CPS must possess a trustworthiness that is lacking in many of today's cyber infrastructures. Overdesign is currently the only path to safe and successful system certification and deployment. Yet this approach is rapidly becoming intractable for complex designs and for systems where interoperability is needed. New models, algorithms, methods, and tools are needed that will incorporate verification and validation of software and systems at the control design stage (see RQ4 (1.2) in General Introduction Chapter 1).

3.6 Existing Related Work

CPS are increasingly applied in critical contexts, where they have to support safe and secure operations, often subject to stringent timing requirements. Typical examples are scenarios involving automated living or working spaces in which humans operate, or human-robot collaborations in modern manufacturing. The physical elements follow the principles of the physical world which includes physics, mathematics and mathematical modeling, probability, statistics and stochastic processes, logic, linear algebra and analysis.

The cyber elements in CPS follow the principles of computer engineering and computer science which includes software programming, computational

hardware, processors and embedded computation, and networking.

The control/decision making elements refer to cyber and physical elements that process and monitor incoming information from sensors and commands the actuators to perform various tasks through feedback control loops. The control and decision making elements follow the principles of control theory, adaptive and robust control, distributed and fault-tolerant control, stability, optimization, hybrid systems, digital and real-time systems.

Formal methods, involving the use of mathematical rigor to model and reason about systems, also hold significant advantages when applied to modeling the three constituent elements of a CPS.

In this section, we review some existing approaches using formal methods to modeling and verification of CPS. On the one hand, we highlight, through the first approaches class, the role of formal methods in addressing challenges, such as heterogeneous modeling, behavior analysis and design space exploration. On the other hand, we identify, in a second class of approaches, the most common techniques used to analyze security in CPS.

3.6.1 Formal Modeling Approaches Review

At present, numerous research works focus on different aspects of CPS engineering and emphasize the significance of formal methods. When applying formal methods to a CPS, the objective is to demonstrate that the system progresses and follows a trajectory that meets the specified constraints and requirements. Various types of formal methods, including discrete, automata-theoretic, hybrid systems, and their communicating counterparts, have been utilized to verify that CPS satisfy formally defined requirements [64, 65]. Hence, in Table 3.3, we examine a number of formal approaches that specifically tackle the modeling and analysis aspects of CPS.

The authors in [66] have shown the need for different models types to represent stakeholder requirements, system behavior, and system architecture. They proposed a compositional modeling language for CPS based on wiring diagrams of category theory. This categorical language aims to formalize the relationships between different model views, manages complexity, enable hierarchical decomposition of system models, and proves consistency between models. Authors have mentioned the use of formalism such as first-order logic for requirements, difference and differential equations for physical behavior, and graphs for architectures. They highlight the potential of category theory to unify diverse views of system models and improve scalability.

Another formal modeling method for physical entities in CPS has been presented in [67]. It extended the traditional Timed Petri net by introducing spatial factors to describe the logical, time-level behavior of physical entities, as well as changes in state caused by their position changes. The paper contributed to the understanding of CPS and to analyze and verify their characteristics.

Tan, Y. et al. [68] have proposed a novel spatio-temporal event model that addresses the characteristics and requirements of CPS, providing a framework for analyzing and representing events in these systems. The event model represents events as a function of attribute-based, temporal, and spatial event conditions. It utilizes logical operators to combine different types of event conditions and to capture composite events.

There was also some interesting works [69, 70, 71] based on BRS formalism, which is stated in our work. Authors in [69] have introduced the concept of topology configuration that captures the environment characteristics of cyber-physical spaces. The topology configuration provides contextual information for the access control system, such as the location of digital files, the access state between subjects and objects, and the proximity relationship between subjects. Based on this topology configuration, a formal inter-domain access control model, called TA-CPAC, adjusts permission assignments adaptively to react to the changing behaviors of subjects and objects. It considers both physical security and cyber security requirements, as well as securing the interaction between the physical and cyber spaces.

The paper [70] continued to use the TA-CPAC model and provided a systematic solution for specifying security policies in a cyber-physical space and ensuring the satisfaction of security requirements by considering the dynamic topology of the environment. It introduced a reduction algorithm to simplify the modeling process and improve the efficiency of model checking.

Authors of [71] proposed to formalize CPS by the use of an integrated formal analysis, combining the BPMN model and Bigraphs. They tried to show how these models complement each other to assist the system designer in establishing formal verification of the business process work-flows involved in CPS. The proposed integrated approach allows, according to different dimensions: functional, organizational and behavioral, to give precise semantics for the considered business process, with the aim of improving their efficiency, adapting them to new technologies and possible extensions and thus gaining a competitive advantage for the CPS-based organization they model. Obviously, formal methods play a vital role in addressing challenges related

to heterogeneous modeling, behavior analysis, and design space exploration in CPS, they provide techniques to ensure their correctness and reliability. However, the formal models given in the approaches cited above, turn out to be too abstract and can neglect important details to capture the behavior and essential properties of the CPS.

In our previous work [72], we showed that BRS extended with specific intelligent nodes (Control Agents) determine the appropriate level of abstraction and granularity, because overly abstract models, such as BRS, may overlook important details, while the agent-based models may capture critical aspects accurately. Besides, they allow to model and reason about the dynamic and uncertain behavior of CPS. In the present work, we continue to use this formalism (CA-BRS) and we show its advantages in terms of modeling the interactions between the CPS heterogeneous constituents. An iterative refinement process between the CA-BRS and BPMN offers designers the gain insight into the behavior and performance of the CPS, under various design choices.

3.6.2 Review of Security Analysis Techniques

The security issues and challenges faced by CPS are presented in [73, 74]. As the integration of the cyber and physical processes in CPS is becoming increasingly closer, CPS may be attacked from the cyber domain, resulting in a series of consequences, such as hardware damage or certain failures. In recent years, some researchers have studied the security issues of CPS [75]. In Table 3.4, we summarize some existing related work and highlight the concerned security issues.

A study in [76][29] systematizes existing research on CPS security under a unified framework. The framework consists of three orthogonal coordinates: 1) from the security perspective, the well-known taxonomy of threats, vulnerabilities, attacks and controls; 2) from the CPS components perspective, the focus is on cyber, physical, and cyberphysical components; and 3) from the CPS systems perspective, the general CPS features as well as representative systems (e.g., smart grids, medical CPS, and smart cars). Authors claim that hopefully by providing a novel CPS security architecture this paper will contribute in CPS security researches and help them to target their effort efficiently.

Lu et al. [76] proposed CPS security theories, which contain security properties and basic theories. Security properties include safety, security,

Table 3.3: CPS Formal Modeling: Existing Approaches

References	**Formalism**	**CPS Architecture (Physical/Cyber Entities)**	**CPS Behavior (Physical/Cyber Entities)**	**Interaction Modeling**	**Requirements/ Properties**
[35]	Category theory	With Graphs	With Differential equations	X	With First Order Logic
[36]	Timed Petri Nets	X	Timed behavior of Physical Entities	X	X
[37]	Event model	X	Spacio-Temporal event Model	X	√
[38]	BRS	Topology configuration	TA-CPAC	√	Security property
[39]	BRS, Model checking algorithm	Dynamic topology	TA-CPAC	√	Security policies
[40]	BRS, BPMN	With Bigraphs	Reaction Rules	X	BPMN simulation
[41]	CA-BRS	With Bigraphs	Control Reaction Rules	√	BPMN simulation

Legend:
X: not supported
√ : supported
TA-CPAC: A topology-aware access control model
BRS: Bigrapical Reactive Systems
BPMN: Business Process Modeling Notation
CA: Control Agents

reliability and resilience. Basic theories consist of information theory, control theory and game theory.

Dibaji et al. [77] reviewed the security of CPS from the perspective of system and control. A unified threat assessment metric was proposed in order to evaluate how CPS security is achieved in each of these three cases: prevention, resilience, and detection & isolation.

Different CPS security objectives are discussed in [78]. Authors brought together models, schemes and implementation aspects, regarding the alternative technologies and devices of IoT. Their works focussed on the knowledge of the implementation efficiency, basically in hardware aspects, of communications confidentiality, user authentication, data integrity and services availability. Attacks and modern threats, as well as to countermeasures against them, were considered. Authors of this study [79] presented the results of a systematic mapping study on the existing model-based security engi-

neering studies for cyber-physical systems (MBSE4CPS). The results could shed some light on an emerging research area, which is interdisci- plinary among research domains such as system engineering, soft- ware engineering, and security engineering. More specifically, this study was designed and conducted based on a rigorous SMS pro- tocol for identifying a set of primary MBSE4CPS studies.

EC 61850, an international standard for communication networks, is becoming prevalent in the CPS environment, especially with regard to the electrical grid. Authors of [73] examined security vulnerabilities, security requirements, and security architecture for a CPS in which IEC 61850-based heterogeneous protocols are connected. On the other hand, in [74] the impact of cyber threats on authenticity, confidentiality, reliability, resilience, and integrity, were considered. This was reflected as a tree of attacks and threats on sensor devices, actuators, computing components, communications, and feedback. The paper presented a review of relevant literature on the discussion of practical applications in the areas of SCADA and Smart Grid security, countermeasures against cyber-attacks and communication security and the dominant areas of research

In this paper [80], authors highlight challenges and some missing pieces in CPS security research, and hope to stimulate more interests in the research community. They pretend that t traditional IT security measures, new generation IT security solutions or security solutions specific to control systems will be insufficient alone, and the need for integrated security solutions that include all components in the CPS network and include operational scenarios. Although some defense mechanisms have been proposed/deployed, new and systems specific solutions are still expected in response to the newly identified threats and vulnerabilities. In addition, these anticipated new security solutions should have the feature of preventing the destructive and irreversible damage that may occur in CPS networks

This study [81] presents a comprehensive threat modeling framework for CPS using STRIDE, a systematic approach for ensuring system security at the component level. t STRIDE is a light-weight and effective threat modeling methodology for CPS that simplifies the task for security analysts to identify vulnerabilities and plan appropriate component level security measures at the system design stage. The primary contribution of this study is to formalize a systematic methodology that can be used for effective characterization of system-specific threat using the STRIDE approach.

In [82], authors described a pattern that assists software developers in

creating an architecture which captures the relevant elements for a security analysis. The interfaces of components may not only be accessible for authorized entities, but also for attackers. Therefore, they specified different interface types which enables one to identify relevant attacks for a specific interface type. First, the solution part of the pattern is given as a meta-model, then some guidelines for its instantiation are provided. As an example, we instantiate the pattern for a typical automation and control system. Last, we evaluate the suitability of our pattern by discussing how typical threats could be mapped to the different interface types.

Authors in [83] have shown how model-based requirements can be used to model effectively security requirements for secure CPS. By semantically extending behavioral and physical model elements and diagrams of SysML, they were able to build a problem space modeling framework that aligns with the foundations of systems theory. Their main contribution consists in modeling requirements as required transformations of inputs into outputs by the system of interest, the resulting model-based requirements do not (and in fact cannot) prescribe any particular system solution. Therefore, the model-based requirements presented in this paper protect the requirements author/modeler against unnecessarily over constraining the solution space.

From these insightful research studies, several research gaps can be identified, including the lack of CPS threat categorization and the absence of theoretical analysis approaches for detecting CPS threats that integrate different security dimensions (Physical, Cyber, or Network).

Indeed, the above brief taxonomy highlights several research gaps, which we hope could be useful in tackling our research in the area of CPS security. Without theoretical frameworks that consider all security dimensions, researchers and practitioners may fail to account for the interdependencies between physical, cyber, and network domains. This omission can result in an incomplete understanding of threats and their potential impact. In addition, integrated theoretical approaches are essential for designing resilient CPSs capable of identifying, mitigating, and responding to threats across all dimensions. This is crucial for ensuring the safety, reliability, and continuity of critical CPS applications, such as smart grids, industrial automation, and autonomous systems.

3.7 Conclusion

CPS are considered by the European H2020 research agenda "the next generation of embedded Information and Communication Technology (ICT) systems that are interconnected and collaborating, providing citizens and businesses with a wide range of innovative applications and services" [84]. In this chapter, we have provided a synthesis of the current state of knowledge in CPS definitions and security, highlighting the main challenges, trends, and potential solutions identified in the literature. We have also offered some recommendations for future research directions and have emphasized the importance of addressing the security concerns in CPS design and deployment. Moreover, we have underscored the deficiencies in the current research and proposed specific domains that require further exploration in order to strengthen the security of CPS systems. In the upcoming chapter, we will present our proposed solution approach to tackle this challenge.

Table 3.4: CPS Security Analysis Techniques Review

References	Used Means	Security Issues	CPS Design/Deployment Phase	Case Studies/Domains
{29]	Information theory, Control theory, Game theory	Security, Resilience, Safety and Reliability.	No, but a survey is given	General
[30]	Control method	Prevention, Resilience, and Detection & Isolation.	No, but a survey is given	Power and Transportation Applications
[31]	Control-theoretic and cyber security Approaches	DoS, Replay attacks, Authentification	No, but a survey is given	Power grid, Distributed network, Control system, Sensor network, Protocol vulnerability, Medical
[32]	Model-Based Security Engineering	Confidential-ity, integrity, availability	Design and a Survey	Industrial applications
[33]	EC 61850-based heterogeneous protocols	Vulnerabilities in Network, Gateway system, Individual protocol	Architecture security for Deployment	Electrical grid in Korea
[34]	--	Attacks in the cyberspace	Architecture security and a Survey	SCADA, Smart Grid security
[35]	Integrated security solutions	Physical and Cyber Attacks: middle attacks, DoS Attacks	No, but a survey is given	Industrial Control System
[36]	STRIDE methodology (Spoofing, Tampering, Repudiation, Information, Denial of Service, Elevation of privilege)	Treats of components	Treat modeling: Design and Deployment	Synchronous Islanding Testbed
[37]	Software Pattern	Typical Treats	Architecture: Design	A typical automation and control system
[38]	SySML language	CPS Requirements: authentication	Design	Authentification system

Chapter 4

Software Architecture of Cyber Physical Systems

4.1 Introduction

The development of large and heterogeneous systems as CPS requires the implementation of complex engineering processes that go so far as to involve several thousand engineers from many different specialties.

The design and implementation of these complex systems is not without major conceptual and technical difficulties because it is increasingly more difficult (or sometimes impossible) for an individual to fully understand such systems as a whole. Addressing these challenges requires advances in modeling languages, formal methods, and tools specifically tailored for CPS. It also requires interdisciplinary collaboration between domain experts, formal methods researchers, and tool developers to develop effective modeling and analysis techniques for CPS.

This chapter primarily addresses the abstraction and granularity aspects of modeling CPS. Achieving a balance between abstraction and granularity is often necessary when modeling CPS. The models should capture the fundamental behavior and properties of the system, while also providing sufficient detail to accurately represent critical aspects. Finding the right level of abstraction and granularity is a challenging task because overly abstract models may overlook important details, whereas overly detailed models can become unwieldy and difficult to analyze.

To accomplish this objective, we propose a new methodology for design-

ing, defining, and reasoning about the dynamic and secure behavior of CPS. We also take into account the existing gap that affects the practical use of formal methods in CPS development. This chapter specifically focuses on the initial phase of our approach, while the subsequent chapters of this thesis will present the remaining phases.

The remainder of this chapter is organized as follows. In Section 4.2, we present an iterative design process that comprises three distinct phases: the formal specification phase of the CPS (deemed the most crucial), the architectural design phase of software and the implementation phase. Section 4.3 is dedicated to present in detail the IEEE 42010 standard for Architecture Description of Software-Intensive Systems. Section 4.4 presents first the motivation for adopting this standard in the design of CPS architecture. Then, it proposes a step-by-step guide on how we can use IEEE 42010 in the CPS architecture design process.

4.2 Our Approach Principle

The formal modeling of CPS faces several challenges due to the complexity and unique characteristics of these systems. Some of the key challenges include:

Ch1 Complexity: CPS are inherently complex due to the integration of physical components, computational elements, and communication networks. Modeling the behavior, interactions, and dependencies of these heterogeneous components accurately is a significant challenge.

Ch2 Heterogeneity: Modeling and representing the heterogeneity in CPS accurately is a challenge. It requires capturing the diverse characteristics of physical components, such as sensors, actuators, and control systems, along with the computational elements, such as software algorithms and communication protocols.

Ch3 Abstraction and Granularity: Determining the appropriate level of abstraction and granularity is challenging, as overly abstract models may overlook important details, while overly detailed models may become unwieldy and difficult to analyze.

Ch4 Uncertainty and Dynamic Behavior: The behavior of physical components, communication channels, and external factors can be subject

to variability and uncertainty. Modeling and reasoning about this dynamic behavior and uncertainty is challenging. Formal methods need to account for uncertainty, variability, and the ability to adapt to changing conditions.

Ch5 Scalability: CPS can range from small-scale systems to large-scale infrastructures. Ensuring that formal models can scale to handle the complexity and size of CPS is a significant challenge. Techniques for model decomposition, abstraction, and compositional analysis are necessary to manage the scalability of formal models.

Ch6 Validation and Verification: Validating and verifying formal models of CPS against real-world systems is a challenge. It is often difficult to ensure that the models capture all relevant aspects and faithfully represent the behavior of the physical system. Additionally, verifying complex and large-scale CPS models can be computationally demanding and time-consuming.

Ch7 Tool Support and Standardization: Effective use of formal methods for modeling CPS requires suitable tools and standardized modeling languages. However, the availability and maturity of tools and languages specific to CPS modeling can be limited. Furthermore, the lack of standardization in CPS modeling languages and notations makes it challenging to share models, collaborate, and ensure interoperability between different modeling and analysis tools.

Overcoming all these challenges (Ch1-Ch7) necessitates advancements in modeling languages, formal methods, and tools specifically designed for CPS. Our proposed solution approach revolves around a suitable formal model, referred to as CA-BRS, which extends BRS with Control Agents and introduces a new computational model called Guided Transition Systems. This approach enables the analysis of correctness, security, and safety aspects of CPS, while also supporting the accurate design and implementation of this new class of engineered systems. Additionally, the utilization of another standard notation, BPMN, is anticipated to bring it closer to the implementation and development of future CPS.

In this section, we present the principle of our approach, which consists of a multi-phase and iterative process to design and analyze CPS of various domains. The main parts of our proposed process are: modeling, design, and

analysis. As depicted in Figure 4.1, these parts overlap and are detailed in the following sections.

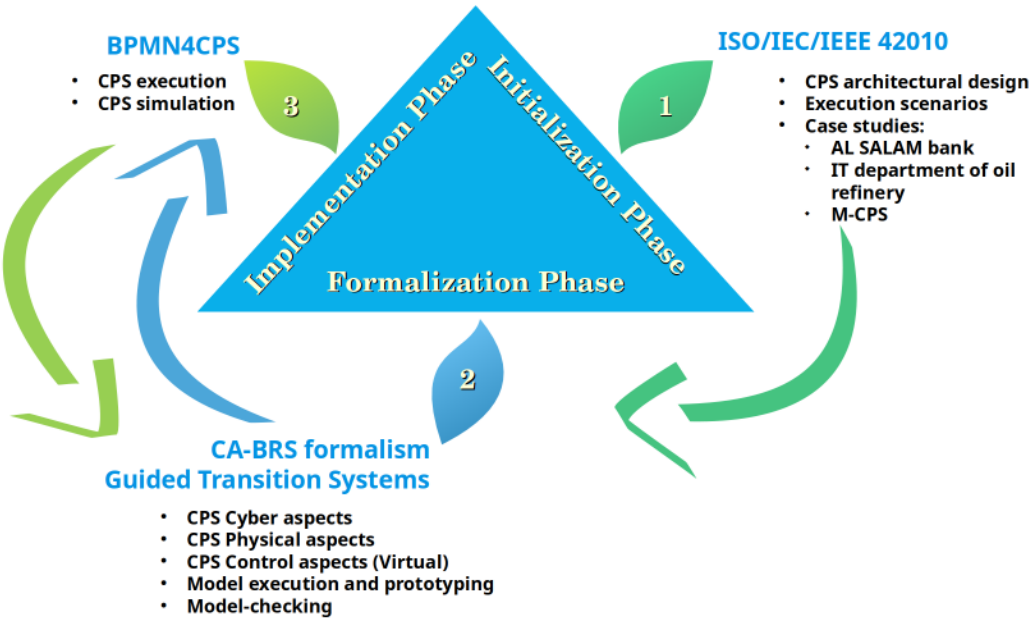


Figure 4.1: The Proposed Approach Phases

4.2.1 Software Architectural Design (Phase 1)

Our given process will begin by initializing the design phase where the goal is to understand the problem, and how a given CPS will evolve and adapt its behavior to preserve some properties while giving some execution scenarios of the considered system. Our contribution in this step offers a comprehensive architectural description for CPS that enhances communication, analysis, and decision-making throughout the entire system lifecycle. What sets our work apart is the innovative aspect of presenting this architecture as a meta model, which encourages the adoption of architectural views and viewpoints, in accordance with the principles and guidelines outlined in ISO/IEC/IEEE 42010. Undoubtedly, this outcome addresses the limitations Ch1, Ch2 and Ch3 mentioned previously.

IEEE 42010 provides a common language and framework for describing system architectures. This helps in achieving consistency in communication

among stakeholders with diverse backgrounds and roles, fostering a shared understanding of the system. The standard was initially published in 2011, and its development was driven by the Institute of Electrical and Electronics Engineers (IEEE) with collaboration from the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). Section 4.3 below serves as a valuable resource for software system architects and developers, offering a standardized approach for describing the architecture of CPS.

4.2.2 CPS Formal Specification (Phase 2)

The proposed process may progress to the formal modeling phase, where we give a sound interpretation to all already identified components and their relationships. The objective here is to gain a better understanding on how to model and manage the physical, the cyber and the virtual entities of CPS in one hand, and their control and interacting during the adaptation of the CPS behavior in response to unanticipated changes on the other hand.

As no single formalism can be used to model all these CPS aspects, we choose to use a new formalism CA-BRS that combines BRS and Control Agents. This BRS extension achieves a clear separation between the virtual level, described with abstract agents; the physical and the cyber levels (execution machines and their environment, software applications, etc.); both specified using bigraphs; and the behavioral level (dynamics) of the CPS. The latter level is represented using an extended form of reaction rules supporting two types of evolution over time; physical, which corresponds to a sequence of bigraphs, and virtual conducted by the agents move and migrate (cyber mobility), while considering some material constraints through the agents observations also represented via rewriting rules.

In this particular stage of our approach, our primary objective is to overcome the limitations Ch3, Ch4, and Ch5, already cited. By doing so, we aim to effectively bridge the identified gaps. Sufficient details about this phase will be given in Chapter 6.

4.2.3 Implementation and execution (Phase 3)

In the last phase of the suggested process, and before implementing the actual CPS, we translate the main features of the CPS model to BPMN one. Indeed, BPMN constitutes a modeling standard for business processes with accepted

semantics, facilitating the interaction between a system engineer and a system modeler. It serves as a standardized bridge for the gap between the CPS formal specification (CA-BRS model) and their implementation. Then, we apply successive BRS-BPMN based transformations in order to capture and analyze run-time behavior of CPS based on its execution traces. An initial BPMN model execution may reveal limits in the formal models, causing a return to the modeling phase and revision of the models. Details concerning this phase will be presented in Chapter 7. Therefore, in a distinct manner from the conventional implementation of CPS, we successfully incorporate and address the challenges: Ch5, Ch6, and Ch7.

4.3 IEEE 42010 Standard Description

Software architecture is a critical aspect of the software development process, providing a strategic foundation for building robust, scalable, and maintainable software systems. Effective software architecture enables teams to manage complexity, facilitate communication among stakeholders, and deliver software that meets both functional and non-functional requirements. It encompasses a set of decisions and patterns that define the organization, relationships, and interactions among the various components or modules of a software application. In this discipline context, the IEEE Systems and Software Engineering Standards Committee (S2ESC) has developed the IEEE 42010 standard (revision: 2011), emerging from the two previous standards: IEEE 1471, which focused on architecture description, and ISO/IEC 42010 (edition: 2007) [85, 86], which addressed system and software engineering documentation.

The key point of this standard is to give the essential concepts and definition and methods of creating architecture description with intent of facilitating the task of creating system architecture and also, the mechanisms of analyzing and sustaining this architecture. To achieve this task the standard gives a conceptual model of architecture description, that includes architecture viewpoints, architecture frameworks and architecture description languages [6].

The scope of this standard is the specification of how system architecture description needs to be expressed and organized along side its main concepts (architecture viewpoints, architecture frameworks and architecture description languages). So, it focuses on presenting guidance on specifying the

description and its main constituents, in other word: "what is needed, not how is done". The effort here is to give a common ground to describe system architecture while still compatible with other standards [6].

4.3.1 History

It is important to note that the standard is not appeared from nowhere. It is rather a build up of works (as in [87]), methods (as in [88, 89]) and also an evolution of existing standards [90], [6]. We indicate in what follows the main versions (For more details please refer to [91]):

2007 (IEEE 1471-2000™) The base version of this standard is the "IEEE Recommended Practice for Architectural Description for Software-Intensive Systems" [92], it was adopted by ISO in 2006 [85]. The participant "Rich Hilliard" has mentioned in [85], [93] the work that needs to be done, through the Working Group 42 (WG42), the Architecture of ISO/IEC JTC1/SC7 (joint technical committee, Subcommittee [6]), the Software and Systems Engineering, in order to establish (revision) the new standard, is mentioned here as:

- Introducing the notion of architecture frameworks and architecture language description.
- Defining the viewpoint and other concepts in the standard.
- Creating a website [91] to assemble the different viewpoints established by many volunteers ("viewpoint repository" [94]) and providing a good reference of the standard to the interstate.

2011 (ISO 1st edition) The first edition of the concerned standard has been released in 2011 [95]. ISO/IEC/IEEE 42010 was prepared by Joint Technical Committee ISO/IEC JTC 1, Information technology, Subcommittee SC 7, Software and systems engineering, in cooperation with the Software and Systems Engineering Standards Committee of the Computer Society of the IEEE, under the Partner Standards Development Organization cooperation agreement between ISO and IEEE. This first edition of ISO/IEC/IEEE 42010 cancels and replaces ISO/IEC 42010:2007, which has been technically revised [6].

2022 (ISO 2nd edition) [91] The first revision (2nd edition) of the standard has been released in 2022, the new elements are colored (in pink) in figure 4.2 [5].

