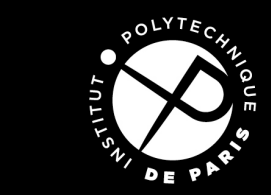
A safety approach for CPS-IoT