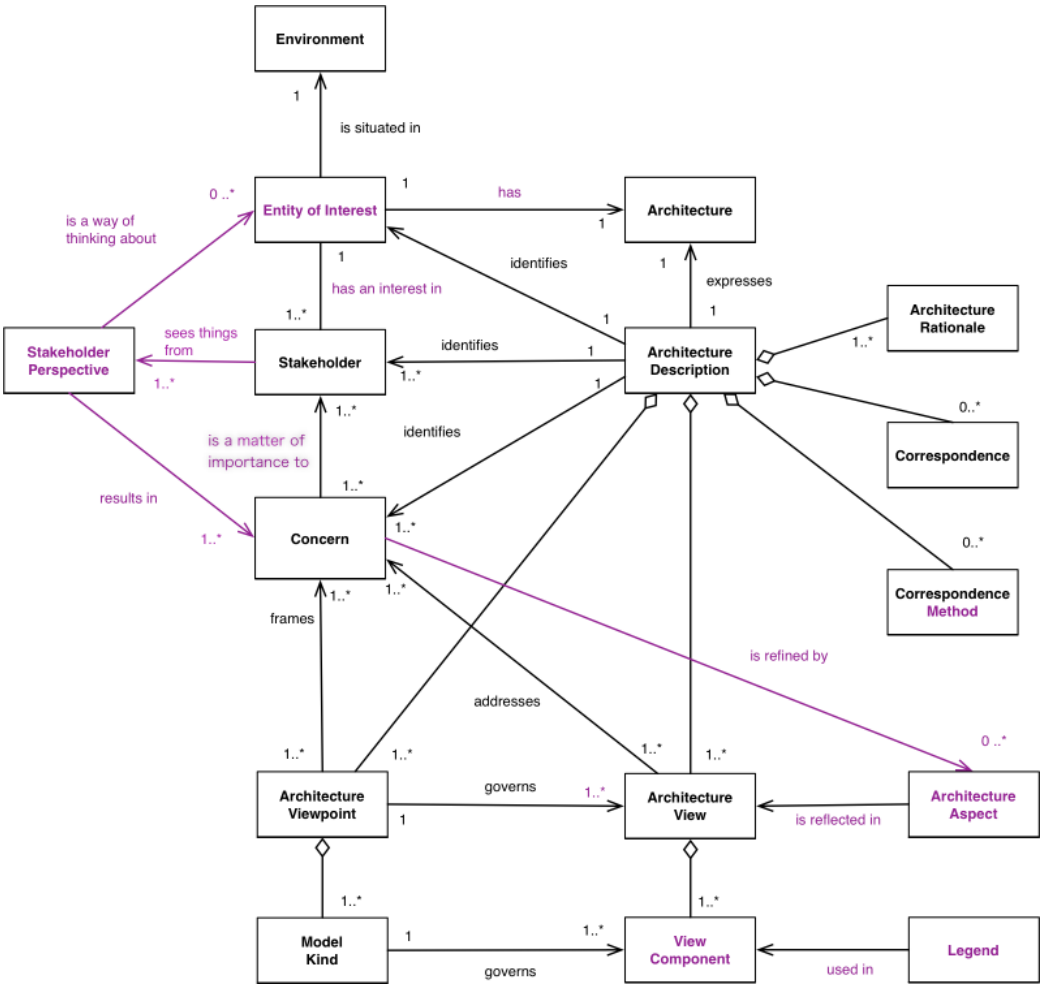


Figure 4.2: ISO/IEC/IEEE 42010 Conceptual Model 2022 Version [5]

4.3.2 Key Concepts

In this subsection, we briefly explain the important concepts of the standard IEEE 42010. The Standard provides six (UML) diagrams which represent its

conceptual model [6]. These diagrams emphasize that the system is actually influenced by its environment, the "stakeholders" who have interests in the system represented by "concerns", the "viewpoint" frames those concerns with the help of specified mechanisms (language, diagrams, methodologies) called "model kinds" which govern models that constitute the resulted views. To keep the consistency and conformity between models and architecture description (AD) elements the "correspondence rules" are used, the standard also urges keeping track of the rationales behind the taken architecture decisions. Besides, in order to simplify the development task, it incorporates the concept of architecture frameworks and architecture description languages as a group of predefined viewpoints and model kinds respectively [6]. The main elements of this standard are explained below through some diagrams:

Environment (Context)

An environment as indicated in Figure 4.3 is a "context" determining the setting and circumstances of all influences upon a system [6].

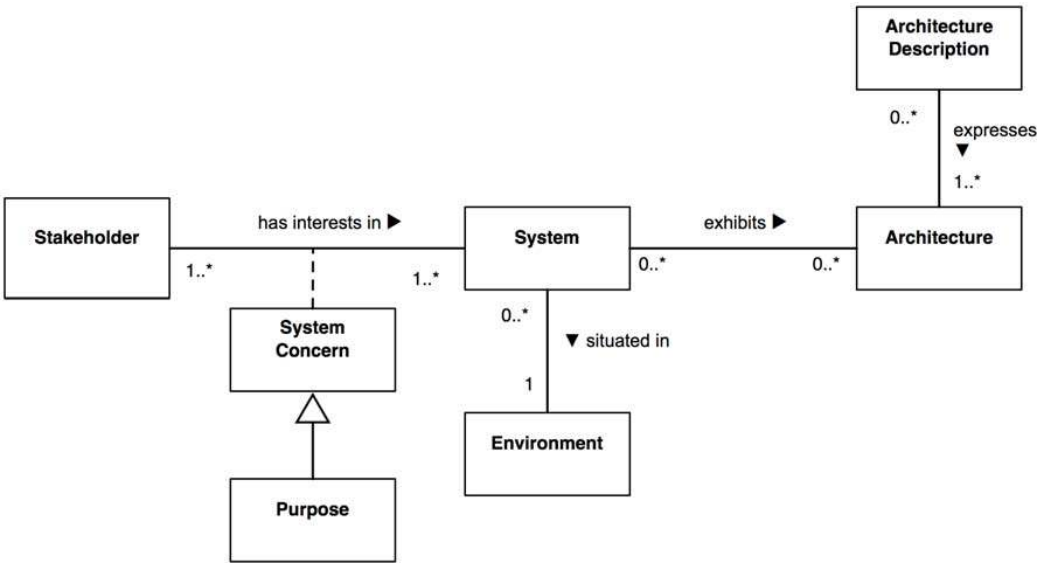


Figure 4.3: Context of architecture description [6]

Stakeholders & concerns

A stakeholder is an "individual, team, organization, or classes thereof, having an interest in a system" [6]. Potential stakeholders may be users, operators, acquirers, owners, suppliers, developers, builders, maintainers of a system [94]. A concern is an "interest in a system relevant to one or more of its stakeholders" [6]. Concerns can be either general or specific, such as: reliability, network latency handling [94]. Examples of concerns may be (system) purpose, functionality, structure, behavior, cost, supportability, safety, interoperability, security [5].

Views & models

A view is a "work product expressing the architecture of a system from the perspective of specific system concerns" [6]. A viewpoint is a way of looking at systems; a view is the result of applying a viewpoint to a particular system-of-interest [6]. Examples of viewpoints: operational, systems, technical, (logical, deployment [85]), process, information [5]. A model kind is a "conventions for a type of modelling" [6]. An architecture model is a work product; its subject is determined by its model kind [6].

Correspondences & rationale

An AD element is any construct in an architecture description [6] as shown in Figure 4.4. A correspondence defines a relation between AD elements, Correspondences and correspondence rules are used to express and enforce architecture relations such as composition, architecture rationale records explanation, justification or reasoning about architecture decisions that have been made [6]. The rationale for a decision can include the basis for a decision, alternatives [6] (see diagram of Figure 4.5).

Frameworks & Architecture Description Languages

An architecture framework as represented by its class in Figure 4.6 establishes a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular domain of application or stakeholder community. In another hand, an architecture description language (ADL) is any form of expression for use in architecture descriptions (see Figure 4.7).

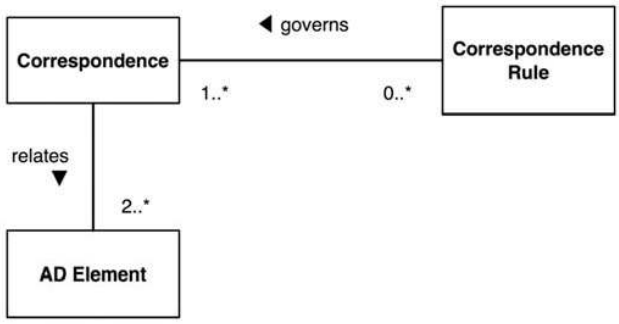


Figure 4.4: Conceptual model of AD elements and correspondences [6]

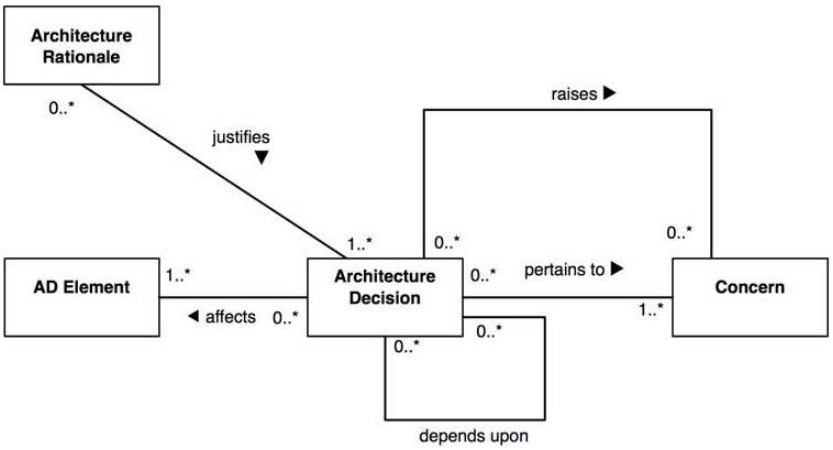


Figure 4.5: Conceptual model of architecture decisions and rationale [6]

We note that Rich Hilliard [93] provides two templates ready to fill, downloadable from [91]: "Architecture Viewpoint Template for ISO/IEC/IEEE 42010" [94], "Architecture description template for use with ISO/IEC/IEEE 42010:2011" [96].

4.3.3 Use Cases

To conclude this section, we should highlight some existing and relevant works that have been conducted according to the IEEE 42010 standard. These works are particularly relevant to our research as they provide valuable insights and established practices that align with the principles and guidelines of this standard. By examining these contributions, we can better

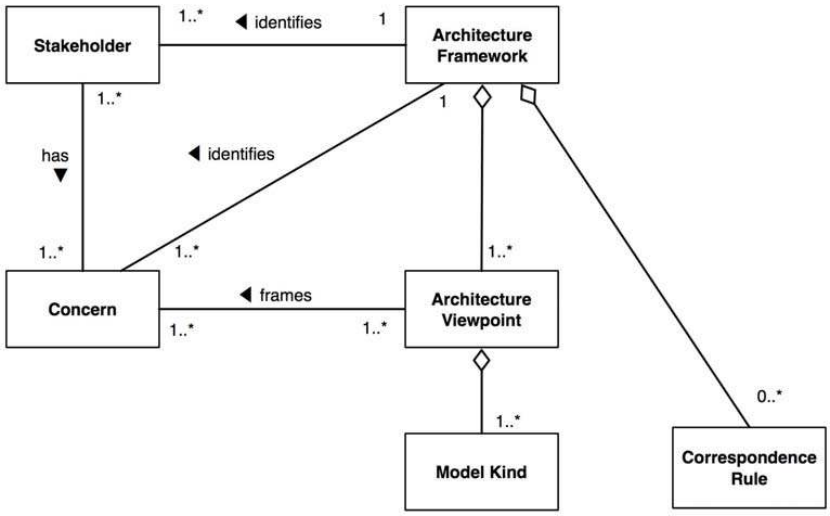


Figure 4.6: Conceptual model of an architecture framework [6]

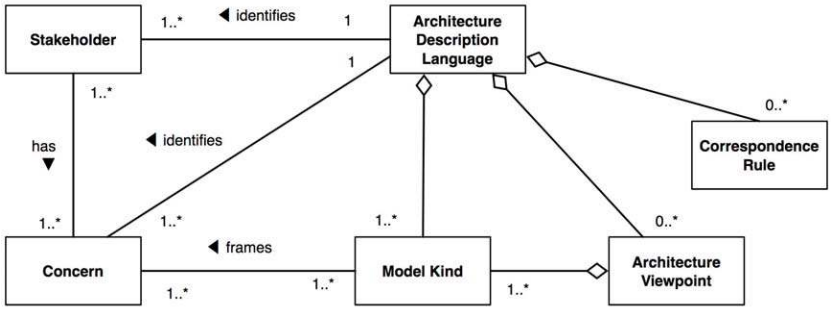


Figure 4.7: Conceptual model of an architecture description language [6]

situate our approach within the broader context of the field, validate our methodologies, and identify areas where our work extends or complements existing efforts. This not only reinforces the credibility of our approach but also emphasizes the significance of the IEEE 42010 standard in advancing architectural practices.

- In [7] and in the frame of "SICHTEN 4.0" project, authors have proposed an engineering method to create a viewpoints (conform to the ISO/IEC/IEEE 42010:2011 standard) of CPS, based on the Reference Architecture Model Industry 4.0 [97] and Semantic Web standards (see Figure 4.8). The goal is to use generative development (dynamically or

on demand code) to create CPS instances.

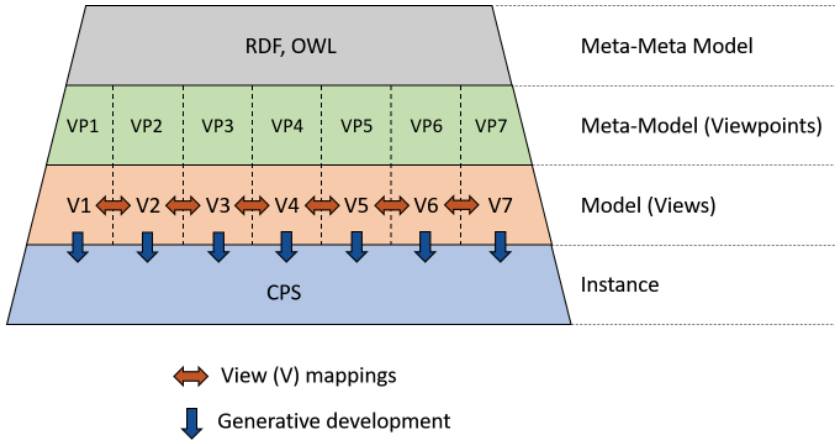


Figure 4.8: Modeling levels involved in engineering CPS [7]

- In PhD work of [8], the author elaborated an architecture, called MAS4SG (illustrated by Figure 4.9), which is focused on smart grids (an example of CPS). He used multi agent system approach (MAS) complying with SGAM framework, the ISO/IEC/IEEE 42010:2011 standard and based model driven architecture (MDA).

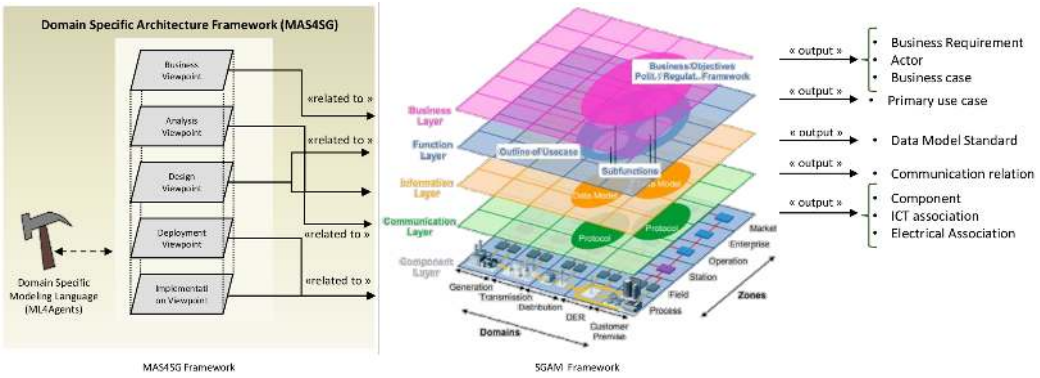


Figure 4.9: MAS4SG's viewpoints related to the SGAM's viewpoint output [8]

- In [9], the authors suggested a framework to document architecture decisions relying on ISO/IEC/IEEE 42010:2011 standard. This work

belongs to architectural knowledge management (AKM). It consists of four viewpoints: decision detail viewpoint, decision relationship viewpoint (as indicated in Figure 4.10), decision chronology viewpoint, decision stakeholder involvement viewpoint.

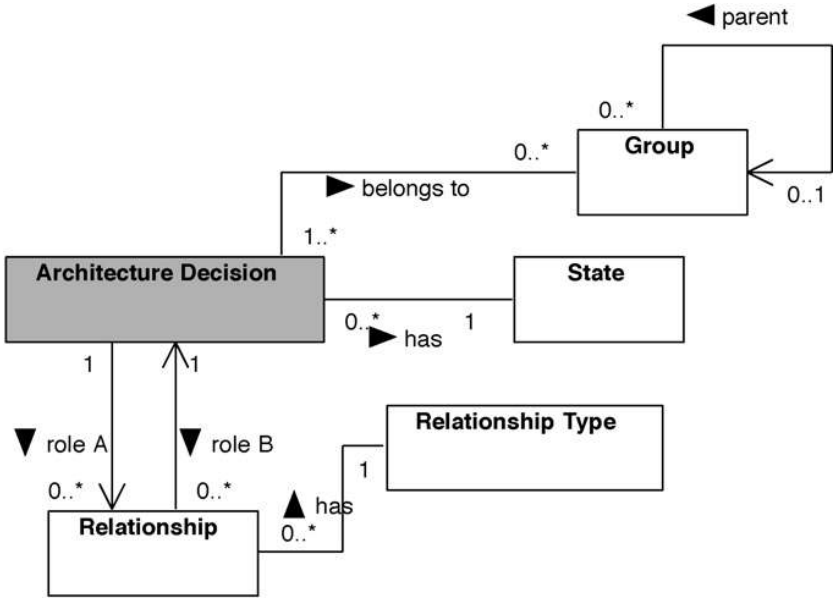


Figure 4.10: Meta model of Decision Relationship viewpoint [9]

4.4 CPS Architecture According to IEEE 42010

The IEEE Standard is dedicated to provide a framework for the description and evaluation of Software-Intensive Systems architectures. In this work, we will adapt it to deal with CPS. In fact, CPS rely heavily on software to control and monitor physical processes (or components), making the software an integral part of the overall functionality. This software often includes algorithms for real-time control, communication protocols, data processing, and decision-making.

4.4.1 Motivation

The motivation for using the ISO/IEC/IEEE 42010:2011 standard in the design of CPS architecture lies in several key benefits:

1. **Consistency and Common Understanding:** IEEE 42010 provides a common language and framework for describing CPS architectures. This helps achieve consistency in communication among stakeholders with diverse backgrounds and roles, fostering a shared understanding of the CPS. So, it provides a common structure for architectural documentation.

2. **Traceability:** The standard supports traceability by linking architectural decisions to stakeholders' concerns and requirements.

3. **Risk Mitigation:** By employing a standardized approach to architecture description, potential risks and issues can be identified and addressed early in the design process.

4. **Interoperability:** The standard promotes interoperability by providing a consistent way to describe and document CPS architectures. This is particularly important in large-scale projects where multiple teams or organizations may be involved, ensuring that the various components of the CPS can work seamlessly together.

5. **Support for Multiple Views:** IEEE 42010 allows the creation of multiple views of the CPS architecture to address the concerns of different stakeholders. The architecture can be understood from various perspectives, such as performance, security, and maintainability.

6. **Facilitation of System Evolution:** As CPS evolve over time, the standard helps to manage changes to their architectures by providing a structured way to document and understand modifications.

7. **Compliance and Quality Assurance:** Adhering to a recognized standard like IEEE 42010 can be essential for compliance with industry regulations and quality assurance processes. Provides a benchmark against which the system architecture can be evaluated and validated.

4.4.2 CPS Architecture Design Process

The development, testing, and maintenance of the software are significant challenges in the design of CPS. Using the IEEE 42010 standard to design their architecture involves following the principles and guidelines described in the standard, while adapting them to these specific systems. In the following

we propose a step-by-step guide on how we use IEEE 42010 in the CPS architecture design process to create a well-documented architecture. The result is given by the conceptual model of Figure 4.11).

1. **Understand Stakeholder Concerns:** Identify and understand the concerns of various stakeholders. Stakeholders may include users, customers, developers, testers, and others who are interested by the given CPS.

2. **Define Architectural Views:** Define the architectural views that need to be created to address the identified concerns. Views are representations of the CPS architecture from different perspectives (e.g., functional, structural, operational, etc.).

3. **Document Architecture Descriptions:** Document the architecture descriptions for each view. Clearly articulate the architectural decisions, rationale, and considerations.

4. **Establish Relationships:** Establish relationships between different architectural elements, decisions, and stakeholder concerns.

5. **Create and Review Views:** The views, that provide a clear representation of the CPS from different angles should be reviewed with stakeholders to ensure alignment with their concerns and expectations.

6. **Iterate and Refine:** Architecture design is an iterative process. Use feedback from stakeholders to refine and improve CPS architectural descriptions and views. Consider alternative design options and evaluate their impact on the system.

7. **Consider Quality Attributes:** Address quality attributes such as performance, reliability, security, and maintainability in the CPS architecture.

8. **Evolve the Architecture:** Update the architecture documentation to reflect any changes or evolution in the CPS, while Ensuring that the architecture remains aligned with the evolving requirements and stakeholder concerns.

9. **Use Tool Support:** Leverage tools that support the documentation and management of architectural descriptions.

It is worth noting that this conceptual model, designed for various purposes, enables us to incrementally derive the following meta-model for CPS (see Figure 4.12). Indeed, Meta-modeling has emerged as a promising approach to tackle the intricacies of CPS. It offers a powerful framework for abstracting and representing CPS models, capturing their essential characteristics and behavior while simplifying the underlying complexity. The ISO/IEC/IEEE 42010 standard is utilized to provide a systematic and structured approach to capturing and communicating the CPS architecture thanks to this meta model. It helps in defining the components, relationships, and interactions within this complex system, as well as the environmental contexts and design decisions.

Figure 4.12 describes our proposed meta model to represent and analyze CPS components. As it is illustrated by the various UML abstract classes of this model, CPS refers to the integration of physical elements and computational (cyber) components, which are tightly interconnected and mutually dependent (Communication UML classes). These systems involve a combination of IoT devices and SBM-SBC (Single-Board Micro controller/Computer) physical elements and software applications, all working together (thanks to Transmission medium) to achieve specific objectives. A classic example of CPS, specifically an architecture of a medical CPS (M-CPS), will be considered in the following chapters while adhering to this meta-model.

4.5 Conclusion

The close coupling of software and hardware in CPS allows for sophisticated control and automation of physical processes, making them more efficient and responsive. In this chapter, we have shown the necessity to conceive a multi-layer architectures for CPS, especially guided by viewpoints which we think is the most appropriate to our kind of systems (CPS). In addition to ensure compatibility with business and industrial aspects (I4.0), our choice has rested on the ISO/IEC/IEEE 42010:2011 standard. We have then recalled the most important keys of this standard and adapted it to the design requirements of CPS. Then, we followed it to develop a meta-model for the semi-formal definition of a CPS and its cyber and physical components. This represents a step toward the formal modeling of such systems (in the following chapter).

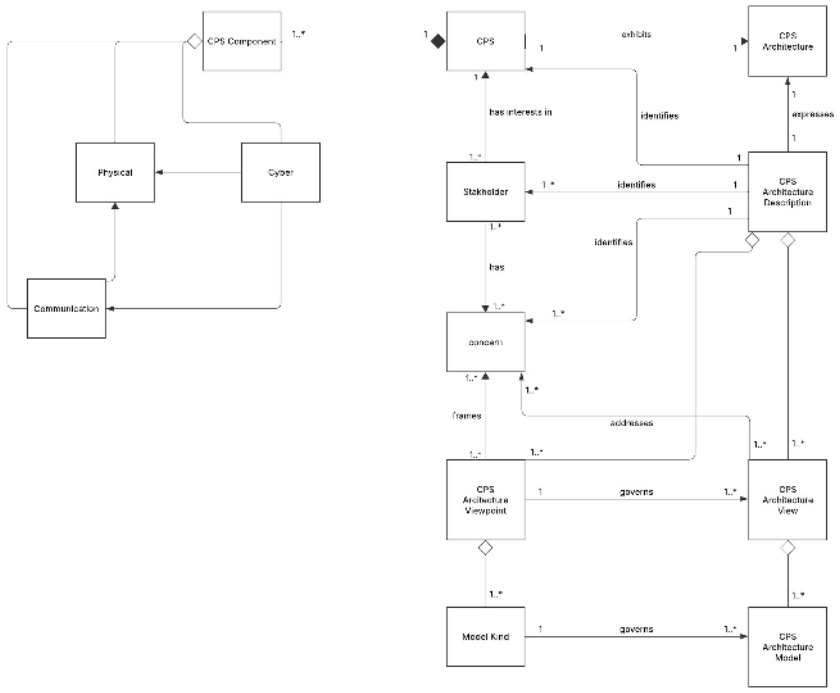


Figure 4.11: A Conceptual Model for CPS Architecture Description

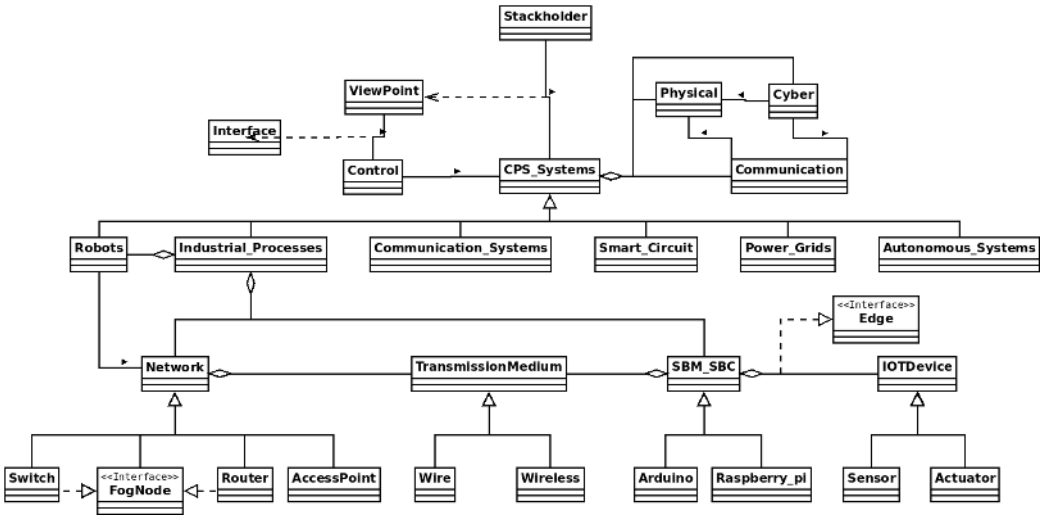


Figure 4.12: A CPS Meta Model According to IEEE 42010

Chapter 5

CA-BRS: A Formal Model for Secure CPS

5.1 Introduction

The modeling and analysis of CPS is an inherently multidisciplinary undertaking. Anyone starting in this field will unavoidably face the need for a solid foundation to outcome the complexity of these systems. Although in specific disciplines many techniques are used already as a matter of standard practice, their fundamentals and application are typically far from practitioners of this area.

This chapter focuses on the application of formal methods for defining complex systems and reasoning about their dynamic and emergent behaviors. Given the inherent complexity of CPS, a distinct formal modeling approach is required, necessitating a unified formalism that enables communication across all CPS dimensions.

However, to date, there is no single formalism capable of adequately supporting the multiple dimensions involved in the CPS design. Therefore, the objective of this chapter is to present essential definitions and terminology related to the CA-BRS formal model that we have proposed. By combining Control Agents and BRS, this model allows for the representation of three key dimensions in CPS: Structural, Virtual, and Behavioral.

5.2 CA-BRS Definition

The modeling of CPS system is inherently represented in multiple views of the system; following the principle of separation of concerns. Obviously, in CPS engineering, the results are models reflecting distinct views of the problem, expressed in multiple notation. One problem for the CPS engineer, is how to integrate different models to form a more appropriate model of the reality? Naturally, modeling of CPS systems calls for a trans-disciplinary approach that merges the different models into a unified abstraction of reality. For this purpose, we propose an extending BRS formalism, called CA-BRS (for Control Agent and BRS) that combines two models, in contrast to other approaches found in the literature, that seek to find a single model for the overall CPS layers. One model (based on BRS) is used to model the structural dimension of a CPS (the *physical and cyber* worlds) and another to model its *virtual (or logical)* dimension.

In the CA-BRS model, multiple agents are composed with a bigraph, and computation can flow in space without flowing in time, or flow in time without flowing in space, or flow in both. CA are able to observe the structural part of a CPS, and map observations to control and migration actions.

Definition 5.2.1. Formally, the model CA-BRS defining a CPS is given by the tuple: $CA\text{-}BRS_{CPS} = (CA_{CPS}, B_{CPS}, Host_{CPS}, TR_{CPS}, AC_{CPS})$, where:

1. $CA_{CPS} = A_{Ph} \cup A_{Cy}$ is a set of control agents monitoring two distinctive types of entities (Cyber and Physical). Each agent type can be in a given state of the set S, i.e., $St_{CPS} : CA_{CPS} \rightarrow 2^S$

2. $B_{CPS} = (V_{CPS}, E_{CPS}, ctrl_{CPS}, GP_{CPS}, GL_{CPS}) : < m, \varnothing > \rightarrow < n, \varnothing >$ is the sorted bigraph where agents CA_{CPS} may host.

   (a) $V_{CPS} = V_{Ph} \cup V_{Cy}$ is a set of nodes that represents a set of physical (V_{Ph}) and cyber (V_{Cy}) entities of the CPS,

   (b) E_{CPS} is a set of edges representing possible relationships and links between the CPS entities,

   (c) $Ctrl_{CPS} : V_{CPS} \rightarrow K_{CPS}$ is a mapping function that associates to each node type its signature in K_{CPS},

   (d) GP_{CPS} is the derived places graph defining explicitly the parent function $Prnt_{CPS}$ of all node types. These nodes may be grouped into roots (regions) according to their membership,

(e) GL_{CPS} is the associated links graph; each node may have a fixed number of ports (P) allowing to attach nodes to each other, through the link map $link_{CPS}$

(f) n and m are ordinal numbers indicating the number of roots and sites respectively.

3. $Host_{CPS}$ is a hosting function that associates to each control agent ($\in CA_{CPS}$), nodes where it may host. In our case, $Host_{CPS} = (Host_{Ph}, Host_{Cy})$, such that $Host_{Ph} : A_{Ph} \rightarrow 2^{V_{Ph}}$ and $Host_{Cy} : A_{Cy} \rightarrow 2^{V_{Cy}}$

4. TR_{CPS}: is a set of Agents State rules, expressing the Agents state evolution given their location,

5. AC_{CPS}: is a set of ordinary reaction rules applied to change the corresponding bigraph topology .

Obviously, each CPS element has a semantic interpretation in the context of CA-BRS. Table 5.1 summarizes the main set of correspondence rules between the CA-BRS model concepts and the different constituents of a given CPS architecture, its control part as well as its dynamics. This generic correspondence is based on the CPS meta-model given in Chapter 4.

Example 5.2.1. We show here how to represent a simple model of M-CPS in order to motivate the need for bigraph based semantics for CPS entities (Cyber V_{Cy} and physical V_{Ph} entities), and control agents (CA) for modeling the virtual aspect of these entities.

We Consider the M-CPS example represented in Figure 5.1. The architecture of this M-CPS comprises multiple cyber and physical entities that can be geographically distributed. To ensure the privacy and security of patient data, it employs an Access Control model. This model defines specific access control rules for designated users, such as doctors, nurses, or billers, granting them access to a predefined subset of actions and content. Notably, the information collected through the Electronic Health Record (EHR) is distributed, meaning it is stored across various heterogeneous hosts. Additionally, the access control rules may be managed by different EHR organizations, each retaining control over their respective resources.

The corresponding B_{CPS} of the considered M-CPS is defined to model not only the location of CPS components with the $Prnt_{CPS}$ function, but

Table 5.1: Correspondence Rules between *CA-BRS* and CPS

CPS Constituents	**CA-BRS Concepts**
Architectural part of a CPS	B_{CPS}
Geographical location	Root (number of roots is n)
Cyber Constituents(Cloud service, Application, etc.)	V_{Cy}
Physical Constituents(Computer, Processing Unit, Power Unit, Communication Unit, Sensor Nodes)	V_{Ph}
Nesting of a component in its composite	$prnt_{CPS}$
Possible interaction between components	$link_{CPS}$
Control part of a CPS	CA_{CPS}
Abstract agent for physical entities monitoring	A_{Ph}
Abstract agent for cyber entities monitoring	A_{Cy}
A possible set of the Ai agent states	$St_{CPS}(Ai)$
Relationship Architecture/Control in CPS	$Host_{CPS}$
Possible location of Agents controlling Cyber entities	$Host_{Cy}$
Possible location of Agents controlling Physical entities	$Host_{Ph}$
Dynamic action in CPS	
Possible changes in CPS architecture	AC_{CPS}
Possible state change of Agents	TR_{CPS}

also their connectivity thanks to the function $link_{CPS}$ (see Figure 5.2). To simplify the figure, we do not consider all the M-CPS constituents.

In our case, we decouple the physical entities of CPS, for example, *Doctor*, *Nurse*, to cyber elements, for example, *Personal Data*, *Conditions*. Links between the two element types allow us to track relationships between entities to represent, for instance, the relationship between the *Personal Data* of *Patient* node on the one hand, and the *Doctor* on the other hand (link *read*).

Thus, for this example,
V_{Ph} = $\{Doctor, Nurse, PatientA$,

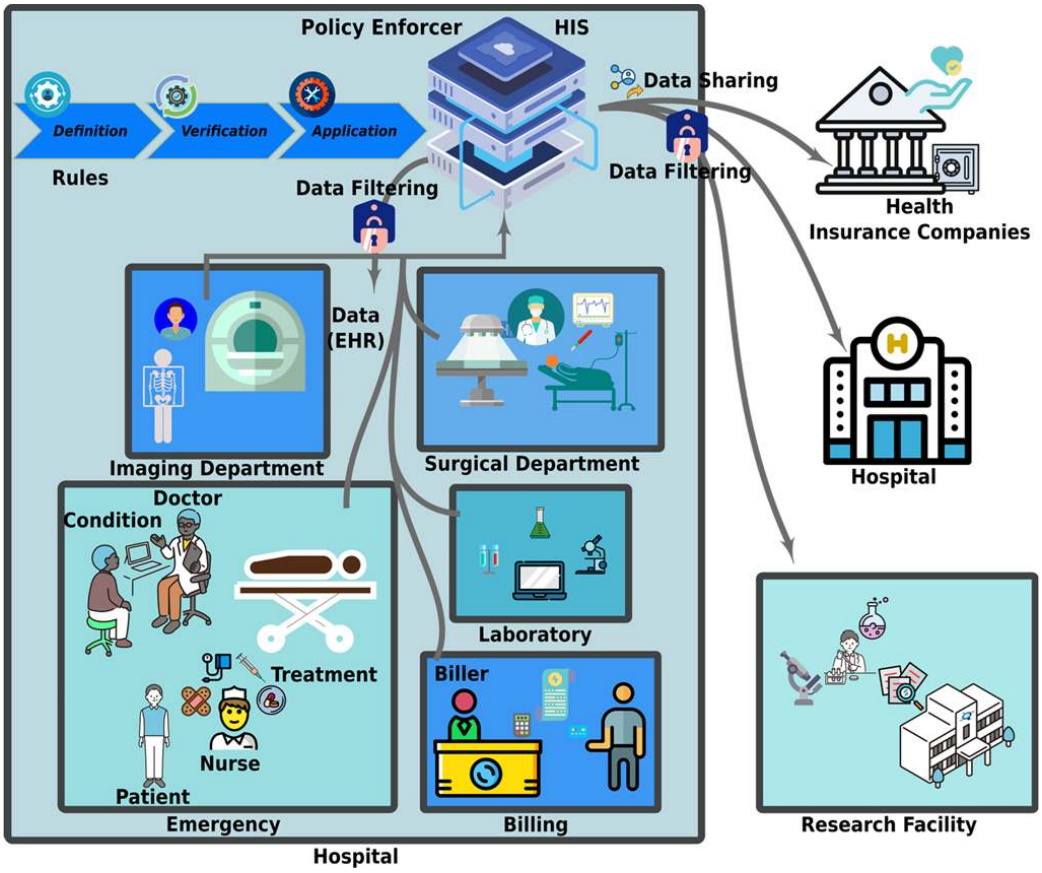


Figure 5.1: A Medical CPS Example

$Billing, Biller, Emergency, Doctor(Research)\}$ and
$V_{Cy} = \{Condition, PersonalData\}$.
The kinds of nodes and their number of ports (arity) constitute the signature K_{CPS} of B_{CPS} defined by the function $Ctrl_{CPS}$. For instance,
$Ctrl_{CPS}(Doctor) = D : 1$, $Ctrl_{CPS}(PersonalData) = P : 1$.
The link graph of our example may contain *read*, *write* and *partialread* hyperedges, represented by lines connecting ports of various nodes or regions.

The regions $HEALTH\text{-}CENTER$ and $LABORATORY$ and holes *0, 1, 2, 3* enable the composition of placing graphs, that is, a hole *0* can be replaced by a region of another bigraph using the composition operator.

We note in this step, the instantiation of the $CA - BRS_{CPS}$ structural part through the considered M-CPS example and the proposition of some

control agent types to manage the M-CPS entities (The virtual part).

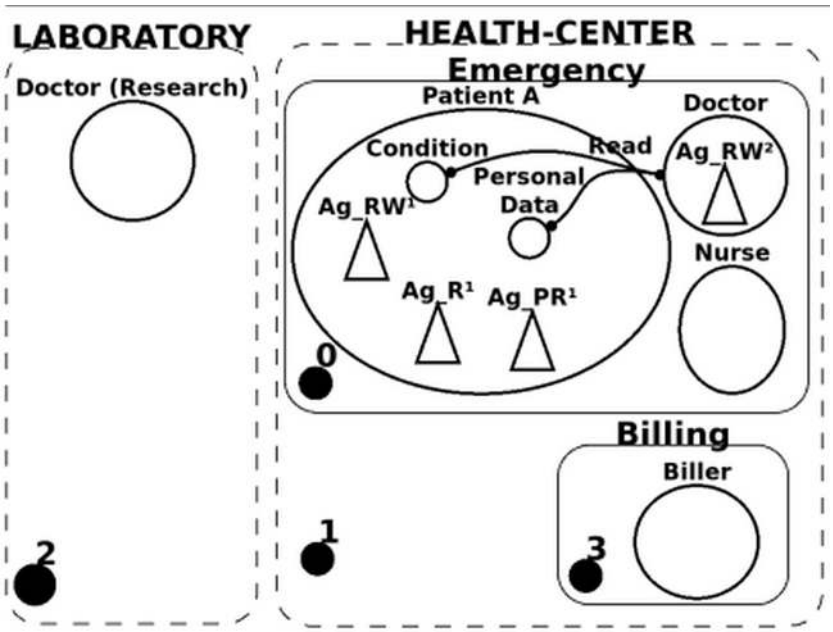


Figure 5.2: CA-BRS modeling a CPS Example

5.3 $CA-BRS$ Term Language

We have introduced far seen two ways of representing a CA-BRS: as a 5-tuple with interfaces, or using a graphical notation, there exists a third way to describe CA-BRS: Textual Term. A term algebraic language is defined to represent $CA\text{-}BRS_{CPS}$ textually. Its operator set extends that of bigraphs [1] to deal with the virtual elements of $CA\text{-}BRS_{CPS}$ as indicated in Table 5.2. For instance, the $U\{A_i\}$ term is used to indicate that the agent A_i is hosted in the node (or the root) U. The communication term $[A_i, A_j]_x$ makes the two (or more) agent in brackets communicating through the link x. Finally, the operator _&_ allows associating the state S to the agent $A, S \in St_{CPS}(A_i)$.

The algebraic term of CA-BRS given in Figure 5.2 is:

$$Laboratory.[Doctor(Research)/d2] \| \\ Health-center.[Emergency.[Patient...]/Billing.[Biller...]/d1]$$

Table 5.2: Algebraic language of $CA\text{-}BRS_{CPS}$

Term	Signification
$U\|\|V$	Parallel product
$U\|V$	Merge product
$U.V$	Nesting (U contains V)
K_x	Node of type ($control$) K having a name or link x
id	Identity or elementary bigraph
d_i	Site numbered i
1	The barren ($empty$) region
$A\&S$	Agent A in a given state S
$[A_i, A_j]_x$	Agent A_i and A_j communicate via the link x
$U\{A\}$	U is controlled by A, i.e. A is hosted in U

In the present terminology, we employ $U\{A\}$ to denote that the agent instance A is hosted within the node U. Additionally, $A\&S$ highlights that the control agent A exists in a possible state S.

5.4 $CA-BRS$ Dynamics

Until now, our illustration has primarily addressed the static aspects of the CPS. However, it is crucial to acknowledge that CA-BRS exhibits dynamic behavior as well. Figure 5.3 depicts three straightforward examples, accompanied by the corresponding terms, illustrating the potential types of CPS evolution.

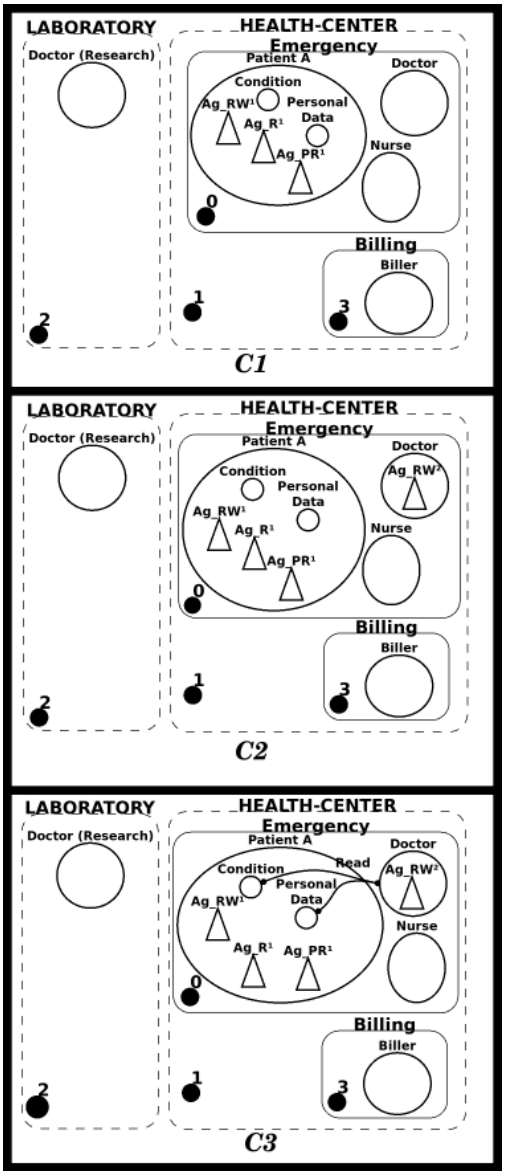


Figure 5.3: Example of CPS evolution

Indeed, CPS dynamic behaviors, involve close-interactions between the two dimensions (Structural and Virtual); a certain change in one world should be reflected in the other world. We define first CPS configuration at a given time, consisting of its actual BRS and its current hosting function (repre-

senting agents location in the structural environment), as it is illustrated in Figure 5.3. Then, we give a generic definition of the complex and adaptive behavior of CPS through several types of rules, each manages a possible evolution of a CPS component (physical, cyber or virtual). This introduces the third dimension of CA-BRS, complementing the two previously discussed dimensions: Structural and virtual.

Definition 5.4.1. Given a CPS specified with the CA-BRS, configuration C describing its current state is defined by the pair $C = (B_{CPS}, H_{CPS})$, where

- B_{CPS} describes the structural part of the CPS at given time,
- H_{CPS} is the distribution of the agent instances through the nodes of B_{CPS}, or precisely their hosting nodes at a given time, that is, if $H_{CPS}(Asi) = \{N1, N2, ...\}$, then Asi is a possible instance of Ai ($\in CA_{CPS}$) and $N1, N2, ... \in Host_{CPS}(Asi)$.

Example 5.4.1. A possible configuration of the above example is depicted in Figure 5.3 where H_{CPS} is given by:

$$H_{CPS}(AgRW^1) = Doctor,$$
$$H_{CPS}(AgRW^2) = Doctor,$$
$$H_{CPS}(AgRW^1) = PatientA,$$
$$H_{CPS}(AgPR^1) = PatientA,$$
$$H_{CPS}(AgR^1) = PatientA.$$

By modeling the interactions and behaviors of individual agents (of CA_{CPS}), emergent properties can arise at a higher level, which may not be immediately predictable from the behavior of individual agents alone. More precisely, the CA-BRS state evolution is dictated by three types of local rules, allowing the behavior of the structural and virtual dimensions to evolve separately:

1. The $CA-BRS_{CPS}$ state change altering only the topology of the CA-BRS is possible owing to the *Action Rules* AC_{CPS}. Autonomous agents (CA_{CPS}) control the physical and cyber entities of their environment through these ordinary reaction rules.
2. Agent observations or *Trigger Rules* (TR_{CPS}) allow agent states to evolve, by rewriting rules, acting upon, and influencing their decision-making processes.

3. A second set of *Rearrangement Rules* (GR_{CPS}) is also defined to represent possible changes in the virtual dimension of the CA-BRS. So, control agents in CA_{CPS} can communicate with each other, they can migrate from one node to another, and even create new instances of agents or destroy them.

In the following subsections, we will explain each type of this rule and it affects the behavior of $CA-BRS$ while using the running example for illustration.

5.4.1 Action Rules (AC_{CPS})

The state change altering only the topology of the $CA-BRS$ is possible thanks to these ordinary reaction rules. Autonomous agents (CA_{CPS}) control physical and cyber entities of their environment through these ordinary reactions rules.

Definition 5.4.2. An Action rule is an ordinary reaction rule of bigraphs, i.e. $\lambda_i : B_{CPS} \rightarrow B'_{CPS}$, allowing to Create/Destroy a node of B_{CPS} or Create/Destroy a link of this bigraph.

Example 5.4.2. Referring to our running example of M-CPS, starting with the initial configuration given in Figure 5.3, the different rewriting rules already defined can be identified in this way, according to the following examples in Figure 5.4. λ_1 allows to create a link "Read" between nodes: Doctor and PersonalData. On the other hand, λ_2 permits to Create a link "Read" between the nodes Doctor and Condition.

5.4.2 Trigger Rules (TR_{CPS})

These agents observations rules allow agent states to evolve, through rewriting rules, acting upon and influencing their decision-making processes.

Definition 5.4.3. A Trigger rule is of the form: $\gamma_i : N\{A\&S1\} \rightarrow N\{A\&S2\}/A \in CA_{CPS}, N \in H_{CPS}(A)$ and $S1, S2 \in St_{CPS}(A)$ (possible states of an agent instance A).

Example 5.4.3. In the context of our example, these rules are defined as shown in Figure 5.5. They allow to model the state evolution of each control agent (CA) instance, present in the actual configuration of the CPS. γ_1

λ1: B1 → B21 Create a link "Read" between nodes: Doctor and PersonalData
...Health-center.[Emergency.[PatientA.[Condition/**Personal Data...**]/**Doctor**/ Nurse]/...] →
...Health-center.[Emergency.[PatientA.[Condition/ **Personal Data**$_{\text{Read}}$...]/**Doctor**$_{\text{Read}}$/ Nurse]/...]

λ2: Create a link "Read" between nodes: Doctor and Condition
...Health-center.[Emergency.[PatientA.[**Condition**/Personal Data$_{\text{Read}}$...]/**Doctor**$_{\text{Read}}$/ Nurse]/...] →
...Health-center.[Emergency.[PatientA.[**Condition**$_{\text{Read}}$/ Personal Data$_{\text{Read}}$...]/**Doctor**$_{\text{Read}}$/ Nurse]/...]

λ5: Create a link "Partialread" between nodes: Doctor Research and Condition
Laboratory.[Doctor Research]//Health-center.[Emergency. [PatientA.[Condition$_{\text{Read}}$/Personal Data...]/Doctor$_{\text{Read}}$/ Nurse]/...] →
Laboratory.[**Doctor Research**$_{\text{PartialRead}}$]//Health-center. [Emergency.[PatientA.[**Condition**$_{\text{PartialRead}}$ $_{\text{Read}}$/Personal Data...]/Doctor$_{\text{Read}}$/ Nurse]/...]

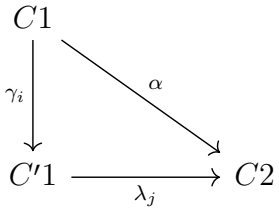
Figure 5.4: Example of M-CPS Rewriting Rules λ_i

makes change of the $AgRW^1$ state from 'Noread' to 'Read'. Similarly for the instance of agent $AgRW^1$ with the trgger rule: γ_2.

The execution semantics of these two types of rules (labeled γ_i and λ_i in the following diagram) can be applied either sequentially or concurrently for CPS configuration changes, utilizing the two combination operators below:

";": for a sequence composition and,

"/": for a parallel composition.



γ1:
PatientA{AgRW[1]& [**Noread**, Nowrite]}→
PatientA{AgRW[1]&[**read**, Nowrite]}

γ2:
Doctor{AgRW[2]& [**Noread**,Nowrite]} →
Doctor{AgRW[2]& [**read**, Nowrite]}

γ5:
PatientA{AgPR[1]& **Nopartialread**}→
PatientA{AgPR[1]&**Partialread**}

γ6:
DoctorResearch{AgPR[2]& **Nopartialread**}→
DoctorResearch{AgPR[2]&**Partialread**}

Figure 5.5: Example of Trigger Rules for the M-CPS model

5.4.3 Rearrangement Rules (GR_{CPS})

A second set of rules altering control agents is also defined to represent possible changes in the virtual dimension of the CA-BRS. So, control agents in CA_{CPS} can communicate with each other, they can migrate from one node to another, we can even create new instances of agents or destroy them.

Definition 5.4.4. A rearrangement rule (GR) of label β, is a local rewriting rule which may be of the following type:

1. $New(Ag, Ag_i, S, H_{CPS})$: create a new instance Ag_i of the type agent Ag, having the state S in a hosting node defined by H_{CPS},
2. $Des(Ag, Ag_i, H_{CPS})$: destroying an existing instance Ag_i of the type agent Ag and hosting in a node defined by H_{CPS},
3. $Mig(Ag_n\&\{S\}, H_{CPSi}, H_{CPSj})$: migrating the agent instance with its current state S from its hosting node H_{CPSi} to another possible hosting node H_{CPSj},
4. $[Ag_i, Ag_j]_x$: creating a communication link of type x between the two agent instances Ag_i and Ag_j.

Example 5.4.4. Continuing in the context of our example, we present in Figure 5.6 several examples of rearrangement rules for illustration.

β1: New(AgRW^2, AgRW, <Noread, Nowrite>, Doctor)

Laboratory.[Doctor Research]//Health-center.
[Emergency.[PatientA.[Condition/Personal
Data...]/**Doctor{ }**/ Nurse]/...] →
Laboratory.[Doctor Research]//Health-center.
[Emergency.[PatientA.[Condition/Personal
Data...]/**Doctor{AgRW^2&<Noread, Nowrite>}**/
Nurse]/...]

β2: New(AgPR^2, AgPR, Nopartialread, Doctor Research)

Laboratory.[**Doctor Research{ }**]//Health-center.
[Emergency.[PatientA.[Condition$_{Read}$/Personal
Data...]/Doctor$_{Read}${AgRW^2&<Noread, Nowrite>}/
Nurse]/...] →
Laboratory.[**Doctor Research{AgPR^2&Nopartialread}**] //
Health-center.[Emergency.[PatientA.
[Condition$_{Read}$/Personal
Data...]/Doctor$_{Read}${AgRW^2&<Noread, Nowrite>}/
Nurse]/...]

Figure 5.6: Rearrangement rules for Illustration

5.4.4 Controlled Reaction Rules (CRR)

The global dynamic behaviors of CPS, involving close interactions between the structural (B_{CPS}) and virtual (CA_{CPS}) worlds (a certain change in one world must be reflected in the other world) should be defined by a set of global rules, noted Controlled Reaction Rules(CRR_{CPS}), involving the two previous definitions of the local rules (AC_{CPS} and TR_{CPS}). They are applied to CPS states (or configurations) in order to represent the dynamic behavior evolution of these systems. Each CRR rule noted α_i (see Figure 5.7) expresses the conditioned change of any CPS state, triggered by one or more controlling agents observations ($\gamma_i \in TR_{CPS}$), and materialized by a change ($\lambda_i \in AC_{CPS}$) in the topology (B_{CPS}) of the CPS.

Example 5.4.5. This example is dedicated to formally define and explain CRR rules, specifying dynamic and adaptive behavior of the M-CPS example that may be controlled or influenced by agents' observations. More details

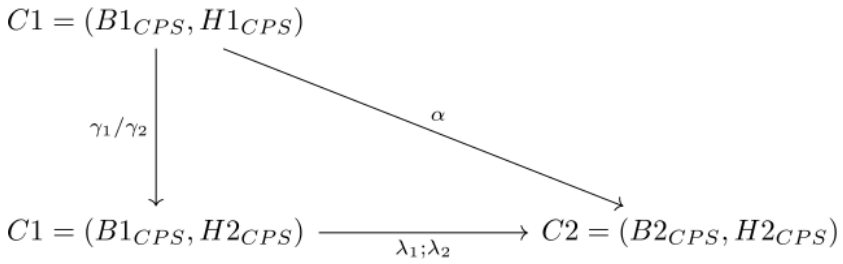


Figure 5.7: CRR Generic Notation

may be found in [98]. We consider two CRR: Rule1 and Rule2 in Figure 5.8 representing the dynamic and secure evolution of the M-CPS regarding respectively the following scenarios:

1. *Scenario1*: We claim that the Doctor responsible for treating patients is allowed to see all medical examinations.

2. *Scenario2*: Neither the Doctor nor the Nurse are authorized to consult their patients' information beyond their work context, for example in the case of scientific research.

The first CRR (Rule1) gives a right to the Doctor to read the medical examinations ("Condition" and "PersonalData") of a given patient. Knowing that the agent $AgRW^1$ hosting in "PatientA" to control it, has changed its state thanks to the rule γ_1. Similarly, an instance of the $AgRW$ agent (i.e $AgRW^2$), created by the rule β_1 to control the Doctor node, also has changed its state (rule γ_2). The execution of the two rewriting rules can be done in parallel (γ_1/γ_2). Therefore, the link "Read" will be established ($\lambda_1;\lambda_2$) between corresponding nodes.

The second CRR (Rule2) expresses that the doctor may only read partial patient information when he is in its research laboratory. It serves to control the Doctor's access to the PatientA's personal information ("Condition"). It is materialized by the establishment of a "Partialread" link between the nodes "DoctorResearch" and "Condition". Obviously, this constitutes the result of different observations (γ_5 and γ_5) and evolution β_2 of the agent instances in question ($AgPR^1$ and $AgPR^2$).

Figure 5.9 summarizes the configurations involved in defining the secure behavior of our example (scenarios relating to Rule1 and Rule 2). According to $CA-BRS$ behavior execution, we have defined in terms of states and

transition rules, the Observation rules of CA_{CPS} and the actions altering the $CA-BRS$ topology that both form CRR rules, as well as Rearrangement rules. Other alternatives can be used to define these rules (the necessary details will be provided in the next chapter).

$CA-BRS_{CPS}$ constitutes an appropriate model for CPS where computational entities are embedded in physical and cyber ones. Since it operates over a shared environment, some undefined behaviors may emerge due to concurrency. Besides, as the number of state variables of CPS entities (virtual, cyber or physical) increases, the size of the system state space grows exponentially. In this context, we have to define a particular transition system to specify the $CA-BRS_{CPS}$ behavior. Therefore, a Guided Transition System (GTS), a state space definition, is proposed to prevent these behaviors based on the parallel composition between the observations of the agents and based on partitioning the physical structure where each agent operates in its own region. Traditional transition systems are insufficient to represent and reason about the interactions and dependencies between concurrently executing CPS components.

In the following chapter, we will define this GTS through its two types of rules (CRR and GR) to deal with the state evolution of the considered CPS example. In this rewriting system, complex structures are built from simple following specific rules. These structures can represent different shapes, patterns, or components, and they can interact with neighboring structures based on predefined rewriting rules.

We note that modeling CPS behaviour as GTS allows for performance analysis, such as deadlock detection, workload balancing, and response time estimation. These analyses can be crucial for adaptive behavior of safe and secure CPS.

5.5 Conclusion

This chapter addresses a significant gap that affects the practical application of formal methods in CPS development. To bridge this gap, we proposed a novel formal model called CA-BRS, which allows defining and analyzing the dynamic behavior of CPS. The CA-BRS model incorporates Control Agents and, therefore, introduces a new computational model known as Guided Transition Systems.

	Scenario1	**Scenario2**
Description	The Doctor responsible for treating patients is allowed to see all medical examinations.	Neither the Doctor nor the Nurse is authorized to consult their patients' information beyond their work context, for example in the case of scientific research.
CRR	β_1 γ_1/γ_2 $C_1 \rightarrow C_2 \rightarrow C_3$ $\lambda_1; \lambda_2$ where $C_1 = (B_1, H_1)$, $C_2 = (B_1, H_2)$, $C_3 = (B_2, H_2)$	β_2 γ_5/γ_6 $C_4 \rightarrow C_5 \rightarrow C_6$ λ_5 where $C_4 = (B_4, H_4)$, $C_5 = (B_4, H_5)$, $C_6 = (B_5, H_5)$
Observations	γ_1: PatientA.AgRW^1& [Noread, Nowrite}→ PatientA.AgRW^1&[read, Nowrite} γ_2:Doctor.AgRW^2& [Noread,Nowrite} → Doctor.AgRW^2& [read, Nowrite}	γ_5: PatientA.AgPR^1& [Nopartialread}→ PatientA. AgPR^1&[Partialread} γ_6: Doctor-Research.AgPR^2& [Nopartialread}→ Doctor-Research.AgPR^2&[Partialread }
Actions	λ_1: $B_1 \rightarrow B_{21}$ Create a link "Read" between nodes: Doctor and PersonalData λ_2: $B_{21} \rightarrow B_2$ Create a link "Read" between nodes: Doctor and Condition	λ_5: $B_4 \rightarrow B_5$ Create a link "Partialread" between nodes: Doctor-Research and Condition
Rearrangement rules	β_1: New(AgRW^2, AgRW, <Noread, Nowrite>, Doctor)	β_2: New(AgPR^2, AgPR, Nopartialread, Doctor-Research)

Figure 5.8: CRR rules for M-CPS Scenarios Examples

By utilizing these models, we demonstrate in the subsequent chapter how

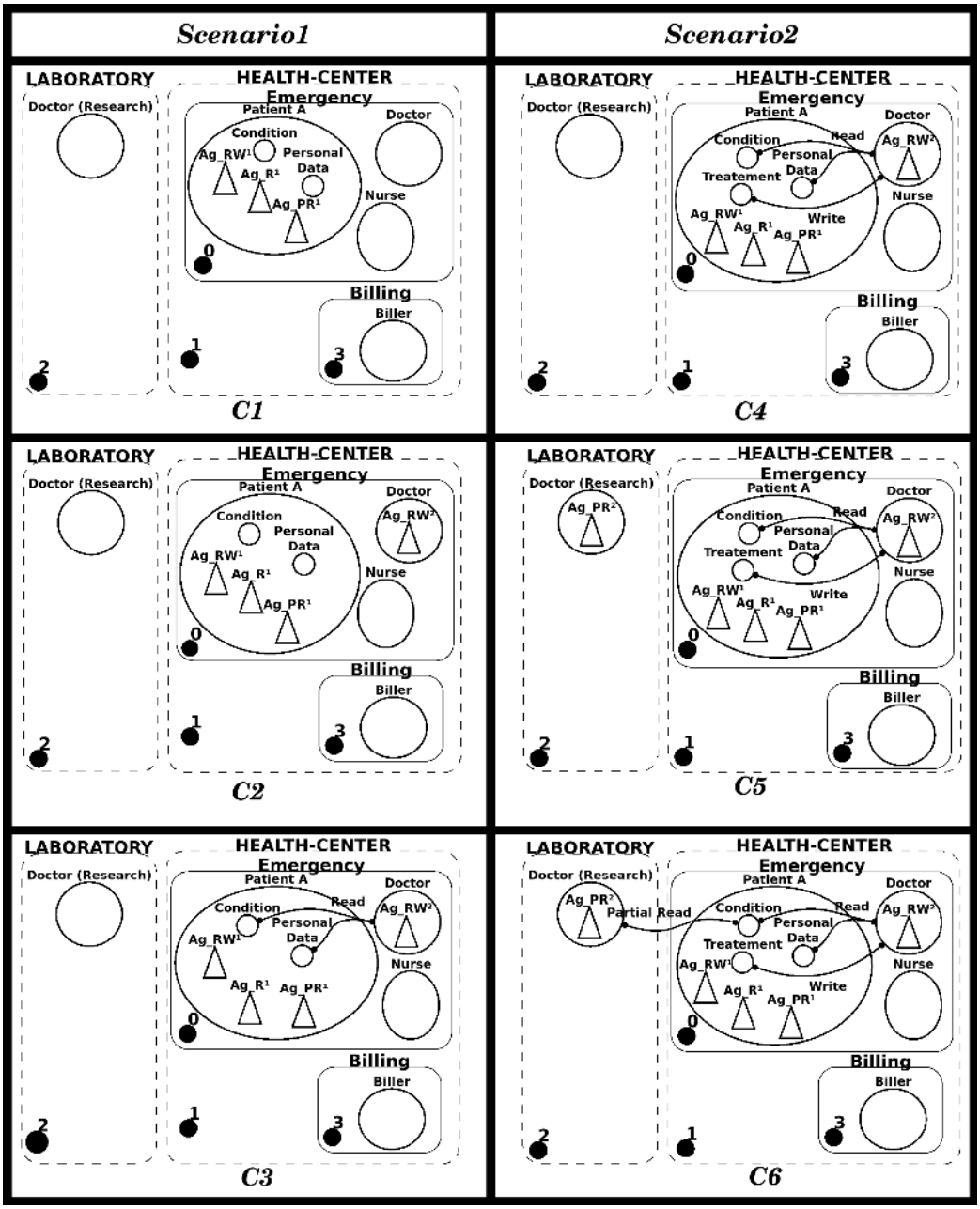


Figure 5.9: CA-BRS Configurations defining M-CPS behavior

the accuracy, safety, and security of CPS can be effectively analyzed. This

analysis facilitates the appropriate design and implementation of this emerging class of engineering systems. Furthermore, we showcase the utilization of well-established and widely recognized notations and languages such as BPMN and Maude, which brings us closer to the future development and implementation processes of CPS.

Chapter 6

Security and Safety Analysis Approach for Cyber Physical Systems

6.1 Introduction

Some environments are inherently security-sensitive. We can think for instance of airports, hospitals, power plants, military sites, and so on. In this thesis we consider the challenge of engineering CPS, related to these domain applications, in order to keep security into account from the very first steps of their development life-cycle. Our approach to the problem starts from the idea that security should be taken into account since the very beginning in the CPS design. We also want to be able to take into account potential changes that may occur later and ensure that the system can adapt automatically to continue to satisfy security requirements.

Moreover, the extensive integration of IoT technologies converts the CPS into smart, efficient, and complex hybrid systems. However, such dependency creates a vast attack space that the attackers can exploit. Thus, to secure the CPS, we investigate formal security analysis techniques applicable to the extended proposed BRS model (CA-BRS).
In this chapter, we define a formal and computational model (GTS for Guided Transition System) offering a powerful means to capture and analyze the complex behavior exhibited by CPS, enabling designers and researchers to reason about their properties, verify their correctness, and further, optimize

their performance. By modeling the interactions and behaviors of individual agents, emergent properties, such as security and safety of CPS entities can arise at a higher level, which may not be immediately predictable from the behavior of individual agents alone. Section 6.2 presents the various concepts defined in the Guided Transition System (GTS) framework to model the adaptive behavior of CPS. Section 6.3 introduces a well-defined semantics to control the behaviors of agents within the CPS. The semantics addresses two complementary perspectives on the CPS, offering different analysis approaches to the specified system. The interpretation and implementation of each perspective provide valuable insights into the CPS. Section 6.4 describes the analysis approach for evaluating the security and safety properties of the CPS. The analysis is based on the execution traces of the GTS model.

6.2 Guided Transition System Definition

The CA-BRS model is an appropriate representation for CPS where computational elements are embedded within both physical and cyber components. Since the CA-BRS model operates over a shared environment, some undefined or emergent behaviors may arise due to concurrent interactions. Furthermore, as the number of state variables associated with the CPS entities (virtual, cyber, or physical) increases, the overall state space of the system grows exponentially in size.

To address these challenges, the Guided Transition System (GTS) is proposed. The Guided Transition System leverages a parallel composition approach between the observations of individual agents, as well as a partitioning of the physical structure, where each agent operates within its own designated region.

Traditional transition system models are insufficient to adequately represent and reason about the complex interactions and dependencies that can arise between the concurrently executing components of CPS.

GTS provides a more suitable formalism for modeling and analyzing the dynamic behavior of such intricate CPS. Therefore, to describe the potential behavior of this CA-BRS model-based CPS, we refer to the definition of a specific transition system (Guided Transition System) that consists of states (or configurations) and transitions between states. In this case, the transitions mathematically correspond to two sets of rules:

1. A set of controlled reaction rules (CRR)

2. A set of rearrangement rules (GR)

These rules were formally defined in the previous chapter.

Definition 6.2.1 (Guided Transition System)**.** Formally a Guided Transition System modeling CPS behavior specified with $CA-BRS$ model, is the tuple $(Cs, \Psi, \rightarrow)$ where:

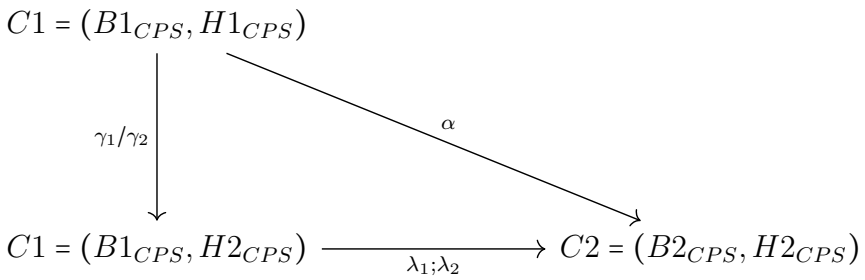
- Cs is the set of configurations $C = (B_{CPS}, H_{CPS})$ as already defined, (Definition 2.2.3)
- $\Psi = \Gamma \cup \Lambda$ is a set of rule labels, Γ (respectively Λ) is a set of controlled reaction rule labels (respectively rearrangement rule labels),
- $\rightarrow \subset C \times \Psi \times C$ is the guided transition relation.

If $C1$ and $C2$ are two possible configurations ($\in Cs$) then $(C1, C2) \in \rightarrow$ means that there is:

- a rearrangement rule of label $\beta \in \Lambda$, from $C1$ to $C2$ noted:

$$C1 \xrightarrow{\beta} C2$$

- a controlled reaction rule of label $\alpha \in \Gamma$, noted:



In the following, we will show how GTS of CA-BRS model is used to specify the M-CPS secure behavior by encoding specific computations into CRR rules (α) or GR rules (β).

Example 6.2.1. We will examine two examples of scenarios that illustrate the dynamic behavior of the M-CPS, as represented using the CA-BRS model shown in Figure 6.1.

Obviously, these examples illustrate how to model, using the agents of $CA - BRS$ formalism, the access control of the various actors present in the CPS. Further details regarding the security aspect of the CPS will be provided in the last section of this chapter. Here, we will focus on how to define execution traces that comply with the GTS definition of this M-CPS example.

- **Scenario1:** We claim that the Doctor responsible for treating patients is allowed to see all medical examinations.
- **Scenario2:** Neither the Doctor nor the Nurse are authorized to consult their patients' information beyond their work context, for example in the case of scientific research.

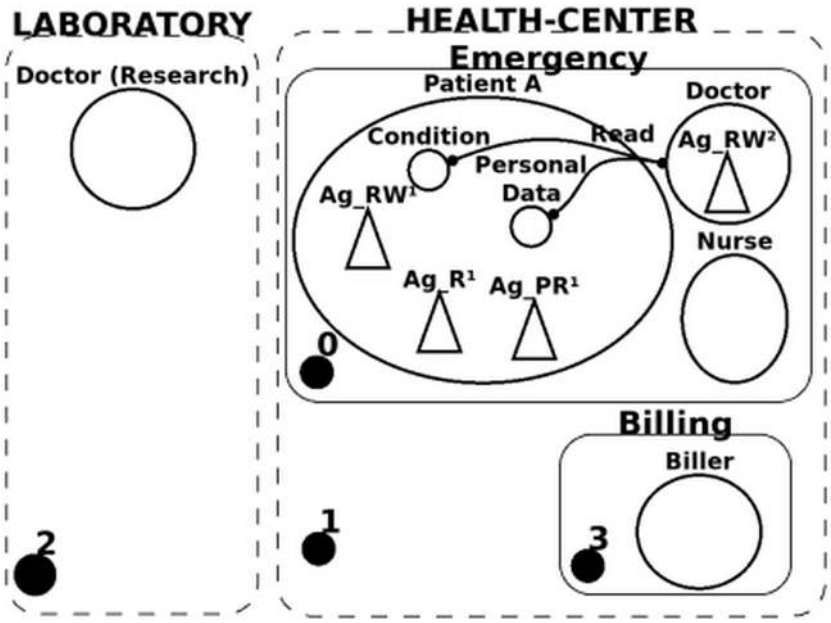


Figure 6.1: CA-BRS model for M-CPS example

We recall in Table 6.1, two CRR rules and two GR rules, as a concise and formal way to represent the concurrent and secure behavior of M-CPS dealing with some access control rules that we will explain in the following.

CRR capture the interactions and dependencies between involved rewriting rules (γ_i, λ_j) and provide a clear picture of how these rules are executed concurrently.

GR rules are proposed to define agents mobility in order to rearrange their execution.

Hence, in scenario 1, we give a right to the Doctor to read the medical examinations ("Condition" and "Personal Data") of a given patient. Knowing that the agent $AgRW^1$ hosting in "PatientA" to control it, has changed its state thanks to the rule γ_1. Similarly, an instance of the $AgRW$ agent (i.e $AgRW^2$), created by the rule β_1 to control the "Doctor" node, also has changed its state (rule γ_2). The execution of the two rewriting rules can be done in parallel (γ_1/γ_2). Therefore, the link "Read" will be established between corresponding nodes.

In scenario 2, we express that the doctor may only read partial patient information when he is in its research laboratory. "Rule2" serves to control the Doctor's access to the PatientA's personal information. It is materialized by the establishment of a "Partialread" link between the nodes "Doctor Research" and "Condition". Obviously, this constitutes the result of different observations (γ_5 and γ_6) and evolution β_2 of the agent instances in question ($AgPR^1$ and $AgPR^2$).

The corresponding execution traces are depicted on the guided transition system (GTS) of Figure 6.2a. and Figure 6.2b., respectively.

The configurations involved in defining the secure behavior of our example (scenarios 1 and 2) are already given in Chapter 5.

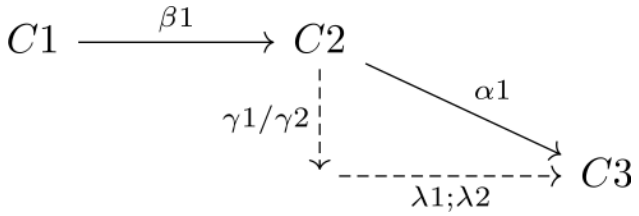
Modeling M-CPS behavior as GTS also enables performance analysis, including deadlock detection, workload balancing, and response time estimation. These analyses play a key role in ensuring the efficient operation of M-CPS and its adherence to design objectives.

6.3 Modeling CPS Secure Behavior

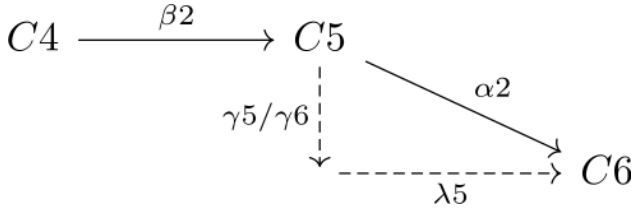
The behavioral model of the CA-BRS depends on the behavior of both the CA-BRS agents and the physical/cyber entities involved.

To implement the corresponding GTS for this model, we propose two possible approaches for interpreting the abstract CA-BRS agents (CA_{CPS}): the *States* perspective and the *Activities* Perspective.

For each of these perspectives, we suggest an appropriate implementation



(a) Execution Trace of scenario 1



(b) Execution Trace of scenario 2

Figure 6.2: Execution Traces Example

language that can achieve more expressive and practical results when putting the CA-BRS model into practice.

To explain our proposal, Table 6.2 indicates the use of two distinct languages for each considered perspective. Indeed, the provided results can be combined to effective design CPS and even reason about their behavior.

- The Maude language [99] seems more suitable to represent control agents (CA_{CPS}) through their states and transition rules. The state space definition of the CA-BRS model, used to represent an adaptive CPS behavior that may be controlled or influenced by agents' observations, is naturally defined in Maude through rewriting rules and strategies [100, 101].

- The BPMN (Business Process Model and Notation) modeling approach is quite common when the designer wants to specify the CA_{CPS} and their hosting entities as a set of interacting processes. In this case, each individual process represents a physical, control, or cyber entity that is part of the CA-BRS model. The control aspect is further divided into physical control and cyber control parts.

As noted in this table 6.2, all the elements that constitute the CA-BRS model across its three dimensions—Structural, Virtual, and Behavioral—find

Table 6.1: CRR and GR rules corresponding to scenario 1 and scenario 2

	Rule 1	**Rule 2**
Description	We claim that the Doctor responsible for treating patients is allowed to see all medical examinations.	Neither the Doctor nor the Nurse are authorized to consult their patients' information beyond their work context, for example in the case of scientific research.
CRR	β1 γ1/γ2 C1 ⟶ C2 ⟶ C3 λ1; λ2 Where C1=(B1, H1), C2=(B1,H2), C3=(B2,H2)	β2 γ5/γ6 C4 ⟶ C5 ⟶ C6 λ5 Where C4=(B4, H4), C5=(B4,H5), C6=(B5,H5)
Observations	γ1: PatientA.AgRW^1&{Noread, Nowrite} → PatientA.AgRW^1&{Read, Nowrite} γ2: Doctor.AgRW^2&{Noread, Nowrite} → Doctor.AgRW^2&{Read, Nowrite}	γ5: : PatientA.AgPR^1&{Nopartialread} → PatientA.AgPR^1&{Partialread } γ6: Doctor-Research.AgPR^2&{Nopartialread} → Doctor-Research.AgPR^2&{Partialread}
Actions	λ1: B1 → B21 Create a link "Read" between nodes: Doctor and PersonalData. λ2: B21 → B2 Create a link "Read" between nodes: Doctor and Condition.	λ5: B4 → B5 Create a link "Partialread" between nodes: Doctor-Research and Condition.
Rearrangement rules	β1: New(AgRW^2 , AgRW, <Noread,Nowrite>, Doctor)	β2: New(AgPR^2 , AgPR, Nopartialread, Doctor-Research)

their appropriate semantic interpretation in either Maude or BPMN. The Structural dimension has a single interpretation in terms of bigraphs in both perspectives. However, the Virtual dimension (the set of agents present in the structural part of the CPS) is modeled differently in the two perspectives, which naturally impacts the Behavioral dimension as well. We will provide more details on each of these modeling perspectives in the following sections.

6.3.1 States Perspective

Maude is a high-level specification language based on rewriting logic, which is particularly well suited for modeling state transition systems. It allows for specifying systems as collections of states and transition rules, making it a powerful tool for formal verification, simulation, and performance analysis. In this approach, the states of GTS (Guided Transition Systems) are represented as algebraic terms, and transitions (CRR or GR rules) are expressed

Table 6.2: Two Possible Perspectives for CA_{CPS} Behavior formalization

CA-BRS Structural Dimension	**BRS semantics**	
$B_{CPS} = (V_{CPS}, E_{CPS}, ctrl_{CPS}, GP_{CPS}, GL_{CPS})$	Bigraph	
CA-BRS Virtual Dimension	**States Perspective (Maude semantics)**	**Activities Perspective (BPMN semantics)**
$CA_{CPS} = A_{Ph} \cup A_{Cy}$	• Algebraic terms having a corresponding sort and generation operators • Declared operator StCPS : associates to each Agent, its possible states	Virtual Participants of BPMN processes
$V_{CPS} = V_{Ph} \cup V_{Cy}$	Algebraic terms having a corresponding sort and generation operators	Physical/Cyber Participants of BPMN processes
$Host_{CPS} = \{Host_{Ph}, Host_{Cy}\}$	Declared operator $Host_{CPS}$: links each Agent to its possible hosting nodes	Connecting objects
CA-BRS Behavioral Dimension	**States Perspective (Maude semantics)**	**Activities Perspective (BPMN semantics)**
$C = (B_{CPS}, H_{CPS})$	A system configuration defined in a Maude module.	A given scenario of BPMN model (Collaboration diagram)
AC_{CPS}: reaction rules set, $\lambda_i : B_{CPS} \rightarrow B'_{CPS}$	Conditional rewriting rules. form a sequence (";") or a parallel ("/") composition application of these rule labels.	—
TR_{CPS}: Trigger rules γ_i of a control agent A_j hosting in N_i, $(\gamma_i : N\{A\&S1\} \rightarrow N\{A\&S2\})$	• A rewriting rule changing the agent actual state. • ";": Sequence operator (strategy) to compose the application of many rules . • "/": Parallel operator (strategy) to compose the application of many rules .	Behavior of a process A_j in terms of Events: $Event(A_j)$ and activities: $Act(A_j)$. Each trigger rule represents a sequence flow γ_i
GR_{CPS}: Rearrangement rules of label β_i	• Rewriting rules altering the hosting of a given Agent. • New (or Destroy): declared operator to create (or delete) an Agent instance. • Link: Declared message between two Agent instances	Connecting objets
CRR_{CPS}: Controlled Reaction Rules of label α, $(\alpha : (B1_{CPS}, H1_{CPS}) \xrightarrow[\lambda_1,\lambda_2,\ldots,\lambda_m]{\gamma_1,\gamma_2,\ldots,\gamma_n} (B2_{CPS}, H2_{CPS}))$	Declared Strategies in a Strategy Maude module	BPMN models simulation, γ_i and λ_i are executed sequentially.

as rewriting rules. Given an initial state of the CPS, Maude applies these rules to compute reachable states, allowing CPS execution and verification.

However, the Maude Strategy Language is employed to manage the state explosion problem caused by the increasing size of the state space in GTS.

The Maude system rewriting engine is a great asset in terms of execution and analysis of the CA-BRS behavior. We note that Maude gives naturally operational semantics to the behavioral dimension of the (CA-BRS) model, due to its rewriting rules or strategies. The virtual dimension is defined with its algebraic terms in the Maude module *CABRS-VIRTUAL*. On the other hand, the static aspects of the CPS are encoded in the Maude module *CABRS-STATIC*. These two modules define the necessary sorts (subsorts) and operations to construct the appropriate algebraic terms that represent the configuration of the CPS. The rewriting rules within these modules are defined to enable the evolution of agent states and the alteration of the structural components of the CPS. In addition, the CRR rules are encoded

as strategy rules in the Maude strategy module called *CABRS*.

All specifics of this implementation approach will be provided in the next chapter.

6.3.2 Activities Perspective

Previously, we defined the behavior of the $CA-BRS$ model or the system in question in terms of states and transitions between these states, based on events that occur. An alternative way to model this behavior is to consider it as a set of activities and processes that include decisions, loops, and concurrent activities. This *"Activities perspective"* approaches the $CA-BRS$ behavior with the BPMN language, as an industry standard developed by the OMG consortium, easily understand by both technical and non-technical stakeholders. Thus, CA_{CPS} are considered participants in the different processes, and their behaviors are specified by the corresponding workflows including possible activities and events during their execution. Thus, the behavior of each control agent A_i is defined by:

- A set $Event(A_i)$ which represents something that happens during the behavior execution of an agent A_i.
- A set $Act(A_i)$ representing tasks performed by an agent A_i within a process.

Table 6.3 illustrates the BPMN based semantics of the virtual and behavioral dimensions of CPS. Its corresponding mapping rules translate the main features of the CA-BRS virtual and behavioral dimensions to BPMN ones. The obtained result forms standard models for business processes with accepted semantics facilitating the interaction between a system engineer and a system modeler.

The control part is divided into physical control part or cyber control part. The physical and cyber parts represent respectively the physical and cyber entities. There is also need to represent the different interactions between the above participants. So, interactions connecting the pools are represented with message flows.

The interactions between activities ($Act(A_i)$ or $Act(A_j)$), triggered by events ($Event(A_i)$ or $Event(A_j)$), represent the behavior of a participant A_i or A_j in a given pool.

The following Figure 6.3 depicts a possible result of this transcription from CA-BRS to BPMN diagram.This BPMN model illustrates the behavior definition of CA_{CPS} during Scenario 1 execution. The control part of this example is divided to physical control part: $AgRW^1$ and $AgRW^2$, or cyber control part. The physical and cyber parts represent respectively the physical and cyber entities, as Doctor, PatientA and Conditions. There is also need to represent the different interactions between the above participants. So, interactions connecting the pools are represented with message flows. Interactions between activities ($Act(A_i)$ or $Act(V_j)$), triggered by Events ($Event(A_i)$ or $Event(V_j)$), represent the behavior of a participant A_i or V_j in a given pool.

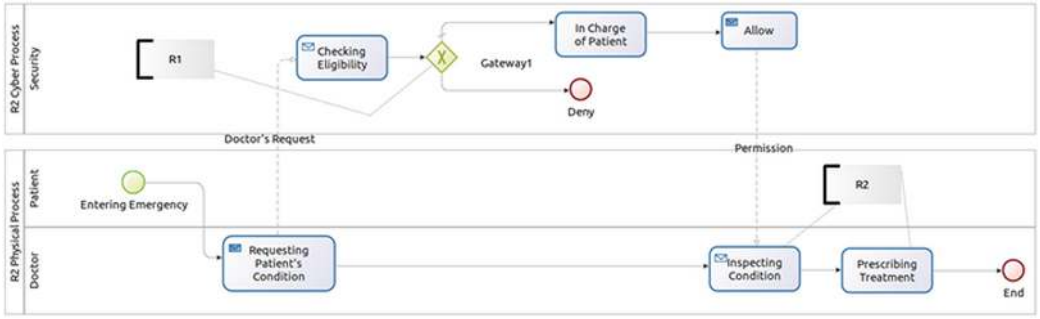


Figure 6.3: CA_{CPS} Behavior in terms of BPMN: Scenario 1

The given approach, translating CPS formal models to BPMN ones, serves as a standardized bridge for the gap between the CPS formal specification (CA-BRS model) and their implementation. Then, we apply successive CA-BRS to BPMN transformations in order to capture and analyze runtime behavior of CPS based on its execution traces. An initial BPMN model execution may reveal limits in the formal models, causing a return to the modeling phase and revision of the models. Any functional inconsistency, due to a faulty design of the CA_{CPS} behavior model, such as the presence of a deadlock situation, or an infinite loop or a situation of multiple terminations, may be detected when we execute the corresponding BPMN model of the specified system.

Table 6.3: Activities Perspective: BPMN based Semantics of CA_{CPS}

CA-BRS Virtual Dimension	**BPMN semantics**
CA-BRS	Pool Types: Physical, Cyber, Control Physical Agent, Control Cyber Agent
$CA_{CPS} = A_{Ph} \cup A_{Cy}$ • A_{Ph} • A_{Cy}	Virtual Participants: • Lanes of the Control Physical Agent Pool. Each Agent type A_{Phi} has its own lane A_{Phi} • Lanes of the Control Cyber Agent Pool. Each Agent type A_{Cyj} has its own lane A_{Cyj}
$V_{CPS} = V_{Ph} \cup V_{Cy}$ • V_{Ph} • V_{Cy}	Physical/Cyber Participants: • Lanes of a given Physical Pool. At each physical entity N_{phi} corresponds its lane N_{phi} • Lanes of a given Cyber Pool. At each physical entity N_{Cyj} corresponds its lane N_{Cyj}
$Host_{Ph} : A_{Ph} \rightarrow \in^{V_{Ph}}$	Connecting objects: between participants of lane A_{Ph} and those of lane V_{Ph}
$Host_{Cy} : A_{Cy} \rightarrow \in^{V_{Cy}}$	Connecting objects: between participants of lane A_{Cy} and those of lane V_{Cy}
CA-BRS Behavioral Dimension	**BPMN semantics**
$C = (B_{CPS}, H_{CPS})$	A given scenario of BPMN model (Collaboration diagram)
A Trigger rule γ_i of a control agent A_{Cyj} hosting in N_{Cyi},	Flow/Connecting objects: Behavior of a process A_{Cyj} in terms of Events: $Event(A_{Cyj})$ and activities: $Act(A_{Cyj})$. Each trigger rule represents a sequence flow γ_i
A Trigger rule γ_i (belonging to TR_{CPS}) of a control agent A_{Phi} hosting in N_{Phi},	Flow/Connecting objects: Behavior of a process A_{Phi} in terms of Events: $Event(A_{Phi})$ and activities: $Act(A_{Phi})$. Each trigger rule represents a sequence flow γ_i
A Rearrangement rule (in GR_{CPS}) of label β_i, • $\beta_i = [A_i, A_j]_x$: • $\beta_i = New(A_i, H_{CPS}, Event(A_i), Act(A_i))$, or $\beta_i = Des(A_i, H_{CPS}, Event(A_i), Act(A_i))$	Connecting objets: • Each rearrangement rule represents a message flow β_i between the two participants A_i and A_j • β_i permis to add (resp. destroy) a new virtual participant A_i to (resp. from) the collaboration diagram, its hosted entity (physical or cyber) should be given, as well as its events $Event(A_i)$ and activities: $Act(A_i)$.
• $\beta_i = Mig(A_i, H_{CPSi}, H_{CPSj})$	• In this case, β_i is used to change the hosting H_{CPSi} (location) of a virtual participant A_i to another H_{CPSj}. Conserving its activities and events sets
Controlled Reaction Rules of label α	BPMN models simulation, γ_i and λ_i are executed sequentially.

6.4 Security and Safety Properties Formal Analysis

Securing a CPS is extremely challenging. Due to the interconnected dynamics of a CPS, a failure in one component can have cascading effects across the entire system. Furthermore, a multi-pronged attack may exploit the weaknesses of the separate components of the system. While the individual vulnerabilities may not pose a serious threat on their own, when combined, they can have catastrophic consequences.Indeed, attacks on CPS can target not only the cyber domain, but also the physical domain, or even both. For example, if an attacker gains unauthorized access to a CPS network and steals data, that would be considered a purely cyber-based attack. However, if the attacker leverages that network access to take control of a physical actuator, the attack becomes a cyber-physical one. In addition to cyber-based and

cyber-physical attacks, it is also possible for an attacker to carry out purely physical attacks, such as the physical theft or damage of CPS equipment.

In this thesis, we focus on formally expressing and analyzing the security and safety of CPS. Our contribution is twofold:

1. We introduce an original approach to express these properties. Various studies in the literature [102] have independently examined the security and safety of CPS. We note that the safety of CPS refers to accidental risks originating from the CPS that could result in unacceptable consequences (such as human losses, heavy material loss, and nature damage). Contrariwise, its security is related to malicious risks (physical or cyber).

   Our contribution consists in the overlap between the CPS security and safety properties. Thus, the promising results obtained ensure the specification of more complex behaviors for this type of system, considering three possible scenarios:

   - Safety must be verified before security.
   - Security must be guaranteed before safety.
   - Safety and security can be considered jointly.

   Examples of these various situations will be examined and explained through case studies in the next chapter.

2. We develop a method for analyzing these properties using the GTS computational model, proposing specific agents whose behaviors ensure these properties within a CPS. Obviously, safety and security analysis remains an ill-defined field, devoid of formal theories, which complicates the risk analysis for CPS. Therefore, the integration of the GTS model with a multi-perspective analysis approach can offer valuable insights to analyze the adaptive behavior of CPS, taking into account these concerns.

Based on our review of the state of the art (Chapter 3) and the established categorization of attacks on CPS layers, our focus here is on the *communication layer (ACL rules)* and the *information layer*. Specifically, Confidentiality, Availability, and Integrity are considered, as they were defined by the CIA triad [103]:

- Confidentiality refers to the protection of data from unauthorized third parties.
- Availability refers to the data being accessible by authorized parties whenever needed.
- Integrity refers to the data being complete and uncorrupted.

However, one CPS attack may target one or more principles at the same time.

The advantage of our CA-BRS and GTS based framework is that all security and safety properties can be considered similarly. This framework takes into account the interaction between cyber and physical components to define CA_{CPS} model that control the cyber and the physical entities and their interactions too. Indeed, the adaptive behavior of CPS addressing both safety and security properties, can be naturally specified using the state space definition of the CA-BRS model. This behavior is influenced by agents' observations. Consequently, we include specific agents in the set of control agents CA_{CPS} to manage the safety and security aspects of all CPS components (virtual or physical).

$Ag_{security}$ and Ag_{safety}, are examples of such agents, which can be instantiated multiple times. These agents can exist in the states, for instance, $St_{CPS}(A_{security}) = \{secure, unsecure\}$ and $St_{CPS}(A_{safety}) = \{safe, unsafe\}$, respectively. They enhance the CA-BRS model of the CPS by providing additional virtual services.

The GTS is used to specify the CPS secure behavior by encoding specific computations into CRR rules or GR rules. Its corresponding execution trace represents the CPS behavior in which multiple independent processes (cyber entities) or physical components run simultaneously and interact with each other, requiring synchronization and control. It is worth noting that this path on the given GTS is unique, since the observation of the agents (Trigger Rules γ_i) according to their hosting, guides the rewriting of the CA-BRS configurations.

The execution traces generated by CRR rules are used to compute an adaptation strategy valid for a specific number of subsequent topological configurations of the CPS state space. Such a strategy forbids the execution of actions (λ_i) that lead to the violation of a security requirement, or alternatively perhaps enforces specific actions in order to mitigate it.

In the next chapter, we will demonstrate how the GTS model can evolve by executing controlled reaction rules (CRR) to define the dynamics of CPS case studies. We will explore concrete scenarios that highlight the significant correlations between security and safety and illustrate how to implement these in practice.

6.5 Conclusion

The main goal of this chapter was to formally describe and analyze security and safety of the high-level behaviors of CPS. The BRS extension with control agent entities allows to understand and model safe and secure CPS, while addressing three essential factors, namely the structure change (the arrival or departure of a physical or cyber entity in the CPS), the agent evolution (transition from one state to another), and finally, how does the first change affect the second one and vice versa.

In this context, we mitigate the computational complexity of the CPS model by associating specific types of agents that manage the security and safety aspects of CPS entities. The evolution of these agents' states forms their observations, which will guide the trace execution in the GTS.

In the next chapter we will focus on the implementation aspects of CA-BRS and its GTS execution model, we encode the semantic interpretation of CA_{CPS} in the Maude and BPMN languages through real case studies.

Chapter 7

Case Studies and Implementation

7.1 Introduction

As CPS increasingly permeate every aspect of modern systems, it is important to define, in addition to the theoretical foundations for modeling and analyzing them, practical techniques for executing their models and adapting them. This chapter outlines some directions towards the objective presented. The previous chapters introduced the CA-BRS theoretical foundation, which modeled CPS at different levels of abstraction and focused on their adaptation and analysis. This foundation is then implemented using the Maude language, with a focus on formal analysis techniques around this language. From a technical point of view, greater decentralization is needed to account for the inherently distributed nature of these systems. Runtime models help accommodate continuous and intelligent changes in CPS environments. Furthermore, it is important to recognize that openness brings threats, so CPS also requiress adaptive methods to protect themselves. Finally, the proposal of two application examples is essential, not only to evaluate the proposed approach but also to demonstrate the careful balance that should exist between the theoretical and practical knowledge of the physical, cybernetic, and safety aspects of CPS.

In the first case study, we illustrate our approach through a Medical CPS. We focus on two important issues: the multiaspect modeling of all the CPS constituents with the CA-BRS formalism and the execution of GTS to control

CPS intelligent behavior.

The second case study concerns the application of CA-BRS model to specify the routing problem in communication networks. Specifically, we are interested in modeling the the Access Control L(ACL),ACL), which is an important technique in ensuring network-wide connectivity and security. In fact, with the emergence of new technologies like cloud computing and fog computing, wide area networks (WANs) are constantly evolving in terms of size and complexity. As a result, the access control list (ACL) rules governing these networks are becoming increasingly complex. Thus, integrating formal methods into ACL design enhances security, reliability, and compliance by rigorously analyzing access control policies and preventing misconfigurations or vulnerabilities at the CPS network layer.

Despite the formal nature of our proposed model (CA-BRS) and its computational model (GTS), the experiments show promising results in which these models can be used effectively to deal with the security analysis to real-world CPS with known security vulnerabilities and threats.

7.2 Case Study 01: SSM-CPS

The healthcare industry has gone through various transformations, i.e., from Healthcare 1.0 to 4.0. The later one (Healthcare 4.0) keeps the patient's record in the centralized Electronic Health Record (EHR) system to monitor the patients' health records and delivers the uninterrupted services to them in real-time [104]. Patients' health can be monitored through wearable devices (WDs) and implantable medical devices (MDs). WDs are equipped with various healthcare sensors to measure blood pressure, heart rate, temperature, and glucose level of the patients remotely [104]. It helps to understand the patients' behavior. Thus, CPS may be considered an enabling key technology of Healthcare 4.0.

In this example, we emphasize the importance of security and privacy in the case of Medical CPS (M-CPS). SSM-CPS (for Secure Smart Medical Cyber Physical System), refers to safety-critical interconnected system that analyzes all the information gathered from medical devices, infers the patient's health condition, and starts treatments issuing information to doctors or directly to medical actuators. A proposed SSM-CPS architecture (see Figure 7.1) consists of three parts: cyber part, physical part, and cyber-physical interaction components. These three parts may closely be connected via in-

ternet including wireless connection types. Patient may be located in home or in health-center (Hospital). When the health parameter value reaches their critical bound, then through actuators the patients are treated inevitably, according to the degree of danger, ambulances and responders (drone in this case) will intervene appropriately, hence response time is instant.

Figure 7.1 gives the architecture of our SSM-CPS example. It models an intelligent hospital managment system which is expected to change the way of how people interact with the physical world and use healthcare applications such as monitoring and decision support systems.

The architecture is divided into three parts based on the usage of healthcare applications. These three parts may closely be connected via internet including wireless connection types. The first part in the architecture is the communication part, then the computation (Cyber) part, and the final part for the proposed CPS healthcare architecture is the resource management (Physical) part.

The SSM-CPS constituents are distributed over three layers: Cloud, Fog and Edge as indicated in Figure 7.1.

- In the Edge Layer is located the user's (patient) devices and the actuators (like responders and ambulance).
- In the Fog Layer (Health center) resides the most cyber functionalities of the SSM-CPS, as well as the received monitoring data, which will be sent to the cloud for further analysis.
- Finally, in the Cloud Layer, the collected information will pass by the following possible actions: Archiving, Business intelligence, Data mining, etc.

In our case study, SSM-CPS of Figure 7.1, continuously monitors the patient's health parameters such as blood glucose level, blood pressure level, body temperature level, and heart beat rate. Patient may be located in home or in health center (Hospital).When the health parameter value reaches their critical bound, then through actuators the patients are treated inevitably, according to the degree of danger, ambulances and responders (drone in this case) will intervene appropriately, hence response time is instant.

SSM-CS benefits the patients, doctors, and clinical assistants in reducing the over head of assisting all the patients during the inconvenience period.

However, we can notice that this SSM-CPS example has a complex architecture, it integrates information technologies, real-time control subsystems, physical components and human operators to influence physical processes by means of cooperative and (semi) automated control functions. Thus, a proper investigation is needed on the CPS security issues to make sure that systems are working safe. The migration towards digital control systems creates new security threats that can endanger the safety.

7.2.1 Illustrative Scenarios

To investigate the issue of security and safety for SSM-CPS, we apply the proposed CA-BRS model. We consider the following scenarios (see Table 7.1) which tackle the challenge of modeling distributed and adaptative behaviors of this SSM-CPS to ensure these two properties (security and safety).

We distinguish in this example, some situations involved in guarantying safety or security of individual entities separately (situations from 1 to 9) from those where the alternation between safety and security rules is essential, as in the other more complex situations 10, 11, 12 and 13, being convinced that safety and security should be examined and treated jointly. According to the ISO 31004 standard [102, 105], in this example, **Safety** refers to accidental risks arising from the system that may lead to unacceptable consequences, such as human casualties, significant material damage, or environmental harm. **Security**, on the other hand, pertains to malicious risks, specifically attacks. These attacks can be carried out either physically, through direct access to the SSM-CPS, or via cyber means, through local or remote system access.

Specifically, in this case study we consider the following running scenario examples summarized in Table 7.1, where the data transfer requires some sorts of authentication and encryption which can be easily done by VPN.

The following section shows our solution feasibility for modeling this SSM-CPS example based on $CA-BRS$ model; we describe first its architectural part, then we identify some control agents to monitor its cyber and physical entities. The main objective here is to show how this provided formal model will consider, in addition to the heterogeneity of the CPS components, their dispersion, their connected dynamics, their self-adaptive and secure property.

7.2.2 SSM-CPS Formal Modeling with CA-BRS

We show in Figure 7.2 how to represent a simple model of SSM-CPS in order to motivate the need for bigraph-based semantics for cyber (V_{Cy}) and physical (V_{ph}) entities. Its corresponding B_{CPS} is defined to model not only the location of CPS components, with the $Prnt_{CPS}$ function, but also their connectivity thanks to the function $link_{CPS}$ (see Figure 7.2). We note that for simplifying the figure, we do not consider all SSMCPS constituents.

Physical/cyber SSM-CPS Constituents:

In our case, we decouple the physical entities of CPS, e.g. RESPONDERS, Doctor's PC, Nurse's PDA, etc. to cyber elements, e.g. Archiving, Data mining, etc. Links between the two element types allow us to track relationships between entities to represent, for instance, a relationship between the Ambulance1 and FOGNODE on the one hand, and the Medicine drone1 on the other hand (link2).

- Thus, for this example, V_{Ph} and V_{Cy} are defined in Table 7.2. Also, the kinds of nodes and their number of ports (arity) constituting the signature K_{CPS} of B_{CPS} are depicted in Table 7.2.

- The link graph of our example may contain hyper-edges Link1. Link2, Link3 representing ports connection of various nodes or regions.

- Regions HEALTH CENTER, HOME, CLOUD and holes 0, 1, 2, 3 enable composition of placing graphs, i.e. a hole 0 can be replaced by a region of another bigraph using the composition operator.

SSM-CPS Control Agents:

Subsequently, we define the CA_{CPS} set of CA-BRS model refering to a set of intelligent agents allowing to control the above structural entities of SSM-CPS.

- Each type of entity (physical or cyber) may host a distinctive type of control agents; A_{Ph} to manage nodes of V_{Ph} and A_{Cy} for nodes of V_{Cy}. So, we define hierarchically $CA_{CPS} = A_{Ph} \cup A_{Cy}$, such that,

$$A_{Ph} = \{Ag_{Act}, Ag_{Sen}, Ag_{A\&S}, Ag_{net}, Ag_{IU}, Ag_{R}, Ag_{Am}\}, \text{ and}$$
$$A_{Cy} = \{Ag_{Ser}\}.$$

- We note also that these types of agents may have several instances according to the considered system behavior, and thus each instance can be in various states defined by $St_{CPS}(A_i)$ as shown by Table 7.3.

- As indicated by Figure 7.3, the control agent Ag_{Act} may have two instances (Ag_{Amb} and Ag_{Dro}) to observe and manage respectively the physical (actuator) entities: Ambulance and Drone on which these agent instances host.

This CA-BRS configuration (an example is in Figure 7.3) deals with structural and virtual aspects of the corresponding SSM-CPS, consisting of physical or cyber entities in which agents of CA-BRS are hosting; it constitutes a possible state of the CPS complex behavior, that we will explain in more details in the following subsection.

7.2.3 Analyzing Security Properties with GTS

The SSM-CPS dynamic behaviors, involve close-interactions between the two CA-BRS dimensions -Structural and Virtual- which have been previously explained. Thus, we explain how this continuously running SSM-CPS, interacts with the environment and humans, in ways that can be fully anticipated at design time, and how it may assure and improve the safety and security properties of these CPS components.

SSM-CPS Dynamic Behavior:

Generally, to describe the potential behavior of the SSM-CPS, we refer to the GTS definition which consists of states (or configurations) and transitions between states. Transitions in this case coincide mathematically with a set of CRR rules as well as a set of GR rules,

- Configuration (or state) of the CPS considered example (as depicted in Figure 7.3), defines the structural part of the CPS combined with the current hosting function (H_{CPS}) applied to each agent instance. For instance,

  $H_{CPS}(Ag_{Amb}) = Ambulance1$, knowing that Ag_{Amb} is of type Ag_{Act} (belonging to A_{Ph}) and $Ambulance1 \in Host_{CPS}(Ag_{Amb})$,

  Similarly, $H_{CPS}(Ag_{Arc}) = Archiving$, $H_{CPS}(Ag_{Dro}) = Drone2$, etc.

  The CA-BRS$_{CPS}$ states can also be specified algebraically, the given configuration of this example is shown in Figure 7.4.

Knowing that Si, i = 1..10 are possible state values of the corresponding agent instances. In conclusion, Figure 7.6 illustrates examples of possible configurations of SSM-CPS behavior, such as $C1$, $C2$, etc., resulting from its evolution.

- In the following, we define and explain two CRR rules presented in Figure 7.5, specifying dynamic and adaptive behavior of the SSM-CPS example regarding, respectively, Situation 1 and Situation 6 of its scenario (indicated in Table 7.1). These rules serve to conceive the corresponding GTS states space to specify dynamic and adaptive behavior of safe and secure SSM-CPS. It is worth noting that traditional transition systems are insufficient to represent and reason about the interactions and dependencies between concurrently executing CPS components. GTS is proposed to prevent SSM-CPS behaviors based on the parallel composition between the observations of the agents (γ_i) and based on the partition of the physical structure where each agent operates in its own region (using rules λ_i and β_i.

SSM-CPS Properties Analysis:

In general, the behavior of SSM-CPS dealing with security and safety properties may be naturally specified by CA-BRS, we just enrich the CA_{CPS} set by supplementary appropriate agent types: $Ag_{security}$ and Ag_{safety}, that will control the security and the safety of any entity (virtual or physical) of the considered system. These agents may be respectively in the states $St_{CPS}(Ag_{security}) = \{secure, unsecure\}$ and $St_{CPS}(Ag_{safety}) = \{safe, unsafe\}$. They enhance the CA-BRS modeling of the given system by providing it with other virtual services.

1. The first CRR rule (related to Scenario 1 in Table 7.1) specifies the concurrent behavior of SSM-CPS dealing with the security of the patient Smart Watch. It captures the interactions and dependencies between involved rewriting rules ($\gamma_i, i = 1\ to\ 3$ and $\lambda_j, j = 1\ to\ 4$) and provides a clear picture of how these rules are executed concurrently. In addition, possible GR rules (β_1 in this case) are proposed to define agents mobility in order to rearrange their execution.

2. In a similar way, we define the second CRR rule defining GTS evolution of the SSM-CPS while executing Situation 6 (see Table 7.1) dealing with the security of the Medicine Drone and safety of the smart watch.

In addition, with these promising results guaranteeing the security and safety of certain entities of the SSM-CPS, we can specify more complex behavior of this type of system, while considering the security and the safety properties jointly.

3. In some cases, safety must be verified before security (Situation 10 in Table 7.1), in other cases security must be guaranteed before safety (Situation 12) and in a third case study, the two properties can be considered jointly (Situation 11). We explain in Figure 7.7 some rewriting rules (γ_i, λ_j, and β_k) that allow the adaptive and secure behavior modeling of SSM-CPS in Situation 10. In Figure 7.8, we conceptualize the corresponding execution trace of the GTS computational model, associated with our formalism (CA-BRS), identifying two types of rules (CRR and GR) to deal with the state evolution of the SSM-CPS example considered in Situation 10. In this rewriting system, complex structures are built from simple following specific rules. These structures can represent different shapes, patterns, or components, and they can interact with neighboring structures based on predefined rewriting rules.

   It is worth to note that this path on the given GTS is unique, since the observation of the agents (Trigger Rules γ_i) according to their hosting (H_{iCPS}), guides the rewriting of the CA-BRS states.

Similar CRR definition may be given to formalize the other security properties of the SSM-CPS example. We note that modeling the behavior of SSM-CPS as GTS allows for performance analysis, such as deadlock detection, workload balancing, and response time estimation. These analyses can be crucial to ensure that the SSM-CPS runs efficiently and meets its design goals. we will show in Section 7.4 and through the Maude system, the implementation of these execution traces and the analysis of certain relevant properties.

7.3 Case Study 02: I4.0-CPS

We conduct case studies representing diverse CPS applications to demonstrate the effectiveness of the proposed formal models (CA-BRS and GTS) in identifying and mitigating security risks constitutes the main objective of

this chapter. In pursuit of this objective, we focus on the use of our formalism to analyze the security of the network dimension in any CPS and specifically in Industry 4.0.

The rapid advancement of technology has propelled the modern industry into the transformative era of CPS-based Industry 4.0. This evolving digital landscape, defined by constant connectivity, also brings inherent risks of cyber-attacks and unauthorized access. In this section, we present a case study (I4.0-CPS) that illustrates the continuous control of the execution of actions in the industrial environment by exploiting the ACL rules and the proposed CA-BRS formalism. We present a fine-grained formal usage control model that meets security and confidentiality needs. We develop formal models and design practices, implemented as Maude modules, that can adapt to various requirements and specifications for routing problems within I4.0-CPS.

Indeed, as CPS-based Industry 4.0 is becoming integrated, a malicious user can have tremendous opportunities to infiltrate networks, steal sensitive information, inject cleverly crafted false data into measurements, or overwhelm networks with fake packets. Such malicious activities can prevent legitimate requests or even mislead the control center to make erroneous decisions. This case study addresses the specification of an Access Control List (ACL) that will be used in a WAN routing problem in the context of Industry 4.0 applications. ACL acts as a set of rules designed to regulate network traffic and mitigate network attacks.

7.3.1 Scenario Example

In general, ACLs proposed by (cisco [106]) and incorporated into the internet according to [107], are used to filter traffic based on the set of rules defined for the incoming or outgoing of the network. They use both source and destination IP addresses. Thus, we can mention which IP traffic should be allowed or denied upon some characteristics (source, destination, protocol). The following rule examples indicate which action will be taken if a packet has the indicated source and destination using a given protocol.

1. Deny 192.168.60.0 192.168.61.1 : to state that packet going from the network “192.168.60.0” (we consider that this IP address belong to a network) to the machine “192.168.61.1” (we consider that this IP address belong to a machine) should be blocked.

2. permit 192.168.60.3 192.168.61.2 80: to state that HTTP traffic (uses port n° 80) going from the machine "192.168.60.3" (w.c.t.t.i.a.b.t.m) to the machine "192.168.61.2" (w.c.t.t.i.a.b.t.m) should be allowed.

In the generic example depicted in Figure 7.9, an ACL network is used to ensure network-wide connectivity and security. However, as the networks involved in modern CPS are constantly evolving in size and complexity, ACL rules are becoming increasingly complex. This poses a significant challenge to the update process of ACL configurations within routing protocols. We utilize CA-BRS formalism for modeling the network ACL spaces as well as its dynamic and adaptive change. Formal reasoning facilities are implemented adopting Maude-based modules.

Obviously, routing is achieved by performing the control actions first and then the routing action. While our model is flexible and expressive, we chose to cut some corners to simplify the example, to focus on the explained matter, to abstract some network concepts, and to show only how CA-BRS can model the control mechanism.

R1 ACLs:
1. permit A ALL E
2. permit B A
3. permit 1 2
4. deny z A, B
5. deny 1 x

- **Routing Exception:** permit 1 E

R2 ACLs:
1. permit C E
2. permit E C
3. deny E A, B
4. permit A, B E

7.3.2 I4.0-CPS Topology Definition with CA-BRS

In Figure 7.10, we propose the CA-BRS model to specify the network topology of our running example. We first note that this network model is represented by a region that can be inserted (composed) into any other large

region for modeling a complete CPS. In addition, both structural and virtual aspects are supported by this model.

Nodes of CA-BRS represent Routers, Sub-nets, etc. Edges allow to join these nodes for establishing possible traffics in the network.

The Control Agents in the CA-BRS are designed to control and manage the routing in this network.

7.3.3 Formal Modeling Secure aware Routing

This section outlines the formal modeling approach for secure-aware routing, ensuring that routing decisions consider security constraints. It would be helpful to clarify the specific modeling rules: CRR and GR, used to run this dynamic routing process. Table 7.4 explains concretely these examples of ACL rules. A set of the involved CA-BRS configurations (states) is depicted in Figures 7.11, 7.12, 7.13.

7.4 Maude based Implementation of CA-BRS

This section provides technical details about the encoding of our BRS extension in Maude. In fact, we note that some tools and software environments exist in the literature, designed to manipulate BRS in a practical way. We can cite:

- BPL Tool [108],
- BigRed [109],
- Dbtk [110],
- BigMC [111] and
- BigraphER [112].

Each of them represents a possible BRS implementation for a specific purpose. For instance, BPLTool, BigRed and Dbtk are dedicated to manipulate, simulate and visualize ordinary BRS, unlike BigMC and BigraphER which are intended for the model-checking based analysis and verification of the systems in question.

Existing tools do not support naturally the extension of BRS that we are developing (CA-BRS). We have tried to exploit the BigraphER tool in a previous contribution; although the expression of the model was simplified by considering the agents as ordinary nodes, its execution led to a very large or even unreadable states space, as depicted in Figure 7.14.

Consequently, we propose a CA-BRS framework based on some Maude modules to execute and formal analyze the proposed CPS specifications. We opt for the transcription of our extended bigraphical model to Maude for the following reasons:

- Both Maude and BRS formalisms are defined on the basis of mathematical categories, the formal interpretation of this transcription could be quite natural and may have an important contribution to the execution of bigraphical models,
- The agent virtual entity added to BRS is better specified in terms of state and change of states in Maude, So far, no existing tool around BRS may support explicitly the behavior aspect of BRS entities (nodes), especially the sorting function and the Agent nodes,
- Maude's strategies can be exploited to direct the rewriting of rules in the conditional evolution of the states of a CPS specified with GTS,
- Maude system incorporates some analysis tools as the Church-Rosser checker (CRC), the inductive theorem prover (ITP), the LTL model-checker, the model checker invariants.

Figure 7.15 provides a comprehensive overview of the Maude modules utilized to encode the CA-BRS model. As illustrated, the static, virtual, and behavioral aspects of the CA-BRS model are defined independently through a collection of Maude modules: BIGRAPH, BRS, CONTROLAGENT, CABRS. These modules contain declarations for specific sorts, operators, and rewrite rules. In the subsequent sections, we will delve into a detailed description of each of these CA-BRS aspects.

7.4.1 Maude code for Static and Virtual Structures

Our implementation of CA-BRS consists of two fundamentals steps. First, we define suitable rewriting logic algebraic terms that specify CA-BRS elements

(Cyber, Physical, and Virtual) using two distinctive modules, **CABRS-STATIC** and **CABRS-VIRTUAL**. Then we implement CA-BRS rules (observations, Actions, Trigers) using Maude rewriting rules via the module **CABRS-BEHAV**. To encode the nesting structure of bigraphs nodes and sites using Maude terms; we use their identifiers i.e., each identifier of a node or a site is concatenated with identifiers of its parents. More precisely, the following rules (in Figure 7.16) expressed in Backus-Naur form (BNF) describe the adopted syntax for CA-BRS.

The Maude code in Figure 7.17 and Figure 7.18 describes the fundamental physical and cyber entities within the CPS, specified with the CA-BRS syntax part, as soon as its virtual part. This CPS model forms the key concepts upon which its dynamic behaviors are contextualized and managed.

To return to our CPS example (SSM-CPS) in Section 7.2, we develop the module depicted in Figure 7.19 to associate formal semantics based on rewriting logic to each constituent (cyber, physical, or virtual).

The SSM-CPS module in Figure 7.19 shows how to use the proposed Maude implementation of CA-BRS to specify the static part of the SSM-CPS example. At first, the CABRS-STATIC module defining the reusable structure of CA-BRS components is imported; then, specific constructor operations introduce the elements of the SSM-CPS. The unique equation of the SSM-CPS module brings together all constructor terms to form the global SSM-CPS structure as an algebraic term. Our approach is well suited for intelligent distributed systems, formed of components involving computation and communication. In the following, we illustrate its potentialities through the formal specification of the relevant security property.

7.4.2 Dynamic Behavior Support in Maude

The behavioral specification of our case study is done using the CA-BRS Control Reaction Rules (CRR) set. We show the transcription of these possible secure and intelligent behaviors in terms of Maude rewriting rules and strategies. The Maude module SSM-CPS of Figure 7.19 may include also rewrite rules that implement BRS reaction rules, observations and of GTS.

7.5 Conclusion

Cyber-physical systems are an extension of traditional software systems to the physical world, which are different from the construction and composition of traditional software systems. The integration of information space and physical space makes the evolution of such systems more complex.

The proposed CA-BRS model was implemented on the basis of Maude langage in order to test and show that it can contribute to the construction of safe and secure medical CPS, as soon as other complex CPS.

A natural consequence of this encoding approach consists of providing an executable computation model and a formal analysis of some inherent properties. We choose to use the Maude rewriting enginer to execute and prototype the considered examples of CPS.

The operational semantics of the proposed CA-BRS and GTS models is defined in terms of transition systems and implemented using rewriting logic. Rewrite rules and strategies have been used to naturally represent conditional reaction rules (CRR) of CA-BRS.

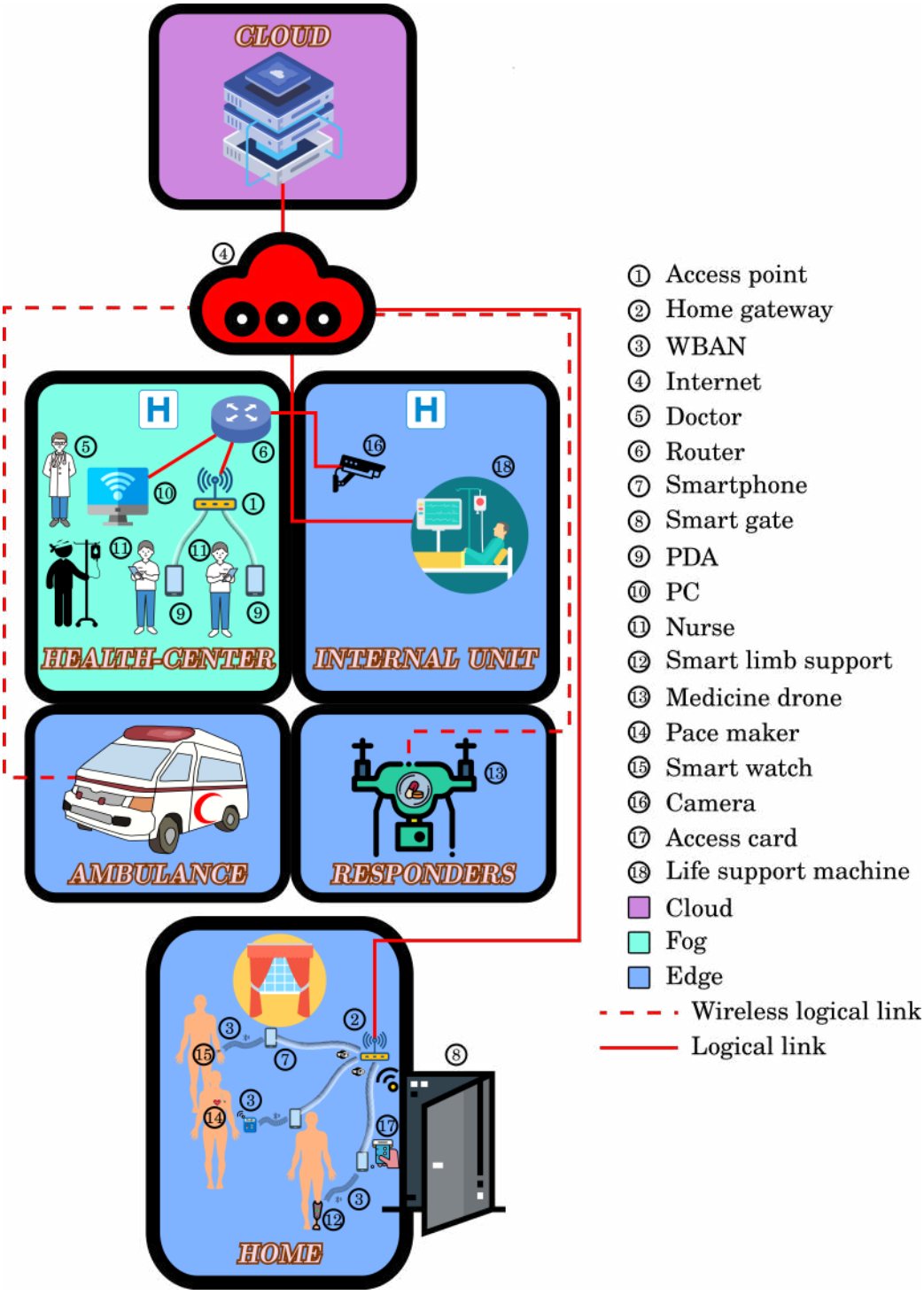


Figure 7.1: SSM-CPS Architecture

Table 7.1: SSM-CPS Secure and Safe Behavior Example

Possible Situations in SSM-CPS Behavior	Description
1: security	A smart watch takes health measurements for a diabetic patient, so his smartphone may notify the action type to be taken: insulin, food, emergency, etc.
2:security	A pacemaker connected to the phone helps determining the remaining life of the battery or any condition occurring to the patient enabling a real time response.
3:security	A smart artificial limb also connected to the smart phone serves to automatically adapt the patient move to the various environment changes.
4:security	The home access is protected by a smart gate requiring an access card.
5:safety	In the internal unit the machine auto injects drugs if the dose is considered as safe, if not the approval of the doctor is needed.
6:security	The drone facilitates the delivery of needed medicine in the case of a fast emergency.
7:safety	The ambulance is in standby waiting for any call, reporting the data to the health center in any time.
8:safety	The doctor receives the information about his patient through his computer and can then give the nurses corresponding orders.
9:safety	The nurses use their Personal Digital Assistant (PDA) to know their patient's status and acts according to the recommendations of the doctors.
10: safety	In a life-threatening circumstance the gate should allow nurses enter the building to execute their work, so in this case safety of the patient (The smart watch indicates "bad" as notification) is ensuring before security property, guarded by the smart door.
11: security + safety	All data transfers are protected by the hospital VPN (the company), so no needs to check data integrity (low tempering probability) safety is complementary with security here.
12: security	All data sources must be authenticated by the company, so in this case security is preferred over safety.
13:+ security + safety	In case of VPN unavailability or (HTTPS) we can regress to less secure solution or (HTTP) to ensure safety

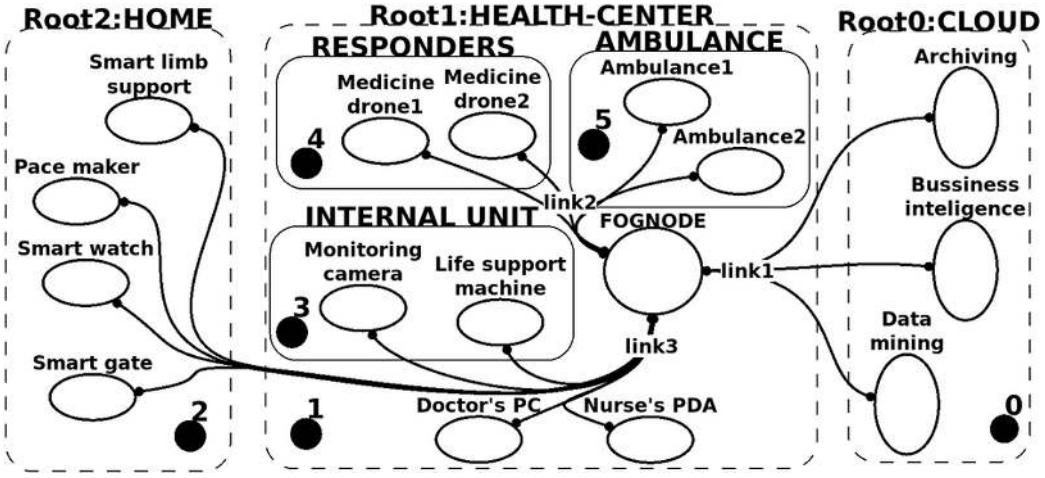


Figure 7.2: B_{CPS} for SSM-CPS Structural Part

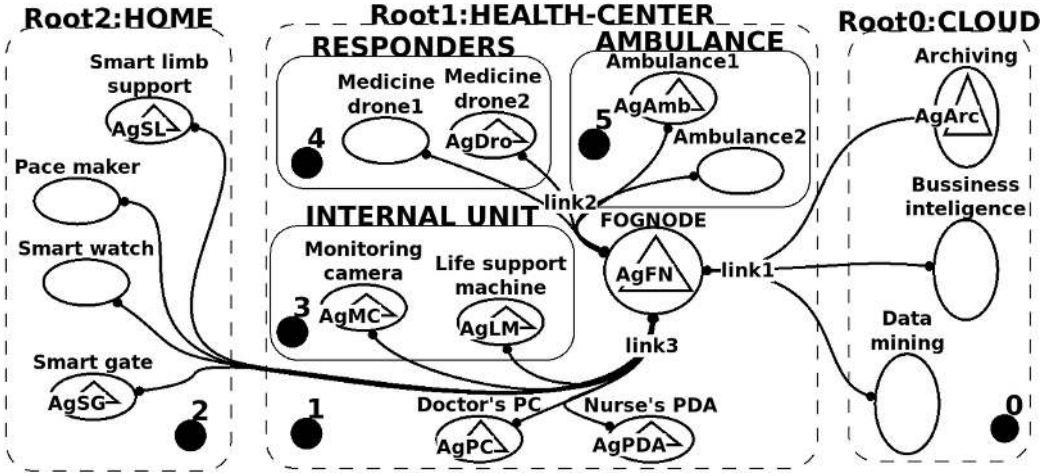


Figure 7.3: An SSM-CPS Configuration

$$C = Home[SmartLimbSupport_{link2}\{Ag_{SL}\&S1\}|PaceMaker_{link3}|SmartWatch_{link3}|SmartGate_{link2}\{Ag_{SG}\&S2\}|d2]||$$
$$\text{HealthCenter [Responders.}Medicinedrone1_{link2}|Medicinedrone2_{link2}\{Ag_{Dro}\&S3\}|d4\text{ |}$$
$$\text{Ambulance}Ambulance1_{link2}\{Ag_{Amb}\&S4\}|Ambulance2_{link2}|d5\text{ |}$$
$$\text{InternalUnit}MonitoringCamera_{link3}\{Ag_{MC}\&S5\}|LiveSupportmachine_{link3}\{Ag_{LM}\&S6\}|d3\text{ |}$$
$$\text{FogNode}_{link2link3link1}\{Ag_{FN}\&S7\}|NursePDA_{link3}\{Ag_{PDA}\&S8\}|DoctorPC_{link3}\{Ag_{PC}\&S9\}|d1]||$$
$$\text{Cloud [Archiving}_{link1}\{Ag_{Arc}\&S10\}|BusinessIntelligence_{link1}|DataMining_{link1}|d0]$$

Figure 7.4: An Example of The CA-BRS$_{CPS}$ States Algebraic Specification

Table 7.2: Physical and Cyber components of SSM-CPS: Structural Part

M-CPS Structural Elements	$\mathbf{Controls = K_{CPS}}$	Arity	$\mathbf{Sorts = \Theta_{CPS}}$
Physical Entities	Ambulance1	1	**Act**
–	Ambulance2	1	**Act**
–	Drone1	1	**Act**
–	Drone2	1	**Act**
–	Doctor PC	1	**Sen**
–	Smart watch	1	**Sen**
–	Monitoring camera	1	**Sen**
–	Smart limb support	1	**A&S**
–	Pace maker	1	**A&S**
–	Smart gate	1	**A&S**
–	Life support machine	1	**A&S**
–	Nurse PDA	1	**A&S**
–	Fog Node	3	**Net**
Cyber entities	Archiving	1	**Ser**
–	Business intelligence	1	**Ser**
–	Data mining	1	**Ser**
Physical Entities	Internal Unit	0	**IU**
–	Responders	0	**R**
–	Ambulance	0	**A**

Table 7.3: Possible Agents controlling SSM-CPS entities: Virtual Part

Agent Types	Agent Instances	Possible Attributes for $\mathbf{St_{CPS}}$	$\mathbf{Host_{CPS}}$
Ag_{Act}	Ag_{Amb}	$\{Active, Sending\}$	$\{Ambulance1, Ambulance2\}$
	Ag_{Dro}	$\{Active, KnowingLoc, SafeMedicine, Emergency\}$	$\{Drone1, Drone2\}$
Ag_{Sen}	Ag_{PC}	$\{Receiving, Sending\}$	$\{DoctorPC\}$
	Ag_{SW}	$\{Warning, Sending, Receiving\}$	$\{Smartwatch\}$
	Ag_{MC}	$\{Sending\}$	$\{Monitoringcamera\}$
$Ag_{A\&S}$	Ag_{SL}	$\{ChangingEnv, Adapting, Sending\}$	$\{Smartlimbsupport\}$
	Ag_{PM}	$\{StatueBattery, Sending\}$	$\{Pacemaker\}$
	Ag_{SG}	$\{AuthorizedAc, Emergency, Closed\}$	$\{Smartgate\}$
	Ag_{LM}	$\{Injecting, Statue, Dose\}$	$\{Lifesupportmachine\}$
	Ag_{PDA}	$\{Receiving, Emergency, Working\}$	$\{NursePDA\}$
Ag_{net}	Ag_{FN}	$\{Connected\}$	$\{FogNode\}$
Ag_{Ser}	Ag_{Arc}	$\{Disabled\}$	$\{Archiving\}$
	Ag_{BI}	$\{Analytics, Consulting, Implementation\}$	$\{Businessintelligence\}$
	Ag_{DM}	$\{Monitoring, Analysis\}$	$\{Datamining\}$

Scenario Description	A smart watch takes health measurements for a diabetic patient, so his smartphone may notify the action type to be taken: insulin, food, emergency, etc.	The drone facilitates the delivery of needed medicine in the case of a fast emergency
$\mathbf{Ag_{security}}$ **hosting**	Smartwatch	Drone1
$\mathbf{Ag_{safety}}$ **hosting**	–	Smartwatch
CRR rule	$C1 \xrightarrow[\lambda_1;\lambda_2]{\gamma_1} C2 \xrightarrow[\lambda_3;\lambda_4]{\gamma_2/\gamma_3} C3$ Where: $C1 = (B1, H1), C2 = (B2, H1)$	$C'1 \xrightarrow[\lambda_1;\lambda_2]{\gamma_1/\gamma_2} C'2 \xrightarrow[\lambda_3]{\gamma_3} C'3$ Where: $C'1 = (B'1, H'1), C'2 = (B'2, H'1), C'3 = (B'3, H'1)$
Rules $\gamma_\mathbf{i}$	γ_1 : $Smartwatch\{Ag_{SW}\&(nowarning, nosending, noreceiving)\}$ $\rightarrow$ $Smartwatch\{Ag_{SW}\&(warning, sending, noreceiving)\}$ γ_2 : $DoctorPC\{Ag_{PC}\&(nosending, noreceiving)\}$ $\rightarrow$ $DoctorPC\{Ag_{PC}\&(sending, receiving)\}$ γ_3 : $Smartwatch\{Ag_{SW}\&(warning, sending, noreceiving)\}$ $\rightarrow$ $Smartwatch\{Ag_{SW}\&(warning, sending, receiving)\}$	$\gamma_{r_1} = \gamma_1$: $Smartwatch\{Ag_{SW}\&(nowarning, nosending, noreceiving)\}$ $\rightarrow$ $Smartwatch\{Ag_{SW}\&(warning, sending, receiving)\}$ γ_{r_2} : $MedicineDrone\{Ag_{security}\&(unsecure)\}$ $\rightarrow$ $MedicineDrone\{Ag_{security}\&(secure)\}$ γ_{r_3} : $MedicineDrone\{Ag_{Dro}\&(inactive, known, unsafe, emergency)\}$ $\rightarrow$ $MedicineDrone\{Ag_{Dro}\&(active, known, unsafe, emergency)\}$
Rules $\lambda_\mathbf{j}$	$\lambda_1 : B1 \rightarrow B11$, create a link3 between Smartwatch and FogNode $\lambda_2 : B11 \rightarrow B2$, create a link3 between FogNode and DoctorPC $\lambda_3 : B2 \rightarrow B22$, destroy a link3 between FogNode and DoctorPC $\lambda_4 : B22 \rightarrow B1$, destroy a link3 between Smartwatch and FogNode	$\lambda_{r_1} : B'1 \rightarrow B'11$, create a link2 between MedicineDrone and FogNode $\lambda_{r_2} : B'11 \rightarrow B'2$, create a link3 between FogNode and Smartchwatch $\lambda_{r_3} : B'2 \rightarrow B'3$, Destroy a link2 between between MedicineDrone and FogNode
Rules $\beta_\mathbf{k}$	$C1 \xrightarrow{\beta_1} C3$, β_1: $Des(Ag_{SS}, Ag_{Security}, Smartwatch)$: Where: $C3 = (B1, H2)$	$C'3 \xrightarrow{\beta_{r_1}} C'4$, β_{r_1}: $New(Ag_{SS}, Ag_{Safety}, Smartwatch)$: Where: $C'4 = (B'3, H'2)$

Figure 7.5: CRR rules Defining SSM-CPS Secure Behavior

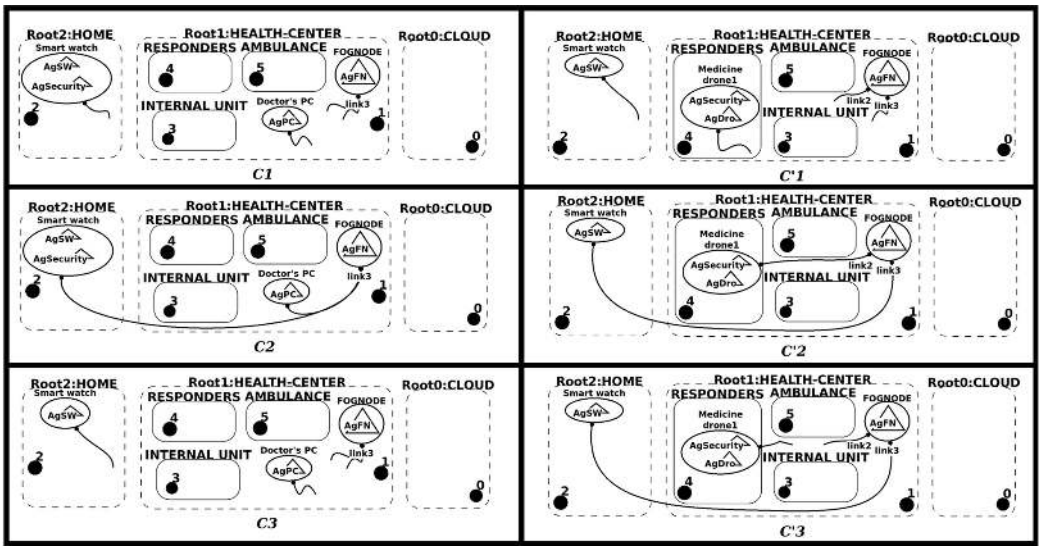


Figure 7.6: Possible Configurations C_i(i=1 to 6) involved in the GTS example

$\mathbf{Ag_{security}}$ **hosting**	Smartgate
$\mathbf{Ag_{safety}}$ **hosting**	Smartwatch
Trigger Rules γ_i	γ_1 : $Smartwatch\{Ag_{SW}\&(warning, sending, noreceiving)\}$ $\rightarrow Smartwatch\{Ag_{SW}\&(bad, sending, receiving)\}$ γ_2 : $NursePDA\{Ag_{PDA}\&(noreceiving, normal, working)\}$ $\rightarrow NursePDA\{Ag_{PDA}\&(receiving, emergency, working)\}$ γ_3 : $DoctorPC\{Ag_{PC}\&(noreceiving, nosending)\}$ $\rightarrow DoctorPC\{Ag_{PC}\&(receiving, sending)\}$ γ_4 : $Ambulance1\{Ag_{Amb}\&(inactive, nosending)\}$ $\rightarrow Ambulance1\{Ag_{Amb}\&(active, sending)\}$ γ_5 : $SmartGate\{Ag_{SG}\&(noauthorizedAc, Emergency, Closed)\}$ $\rightarrow SmartGate\{Ag_{SG}\&(noauthorizedAc, Emergency, unclosed)\}$
Action Rules λ_j	$\lambda_1 : B1 \rightarrow B12$, create a link3 between FogNode and Smartwatch $\lambda_2 : B12 \rightarrow B13$, create a link3 between FogNode and DoctorPC $\lambda_3 : B13 \rightarrow B2$, create a link3 between FogNode and NursePDA $\lambda_4 : B2 \rightarrow B21$, create a node Ambulance1 $\lambda_5 : B21 \rightarrow B3$, create a link2 between Ambulance1 and FogNode $\lambda_6 : B3 \rightarrow B4$, create a link3 between SmarteGate and FogNode
Rearrangement Rules β_k	$\beta_1 : New(Ag_{Act}, Ag_{PC}, (Receiving, noending), DoctorPC)$ $\beta_2 : New(Ag_{Act}, Ag_{PDA}, (noreceiving, Normal, working), NursePDA)$ $\beta_3 : Mig(Ag_{Amb}\&(inactive, Sending), Ambulance2, Ambulance1)$ $\beta_4 : [Ag_{Safety}, Ag_{Security}]_x$

Figure 7.7: CRR rule Defining SSM-CPS in Situation 10

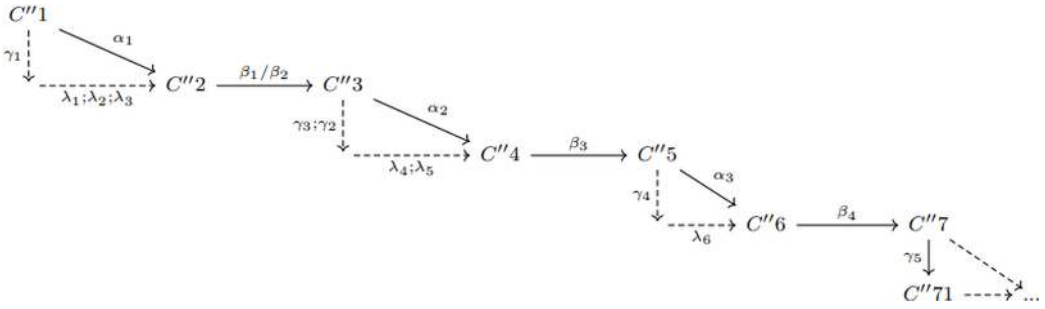


Figure 7.8: GTS Execution Trace

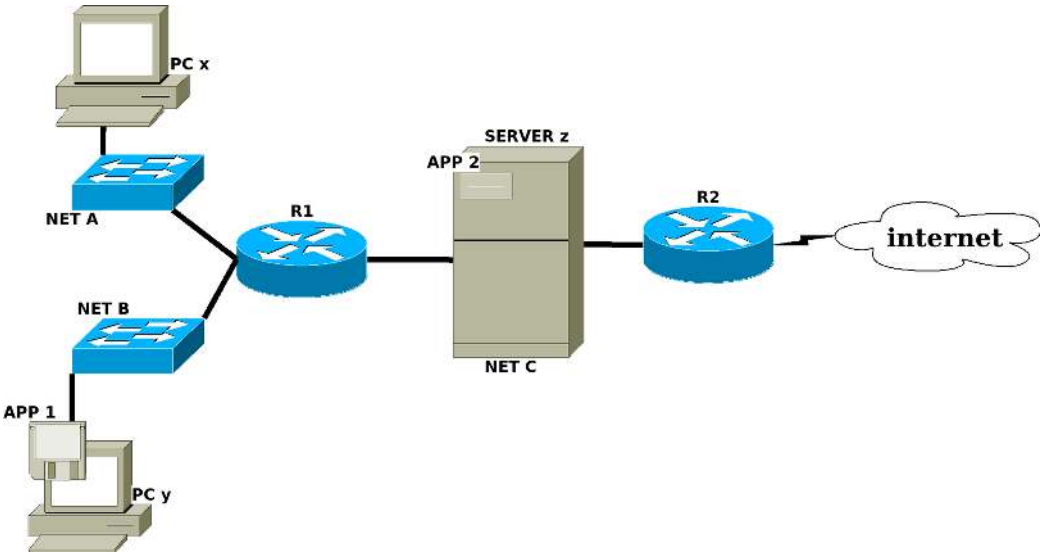


Figure 7.9: A Network Generic Architecture

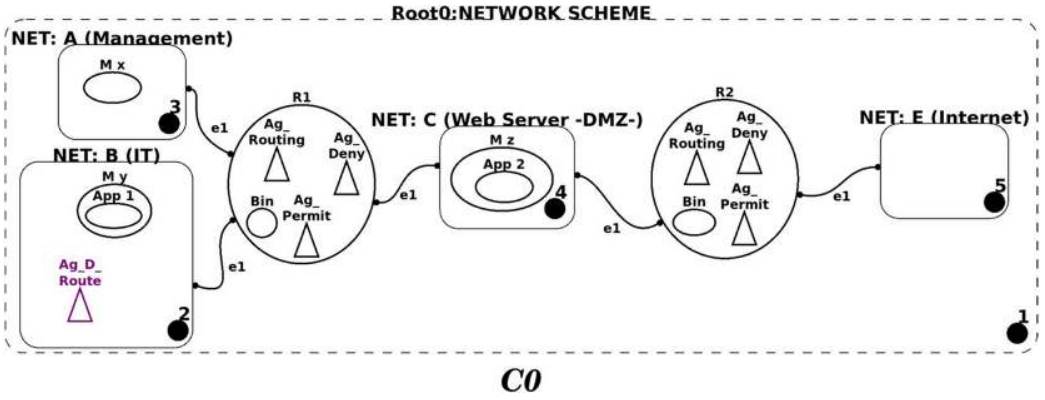


Figure 7.10: CA-BRS Example for I4.0-CPS modeling

Table 7.4: Example of CRR and GR Rules

	Routing Exception	Rule 5 (R1)
Description	This is an example showing the routing mechanism of a packet in our model, while causing the execution according R1 ACL and the (4) R2 ACL.	Discarding the Packet coming from the app 1 and going to machine x.
CRR		
Observations	• γ1: P1(1, E, R1) → P1(1, E, R2) • γ2: P1(1, E, R2) → P1(1, E, Delivered)	
Actions	• λ1: create e2 between app 1 and R1 • λ2: destroy e2 between app 1 and R1 • λ3: create e2 between R1 and C • λ4: destroy e2 between R1 and C • λ5: create e2 between C and R2 • λ6: destroy e2 between C and R2 • λ7: create e2 between R2 and E • λ8: destroy e2 between R2 and E	• λ1': create e2 between app 1 and R1 • λ2': destroy e2 between app 1 and R1
Rearrangement Rules	• β1: create(P1, 1, R=R1)[Ag_D_R_b] • β2: mig(P1, 1, R1) [Ag_Permit_R1] • β3: mig(P1, R1, C) [Ag_Routing_R1] • β4: mig(P1, C, R2) [Ag_Permit_R2] • β5: mig(P1, R2, E) [Ag_Routing_R2]	• β1': create(P3, 1, R=R1)[Ag_D_R_b] • β2': mig(P3, 1, Bin) [Ag_Deny_R1]

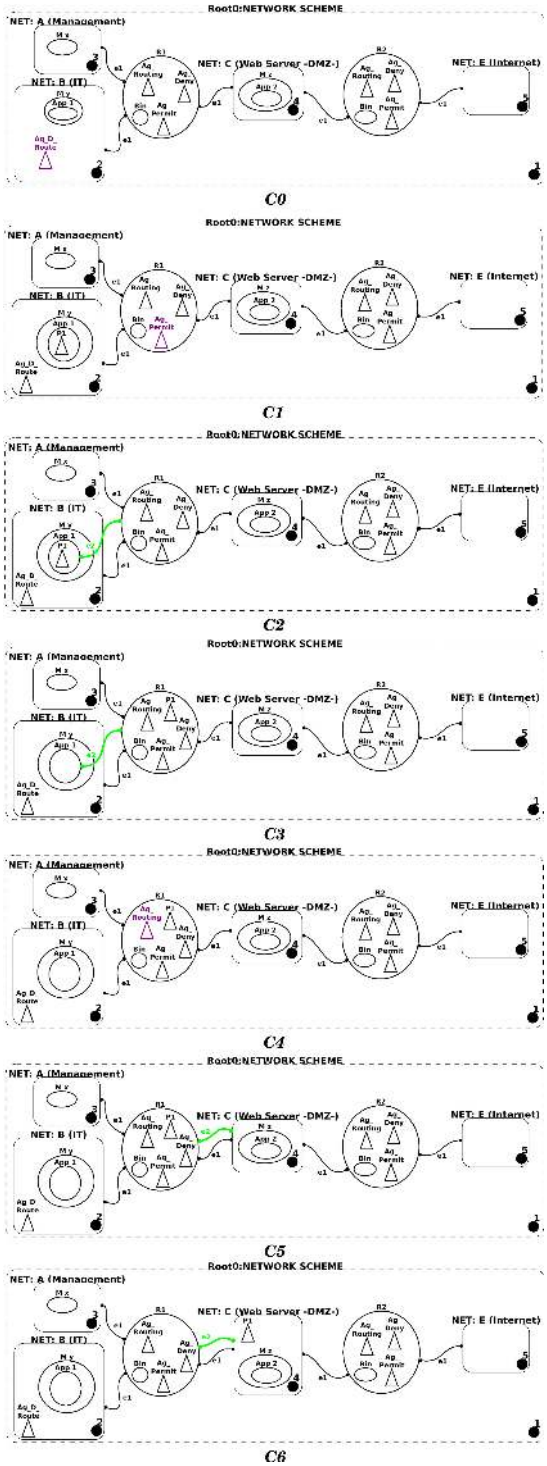


Figure 7.11: CA-BRS States Evolution: Rule 1

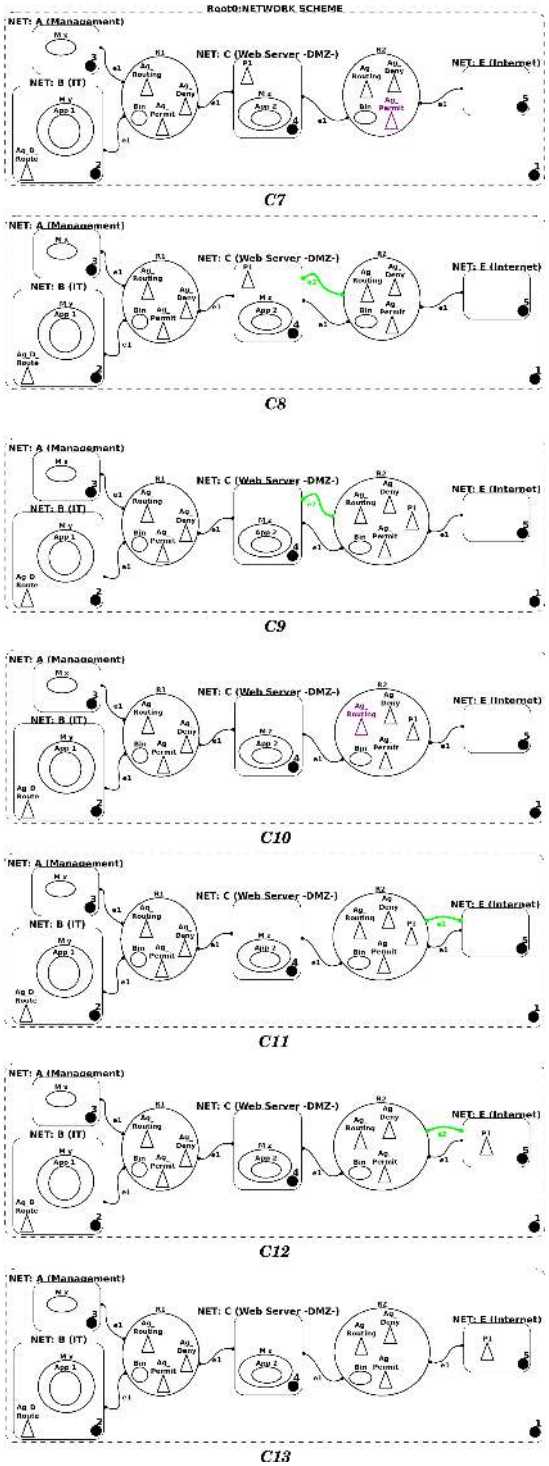


Figure 7.12: CA-BRS States Evolution: Rule 2

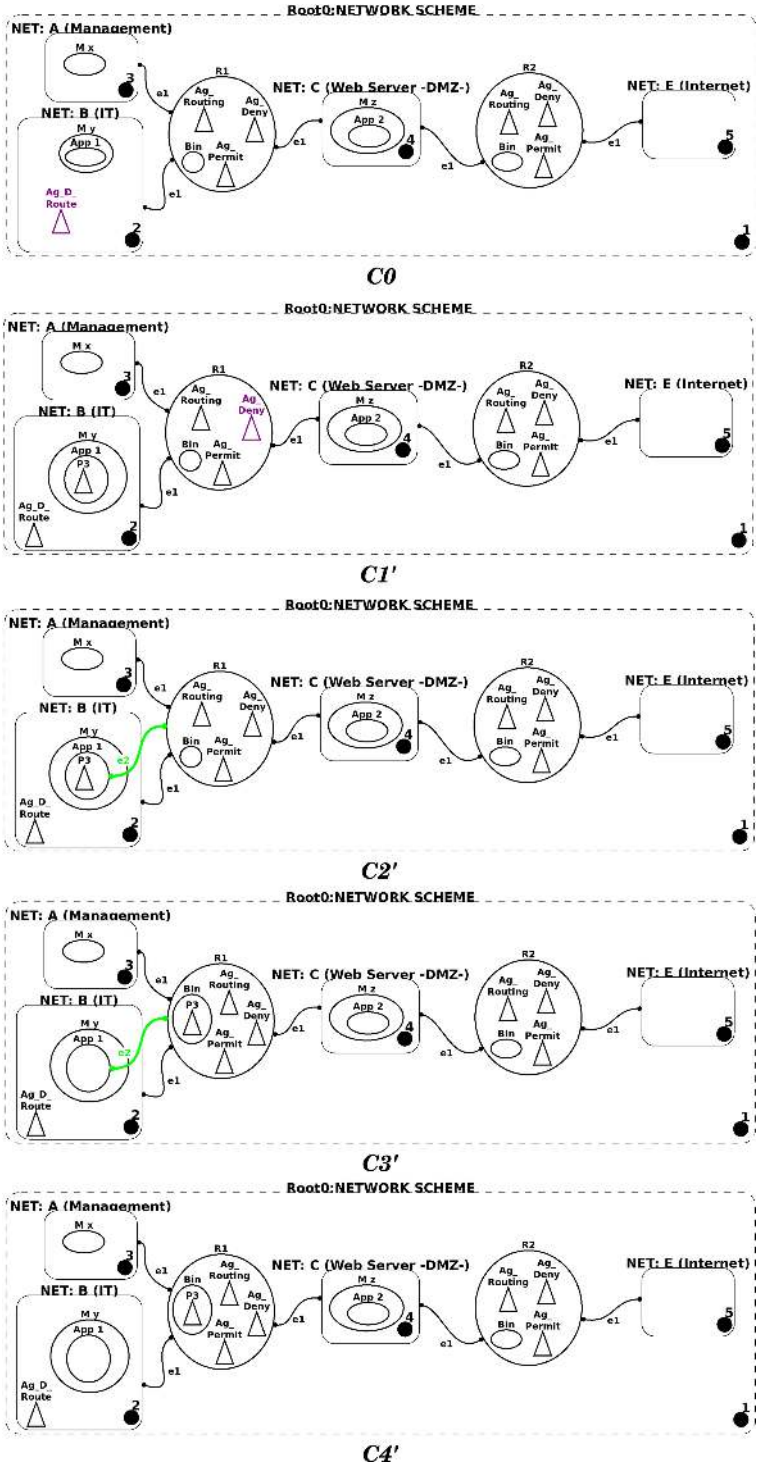


Figure 7.13: CA-BRS States Evolution: Rule 1

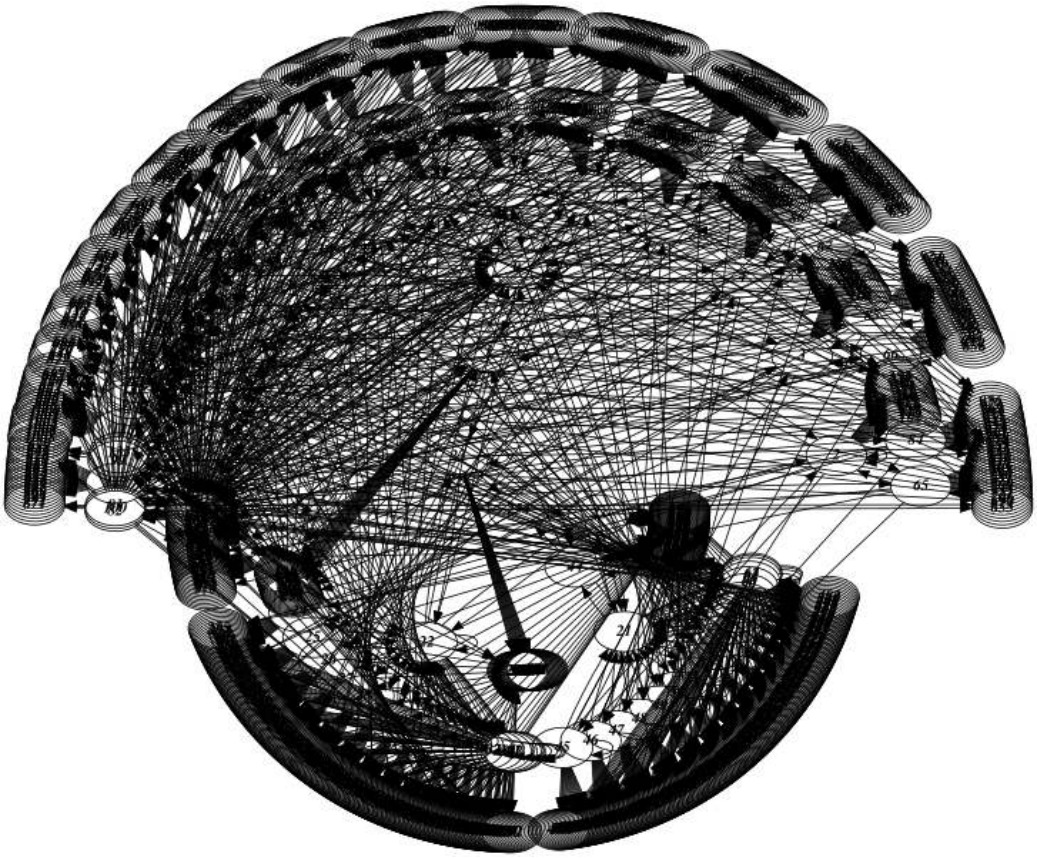


Figure 7.14: A possible Execution of CA-BRS model Example on BigraphER

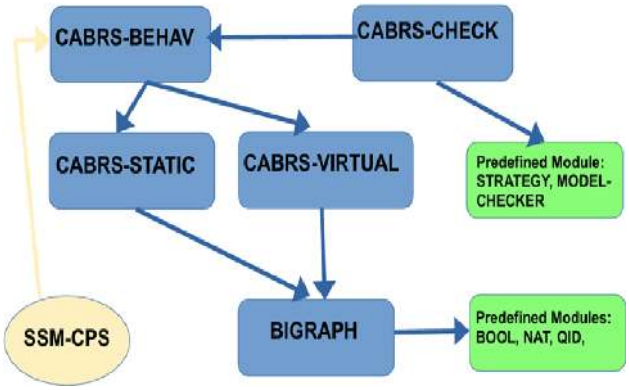


Figure 7.15: Maude Modules for CA-BRS Encoding

```
<CA-BRS> ::= ( <bigraph> + <agents>)
<bigraph> ::= ( <regions> ; <edges>)

<regions> ::= (<region> | |<regions>)
<regions> ::= nil
<region> ::= (<regionid> [<nodes>])
<regionid>::= ident

<edges> ::= (<edge> <edges> )
<edges> ::= nil
<edge> ::= ident

<nodes> ::= (<node> | <nodes>)
<node> ::= (<id>:<crtl>E <edges>)
<node> ::= (<id>:<crtl> , <nodes>)
<node> ::= (S:<sitenumb>)
<nodeid> ::= {<id>:<crtl>}:<nodeid>
<nodeid> ::= ident

<sitenumb> ::= integer
<crtl> ::= ident

<agents> ::= (<agent>;<agents>)
<agents> ::= nil
<agent> ::= ident
```

Figure 7.16: Rules In BNF Example

```
fmod BIGRAPH is
protecting NAT QID.

sorts Bigraph .
sorts Region Regions Node Nodes Site Edge Edges Ctrl .
sorts NodeId RegionName NodeName IndexSite .

subsorts NodeId IndexSite RegionName Ctrl Edge < Qid .
subsort Site < Node .
subsort Region < Regions .
subsort Node < Nodes.
subsort Edge < Edges .

op nilNS : → Nodes[ctor] .
op nilE : → Edges[ctor] .
op nilRg : → Regions[ctor] .

op _ ‘ | _: Nodes Nodes → Nodes [ctor assoc comm id: nilNS ] .
op _ _ : Edges Edges → Edges [ctor assoc comm id: nilE ] .
op _’| | _ : Regions Regions → Regions [ctor assoc id: nilRg ] .

op _ _ : IndexSite Nat → Site[ctor] .
op _‘: _ : NodeId Ctrl → NodeName [ctor] .
op _E_ : NodeName Edges → Node[ctor]
op _‘._ : NodeName Nodes → Node[ctor] .
op _‘[ _ ‘] : RegionName Nodes → Region [ctor] .

op _‘;_ : Regions Edges → Bigraph [ctor] .
op parent: Node Nodes → NodeId

Endfm

fmod CABRS-STATIC is
including BIGRAPH.

vars N: NodeId K: Ctrl Es: Edges Ns1, Ns2: Nodes.
eq parent(N, Ns1 . N:K E Es | Ns2 ) = Ns1 .
eq parent(N, N:K E Es | Ns2 ) = NilNS .
eq parent(N, Ns1 . N:K | Ns2) = Ns1 .
eq parent(N, N:K | Ns2) = NilNS .

Endfm
```

Figure 7.17: Maude Modules for Encoding CA-BRS Static Part

```
fmod CABRS-VIRTUAL is
including BIGRAPH.

sorts Cabrs Cagent Cagents .
subsort   Cagent < Qid .
subsort Cagent < Cagents .

op nilA : → Cagents[ctor] .

op _‘,_ : Cagent Cagents → Cagents [ctor] .
op _‘+_ : Bigraph Cagents → Cabrs [ctor] .

op Virt : Cabrs → Cagents .
op Phys : Cabrs → Bigraph .

vars B: Bigraph   CA: Agents .
eq Virt(B + CA) = CA .
eq Phys(B + CA) = B .
Endfm
```

Figure 7.18: Maude Module for Encoding CA-BRS Virtual Part

```
mod SSM-CPS-STATIC is
including  CABRS-STATIC .

ops  Home Healthcenter Cloud   : → NameRegion  [ctor] .

ops  Smartlimbsupport   Pacemaker   Responder Ambulance Ambulance1 Ambulance2 Smartwatch Smartgate Medicinedrone1
Medicinedrone2 Internalunit Minitoringcamera Livesupportmachine Fognode NursepDA Doctorpc Archieving  Businessinteligence  Datamining
  : → NodeId  [ctor] .

ops  Agsl Agsg Agdro Agmc Agamb Aglm Agfn Agpda Agpc Agarch  : → Cagent  [ctor] .
ops  Act Sen As Net Iu Ser R A : → Ctrl  [ctor] .
ops  Link1 Link2 Link3 : → Edge  [ctor] .
op   D :  → IndexSite  [ctor] .
op Ssmcps-init :  →Cabrs  [ctor] .

eq  Ssmcps-init  =
Home[Smartlimbsupport E Link2 ⊠Pacemaker E Link3 ⊠ Smartwatch E Link3 ⊠Smartgate E Link3 ⊠ D2 ⊠nilNS]
⊠ ⊠
Healthcenter[ Responder .  (Medicinedrone1 E  Link2 ⊠ Medicinedrone2 E  Link2 ⊠D4) ⊠ Ambulance . (Ambulance1 E  Link2 ⊠Ambulance2
E  Link2 ⊠ D5)  ⊠ Internalunit . (Monitoringcamera E  Link3 ⊠Livesupportmachine E  Link3 ⊠ D3) ⊠ Fognode E  Link1 Link2 Link3
⊠Nursepda E  Link3 ⊸ DoctorPC E  Link3 ⊠ D1
⊠ ⊠
Cloud[Archieving E  Link1 ⊠ Businessinteligence E  Link1 ⊠ Businessinteligence E  Link1 ⊠S0 ⊠nilNS]
⊠ ⊠
NilRg
; (Link1 Link2 Link3 NilE) + Agsl,Agsg,Agdro,Agmc,Agamb,Aglm,Agfn,Agpda,Agpc,Agarch,nilA) .

endm
```

Figure 7.19: Maude Module specifying SSM-CPS static and virtual parts

Chapter 8

Conclusion

Cyber Physical Systems (CPS) couple the cyber aspects of computing and communications tightly with the dynamics and physics of physical systems operating in the world around us. This emerging multidisciplinary frontier will enable revolutionary changes in the way humans live. CPS will transform the way humans interact and control the physical environment for the greater benefit of society. The major economic sectors that will see dramatic advances include transportation, energy, buildings, healthcare, manufacturing, physical infrastructure, agriculture, and defense, among others. On the other hand, the CPS community will comprise computer scientists, engineers of all stripes, as well as many biologists, chemists, and physicians.

The design and implementation of complex systems pose significant conceptual and technical challenges for the computer science community, particularly in software engineering. As these systems become increasingly intricate, it becomes more difficult, if not impossible, for an individual to comprehend them in their entirety. Overcoming these challenges requires progress in modeling languages, formal methods, and tools specifically designed for CPS. In addition, it requires interdisciplinary collaboration among domain experts, formal methods researchers, and tool developers to create effective techniques for modeling and analyzing CPS.

This doctoral thesis has focused on the advancement of the specification and formal analysis of CPS, with a particular emphasis on their application in the field of medical systems. CPS have become a cornerstone of modern technology, addressing complex, safety-critical domains. Among these, medical systems (M-CPS) stand out as they directly impact human lives, necessitating robust methods for their development and validation. We have

proposed a new methodology for designing, defining and reasoning about the dynamic and secure behavior of CPS, taking into account the existing gap that affects the practical use of formal methods in the development of large and heterogeneous systems. The main contributions of this work can be summarized as follows.

- In our research, we have investigated the state-of-the-art in the specification and formal analysis of CPS, highlighting the need for precise, systematic approaches. After this finding, the next natural step has been reviewing what has been done in terms of collaboration between CPS and Security. We also wanted to understand the state of the art in the field of medical CPS to handle physical and cyber security, for this reason, we have reviewed some generic approaches and have reached the conclusion that the application of security-by-design principles during the development of CPS is essential to have an acceptable level of security assurance; Safety and Security modeling should be considered at the early stages of the development life-cycle.

- In this context, we have proposed a meta-model grounded in the IEEE 42010 standard, which provides a structured framework for specifying secure CPS. This meta-model not only ensures clarity and consistency but also facilitates the alignment of system requirements with design and implementation. Thus, a set of CPS execution scenarios has been collectively defined to analyze and simulate the system's behavior.

- We have assigned rigorous semantics to each element of the CPS meta model using bigraphs to enable formal analysis. These methods ensure the correctness, reliability, and safety of CPS, which is especially critical in medical applications where even minor errors can have severe consequences. Since no single formalism can model all aspects of CPS, we proposed a new formalism, CA-BRS, which combines Bigraphical Reactive Systems (BRS) and Control Agents. This extended BRS formalism achieved a clear separation between various levels of the CPS: the virtual level, described using abstract agents; the physical and cyber levels, encompassing execution machines, their environments, and software applications, all specified through bigraphs; and the behavioral level, which captures the CPS dynamics. The behavioral level is represented using an enhanced form of reaction rules (Controlled Reaction Rules) that support two types of temporal evolution: physical

evolution, modeled as a sequence of bigraphs, and virtual evolution, driven by agent movement and migration (cyber mobility). This approach also considered material constraints through agent observations, represented using rewriting rules.

- In the context of CPS behavior analysis, the high-level reasoning enabled by CA-BRS allowed us to propose two distinct perspectives for defining Control Agents (CA) and their possible implementations. The States perspective focuses on modeling the behavior of reactive agents by describing their possible states and the transitions between them in response to specific events. On the other hand, the Activities perspective is used to represent the processes of non-reactive agents, defining workflows that include decisions, loops, and concurrent activities. Each perspective employs specialized notations to represent the different states and transitions of CAs, providing valuable tools for understanding and communicating their behavior to stakeholders. To demonstrate their applicability, we implemented the Activities perspective using the BPMN notation, an industry-standard developed by the OMG consortium, which is accessible to both technical and non-technical stakeholders. Similarly, we implemented the States perspective using Maude, a high-performance language and system that supports equational and rewriting logic computation for a wide range of applications.

- Finally, by leveraging these approaches, we demonstrated the feasibility of verifying critical properties such as safety and security, thereby enhancing the dependability of medical CPS. To validate this, we conducted a realistic case study on Medical-CPS, focusing on network routing through access control lists and ensuring data confidentiality in electronic health records. This case study underscored the need to balance theoretical and practical considerations when addressing the physical, cyber, and security dimensions of CPS.

Although this thesis has made significant strides, several avenues remain for future research. We can outline the following directions without being exhaustive.

- On the theoretical side, we aim to extend the CA-BRS formalism to incorporate the time aspect in CPS, enabling the definition of new agent types to address real-time constraints. Additionally, the developed

metamodel could benefit from the Model-Driven Architecture (MDA) approach, using model transformation techniques to automatically generate CA-BRS-based specifications for any CPS. Other promising directions include expanding the applicability of our approaches to diverse domains, addressing scalability challenges in formal methods, and integrating advanced machine learning techniques to support adaptive CPS.

- On the practical side, we intend to develop a Tool chain, which leverages the CA-BRS model, involving the following roles collaborating closely, to ensure that the specified CPS meets its security requirements and behaves as expected.
  - **Formal Modeler:** uses formal languages and tools to specify CPS behavior, and properties.
  - **Software Engineer/Developer:** responsible for implementing the CPS based on its formal specification provided by the formal modeler. He translates the CA-BRS model into executable ones (based on Maude and BPMN). The software engineer will have the ability to integrate formal verification tools into the development process to ensure accuracy and reliability.
  - **Verification Engineer:** will use formal verification techniques such as model checking (of Maude) and simulation (based on the BPMN model) to identify potential errors or inconsistencies in the CPS design.

Through this work, we have contributed to bridge the gap between conceptual modeling and formal analysis in the CPS domain. The methodologies and programs (Maude and BPMN) developed provide not only theoretical insights, but also practical solutions for designing reliable and safe systems.

Bibliography

[1] Ole Høgh Jensen and Robin Milner. Bigraphs and mobile processes (revised). Technical report, University of Cambridge, Computer Laboratory, 2004.

[2] Imen Graja, Slim Kallel, Nawal Guermouche, and Ahmed Hadj Kacem. Bpmn4cps: A bpmn extension for modeling cyber-physical systems. In *2016 IEEE 25th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE)*, pages 152–157. IEEE, 2016.

[3] Kazeem B Adedeji and Yskandar Hamam. Cyber-physical systems for water supply network management: Basics, challenges, and roadmap. *Sustainability*, 12(22):9555, 2020.

[4] Zhenhua Yu, Hongxia Gao, Xuya Cong, Naiqi Wu, and Houbing Herbert Song. A survey on cyber-physical systems security. *IEEE Internet of Things Journal*, 2023.

[5] A conceptual model of architecture description. Last accessed 6 November 2023.

[6] Iso/iec/ieee systems and software engineering – architecture description. *ISO/IEC/IEEE 42010:2011(E) (Revision of ISO/IEC 42010:2007 and IEEE Std 1471-2000)*, pages 1–46, 2011.

[7] Udo Kannengiesser and Harald Müller. Multi-level, viewpoint-oriented engineering of cyber-physical production systems: An approach based on industry 4.0, system architecture and semantic web standards. In *2018 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 331–334, 2018.

[8] Lamia Ben Romdhane. *A Multi-Agent Architecture Framework for Energy Systems Design*. PhD thesis, Université Paris-Saclay, 2020.

[9] Uwe Van Heesch, Paris Avgeriou, and Rich Hilliard. A documentation framework for architecture decisions. *Journal of Systems and Software*, 85(4):795–820, 2012.

[10] Rabeh Morrar, Husam Arman, and Saeed Mousa. The fourth industrial revolution (industry 4.0): A social innovation perspective. *Technology innovation management review*, 7(11):12–20, 2017.

[11] Xifan Yao and Yingzi Lin. Emerging manufacturing paradigm shifts for the incoming industrial revolution. *The International Journal of Advanced Manufacturing Technology*, 85:1665–1676, 2016.

[12] Jingwei Huang. Digital engineering transformation with trustworthy ai towards industry 4.0: emerging paradigm shifts. *Journal of Integrated Design and Process Science*, 26(3-4):267–290, 2022.

[13] Isha Batra, Chetan Sharma, Arun Malik, Shamneesh Sharma, Mahender Singh Kaswan, and Jose Arturo Garza-Reyes. Industrial revolution and smart farming: a critical analysis of research components in industry 4.0. *The TQM Journal*, 2024.

[14] Michael Rüßmann, Markus Lorenz, Philipp Gerbert, Manuela Waldner, Jan Justus, Pascal Engel, and Michael Harnisch. Industry 4.0: The future of productivity and growth in manufacturing industries. *Boston consulting group*, 9(1):54–89, 2015.

[15] Mamad Mohamed. Challenges and benefits of industry 4.0: An overview. *International Journal of Supply and Operations Management*, 5(3):256–265, 2018.

[16] Baudouin Dafflon, Nejib Moalla, and Yacine Ouzrout. The challenges, approaches, and used techniques of cps for manufacturing in industry 4.0: a literature review. *The International Journal of Advanced Manufacturing Technology*, 113:2395–2412, 2021.

[17] Bingjie Ding, Xavier Ferras Hernandez, and Nuria Agell Jane. Combining lean and agile manufacturing competitive advantages through

industry 4.0 technologies: an integrative approach. *Production planning & control*, 34(5):442–458, 2023.

[18] Siddhartha Kumar Khaitan and James D McCalley. Design techniques and applications of cyberphysical systems: A survey. *IEEE systems journal*, 9(2):350–365, 2014.

[19] Hee-Woon Cheong and Hwally Lee. Technology and policy strategies in the era of cps (cyber physical system) and automated driving. *Procedia computer science*, 122:102–105, 2017.

[20] Tamara Denning, Daniel B Kramer, Batya Friedman, Matthew R Reynolds, Brian Gill, and Tadayoshi Kohno. Cps: Beyond usability: Applying value sensitive design based methods to investigate domain characteristics for security for implantable cardiac devices. In *Proceedings of the 30th Annual Computer Security Applications Conference*, pages 426–435, 2014.

[21] Liviu Miclea and Teodora Sanislav. About dependability in cyber-physical systems. In *2011 9th East-West Design & Test Symposium (EWDTS)*, pages 17–21. IEEE, 2011.

[22] Shaymaa Mamdouh Khalil, Hayretdin Bahsi, Henry Ochieng'Dola, Tarmo Korõtko, Kieran McLaughlin, and Vahur Kotkas. Threat modeling of cyber-physical systems-a case study of a microgrid system. *Computers & Security*, 124:102950, 2023.

[23] Robin Milner. Bigraphs and their algebra. *Electronic Notes in Theoretical Computer Science*, 209:5–19, 2008.

[24] E Pereira, C Kirsch, and R Sengupta. Biagents a bigraphical agent model for structure-aware computation. *Cyber-Physical Cloud Computing Working Papers, CPCC Berkeley*, pages 1–13, 2012.

[25] Eloi Pereira, Christoph M Kirsch, Raja Sengupta, and João Borges de Sousa. Bigactors a model for structure-aware computation. In *2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*, pages 199–208. IEEE, 2013.

[26] Gul Agha, Ian A Mason, Scott Smith, and Carolyn Talcott. Towards a theory of actor computation. In *CONCUR'92: Third International*

Conference on Concurrency Theory Stony Brook, NY, USA, August 24–27, 1992 Proceedings 3, pages 565–579. Springer, 1992.

[27] Jean Krivine, Robin Milner, and Angelo Troina. Stochastic bigraphs. *Electronic Notes in Theoretical Computer Science*, 218:73–96, 2008.

[28] Michele Sevegnani and Muffy Calder. Bigraphs with sharing. *Theoretical Computer Science*, 577:43–73, 2015.

[29] Tomas Skersys, Rimantas Butleris, and Kestutis Kapocius. Extracting business vocabularies from business process models: Sbvr and bpmn standards-based approach. In *AIP Conference Proceedings*, volume 1558, pages 341–344. American Institute of Physics, 2013.

[30] Paolo Bocciarelli, Andrea D'Ambrogio, Andrea Giglio, and Emiliano Paglia. A bpmn extension for modeling cyber-physical-production-systems in the context of industry 4.0. In *2017 IEEE 14th international conference on networking, sensing and control (icnsc)*, pages 599–604. IEEE, 2017.

[31] Alberto Verdejo and Narciso Martí-Oliet. Executable structural operational semantics in maude. *The Journal of Logic and Algebraic Programming*, 67(1-2):226–293, 2006.

[32] USA SRI International, CA. The maude system, Year of publication. (Accessed on 2024-05-23).

[33] Narciso Martí-Oliet and José Meseguer. Rewriting logic as a logical and semantic framework. *Electronic Notes in Theoretical Computer Science*, 4:190–225, 1996.

[34] Steven Eker, Narciso Martí-Oliet, José Meseguer, Rubén Rubio, and Alberto Verdejo. The maude strategy language. *Journal of Logical and Algebraic Methods in Programming*, 134:100887, 2023.

[35] Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martı-Oliet, José Meseguer, and José F Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 285(2):187–243, 2002.

[36] National Science Foundation. Nsf 10-515: Cyber-physical systems (cps), 2009. Accessed: April 2025.

[37] Shafiq Ur Rehman and Volker Gruhn. An effective security requirements engineering framework for cyber-physical systems. *Technologies*, 6(3):65, 2018.

[38] J Sukarno Mertoguno, Ryan M Craven, Matthew S Mickelson, and David P Koller. A physics-based strategy for cyber resilience of cps. In *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2019*, volume 11009, pages 79–90. SPIE, 2019.

[39] Teodora Sanislav and Liviu Miclea. Cyber-physical systems-concept, challenges and research areas. *Journal of Control Engineering and Applied Informatics*, 14(2):28–33, 2012.

[40] Sandeep Kumar, Vikas Manjrekar, Vivek Singh, and Bhupesh Kumar Lad. Integrated yet distributed operations planning approach: A next generation manufacturing planning system. *Journal of Manufacturing Systems*, 54:103–122, 2020.

[41] Tian Shubin and Pan Zhi. "made in china 2025" and "industrie 4.0"—in motion together. In *The Internet of Things: Industrie 4.0 Unleashed*, pages 87–113. Springer, 2017.

[42] Liu Jianjiang, Zhang Zhiyue, and Fidelis Ayangbah and. The impact of industrial internet on high quality development of the manufacturing industry. *Cogent Economics & Finance*, 10(1):2135723, 2022.

[43] The Special Competitive Studies Project (SCSP). National action plan for u.s. leadership in advanced manufacturing, 2024. Accessed: April 2025.

[44] Johannes W Veile, Daniel Kiel, Julian Marius Müller, and Kai-Ingo Voigt. Lessons learned from industry 4.0 implementation in the german manufacturing industry. *Journal of manufacturing technology management*, 31(5):977–997, 2020.

[45] Vojislav B Misic and Jelena Misic. *Machine-to-machine communications: architectures, technology, standards, and applications*. CRC Press, 2014.

[46] Strategic Vision. Business drivers for 21st century cyber-physical systems. *Report from the Executive Roundtable on Cyber-Physical Systems*, 2013.

[47] ISO. Cyber-physical systems, 2023. Accessed: April 2025.

[48] Marty Trevino. Cyber physical systems. *Prism*, 8(3):2–13, 2019.

[49] Lukas Esterle and Radu Grosu. Cyber-physical systems: challenge of the 21st century. *Elektrotechnik und Informationstechnik: e & i*, 133(7):299–303, 2016.

[50] Amir Abiri Jahromi and Deepa Kundur. Fundamentals of cyber-physical systems. *Cyber-Physical Systems in the Built Environment*, pages 1–13, 2020.

[51] Ilias Gerostathopoulos, Tomas Bures, Petr Hnetynka, Jaroslav Keznikl, Michal Kit, Frantisek Plasil, and Noël Plouzeau. Self-adaptation in software-intensive cyber–physical systems: From system goals to architecture configurations. *Journal of Systems and Software*, 122:378–397, 2016.

[52] László Monostori, Botond Kádár, Thomas Bauernhansl, Shinsuke Kondoh, Soundar Kumara, Gunther Reinhart, Olaf Sauer, Gunther Schuh, Wilfried Sihn, and Kenichi Ueda. Cyber-physical systems in manufacturing. *Cirp Annals*, 65(2):621–641, 2016.

[53] Lipika Deka, Sakib M Khan, Mashrur Chowdhury, and Nick Ayres. Transportation cyber-physical system and its importance for future mobility. In *Transportation cyber-physical systems*, pages 1–20. Elsevier, 2018.

[54] None Oladosu Samuel. Cyber physical energy system. *technology*, 1:3, 2019.

[55] W An, D Wu, S Ci, H Luo, V Adamchuk, and Z Xu. Agriculture cyber-physical systems. In *Cyber-physical systems*, pages 399–417. Elsevier, 2017.

[56] Mischa Schmidt and Christer Åhlund. Smart buildings as cyber-physical systems: Data-driven predictive control strategies for energy

efficiency. *Renewable and Sustainable Energy Reviews*, 90:742–756, 2018.

[57] Justyna Zander, Pieter J Mosterman, Taskin Padir, Yan Wan, and Shengli Fu. Cyber-physical systems can make emergency response smart. *Procedia Engineering*, 107:312–318, 2015.

[58] Nilanjan Dey, Amira S Ashour, Fuqian Shi, Simon James Fong, and João Manuel RS Tavares. Medical cyber-physical systems: A survey. *Journal of medical systems*, 42:1–13, 2018.

[59] Muhammad Shafique, Faiq Khalid, and Semeen Rehman. Intelligent security measures for smart cyber physical systems. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 280–287. IEEE, 2018.

[60] Paul Mueller and Babak Yadegari. The stuxnet worm. *Département des sciences de linformatique, Université de lArizona. Recuperado de: https://www2. cs. arizona. edu/˜ collberg/Teaching/466-566/2012/Resources/presentations/topic9-final/report. pdf*, 2012.

[61] Defense Use Case. Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 388(1-29):3, 2016.

[62] Charlie Miller. Lessons learned from hacking a car. *IEEE Design & Test*, 36(6):7–9, 2019.

[63] Roland C Bodenheim. Impact of the shodan computer search engine on internet-facing industrial control system devices. 2014.

[64] John Fitzgerald, Carl Gamble, Peter Gorm Larsen, Kenneth Pierce, and Jim Woodcock. Cyber-physical systems design: formal foundations, methods and integrated tool chains. In *2015 IEEE/ACM 3rd FME Workshop on Formal Methods in Software Engineering*, pages 40–46. IEEE, 2015.

[65] Ezio Bartocci, Jyotirmoy Deshmukh, Alexandre Donzé, Georgios Fainekos, Oded Maler, Dejan Ničković, and Sriram Sankaranarayanan. Specification-based monitoring of cyber-physical systems: a survey on

theory, tools and applications. *Lectures on Runtime Verification: Introductory and Advanced Topics*, pages 135–175, 2018.

[66] Georgios Bakirtzis, Christina Vasilakopoulou, and Cody H Fleming. Compositional cyber-physical systems modeling. *arXiv preprint arXiv:2101.10484*, 2021.

[67] Guangquan Zhang, Mingtai Zhang, Rongjie Yan, Mingcai Chen, Chengkai Xu, and Yejing Li. Modeling and analysis for cps physical entities based on spatio-temporal petri net. *J. Comput.*, 9(2):499–505, 2014.

[68] Ying Tan, Mehmet C Vuran, and Steve Goddard. Spatio-temporal event model for cyber-physical systems. In *2009 29th IEEE International Conference on Distributed Computing Systems Workshops*, pages 44–50. IEEE, 2009.

[69] Yan Cao, Zhiqiu Huang, Changbo Ke, Jian Xie, and Jinyong Wang. A topology-aware access control model for collaborative cyber-physical spaces: Specification and verification. *Computers & security*, 87:101478, 2019.

[70] Yan Cao, Zhiqiu Huang, Shuanglong Kan, Dajuan Fan, and Yang Yang. Specification and verification of a topology-aware access control model for cyber-physical space. *Tsinghua science and technology*, 24(5):497–519, 2019.

[71] Ayoub Bouheroum, Djamel Benmerzoug, Sofiane Mounine Hemam, Faiza Belala, Aya Lehamdi, and Radhia Aouissate. A formal integrated approach for cyber physical systems. In *2022 4th International Conference on Pattern Analysis and Intelligent Systems (PAIS)*, pages 1–7. IEEE, 2022.

[72] Ayoub Bouheroum, Djamel Benmerzoug, Sofiane Mounine Hemam, and Faiza Belala. From ca-brs to bpmn: Formal approach for modeling adaptive security in cyber-physical systems. In *TACC*, pages 149–163, 2021.

[73] Hyunguk Yoo and Taeshik Shon. Challenges and research directions for heterogeneous cyber–physical system based on iec 61850: Vulnerabili-

ties, security requirements, and security architecture. *Future generation computer systems*, 61:128–136, 2016.

[74] Rasim Alguliyev, Yadigar Imamverdiyev, and Lyudmila Sukhostat. Cyber-physical systems and their security issues. *Computers in Industry*, 100:212–223, 2018.

[75] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo. Cyber-physical systems security—a survey. *IEEE Internet of Things Journal*, 4(6):1802–1831, 2017.

[76] Tianbo Lu, Jiaxi Lin, Lingling Zhao, Yang Li, and Yong Peng. An analysis of cyber physical system security theories. In *2014 7th international conference on security technology*, pages 19–21. IEEE, 2014.

[77] Seyed Mehran Dibaji, Mohammad Pirani, David Bezalel Flamholz, Anuradha M Annaswamy, Karl Henrik Johansson, and Aranya Chakrabortty. A systems and control perspective of cps security. *Annual reviews in control*, 47:394–411, 2019.

[78] Elias Bou-Harb. A brief survey of security approaches for cyber-physical systems. In *2016 8th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, pages 1–5. IEEE, 2016.

[79] Phu H Nguyen, Shaukat Ali, and Tao Yue. Model-based security engineering for cyber-physical systems: A systematic mapping study. *Information and Software Technology*, 83:116–135, 2017.

[80] Amin Mahnamfar and Nafiz Ünlü. Cyber-physical systems and their security issues. *Savunma Bilimleri Dergisi*, 1(41):97–118, 2022.

[81] Rafiullah Khan, Kieran McLaughlin, David Laverty, and Sakir Sezer. Stride-based threat modeling for cyber-physical systems. In *2017 IEEE PES Innovative Smart Grid Technologies Conference Europe (ISGT-Europe)*, pages 1–6. IEEE, 2017.

[82] Monika Maidl, Roman Wirtz, Tiange Zhao, Maritta Heisel, and Marvin Wagner. Pattern-based modeling of cyber-physical systems for analyzing security. In *Proceedings of the 24th European Conference on Pattern Languages of Programs*, pages 1–10, 2019.

[83] Paul Wach and Alejandro Salado. Model-based security requirements for cyber-physical systems in sysml. In *2020 IEEE Systems Security Symposium (SSS)*, pages 1–7. IEEE, 2020.

[84] Tomas Bures, Danny Weyns, Christian Berger, Stefan Biffl, Marian Daun, Thomas Gabor, David Garlan, Ilias Gerostathopoulos, Christine Julien, Filip Krikava, et al. Software engineering for smart cyber-physical systems–towards a research agenda: report on the first international workshop on software engineering for smart cps. *ACM SIGSOFT Software Engineering Notes*, 40(6):28–32, 2015.

[85] David Emery and Rich Hilliard. Updating ieee 1471: architecture frameworks and other topics. In *seventh working IEEE/IFIP conference on Software Architecture (WICSA 2008)*, pages 303–306. IEEE, 2008.

[86] Iso/iec 42010:2007 website. Last accessed May 2025.

[87] Rich Hilliard. Viewpoint modeling. In *Proceedings of 1st ICSE Workshop on Describing Software Architecture with UML*, volume 7. Citeseer, 2001.

[88] John A Zachman. A framework for information systems architecture. *IBM systems journal*, 26(3):276–292, 1987.

[89] Philippe B Kruchten. The 4+ 1 view model of architecture. *IEEE software*, 12(6):42–50, 1995.

[90] IEEE. Ieee recommended practice for architectural description for software-intensive systems, 2000. IEEE 1471-2000, Last accessed 6 November 2023.

[91] Iso/iec/ieee 42010 website. Last accessed 6 November 2023.

[92] IEEE. Institute of electrical and electronics engineers, 1963. Last accessed 6 November 2023.

[93] Rich Hilliard.

[94] Rich Hilliard. Architecture viewpoint template for iso/iec/ieee 42010.

[95] ISO. Iso/iec/ieee 42010:2011, 2018. Last accessed 6 November 2023.

[96] Rich Hilliard. Architecture description template for use with iso/iec/ieee 42010:2011.

[97] Martin Hankel and Bosch Rexroth. The reference architectural model industrie 4.0 (rami 4.0). *Zvei*, 2(2):4–9, 2015.

[98] Ayoub Bouheroum, Abdelouahid Derhab, Djamel Benmerzoug, Sofiane Mounine Hemam, and Abdelghani Bouras. A brs-based modeling approach for secure medical cyber-physical systems. *Authorea Preprints*, 2023.

[99] M Clave, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Marti-Oliet, José Meseguer, and José F Quesada. Towards maude 2.0. *Electronic Notes in Theoretical Computer Science*, 36:294–315, 2000.

[100] Rubén Rubio. An overview of the maude strategy language and its applications. In *International Workshop on Rewriting Logic and its Applications*, pages 65–84. Springer, 2022.

[101] Rubén Rubio, Narciso Martí-Oliet, Isabel Pita, and Alberto Verdejo. Strategies, model checking and branching-time properties in maude. *Journal of Logical and Algebraic Methods in Programming*, 123:100700, 2021.

[102] Siwar Kriaa, Ludovic Pietre-Cambacedes, Marc Bouissou, and Yoran Halgand. A survey of approaches combining safety and security for industrial control systems. *Reliability engineering & system safety*, 139:156–178, 2015.

[103] Mike Chapple. Confidentiality, integrity and availability–the cia triad, 2020.

[104] Joseph Merhej, Hassan Harb, Abdelhafid Abouaissa, Lhassane Idoumghar, and Samir Ouchani. Elso: a blockchain-based technique for a reliable and secure healthcare information exchange. *Arabian Journal for Science and Engineering*, pages 1–23, 2023.

[105] Iso/tr 31004, 2013.

[106] Brandon Carroll. *Cisco access control security: AAA administrative services*. Cisco Press, 2004.

[107] A Melnikov. Rfc 4314: Imap4 access control list (acl) extension, 2005.

[108] Espen Højsgaard and Arne J Glenstrup. The bpl tool: A tool for experimenting with bigraphical reactive systems. *Bigraphical Languages and their Simulation*, page 85, 2011.

[109] Alexander Faithfull, Gian Perrone, and Thomas T Hildebrandt. Big red: A development environment for bigraphs. *Selected Revised Papers from the 4th International Workshop on Graph Computation Models (GCM 2012)*, 61, 2012.

[110] Giorgio Bacci, Davide Grohmann, and Marino Miculan. Dbtk: a toolkit for directed bigraphs. In *Algebra and Coalgebra in Computer Science: Third International Conference, CALCO 2009, Udine, Italy, September 7-10, 2009. Proceedings 3*, pages 413–422. Springer, 2009.

[111] Gian Perrone, Søren Debois, and Thomas T Hildebrandt. A model checker for bigraphs. In *Proceedings of the 27th annual ACM symposium on applied computing*, pages 1320–1325, 2012.

[112] Michele Sevegnani and Muffy Calder. Bigrapher: Rewriting and analysis engine for bigraphs. In *Computer Aided Verification: 28th International Conference, CAV 2016, Toronto, ON, Canada, July 17-23, 2016, Proceedings, Part II 28*, pages 494–501. Springer, 2016.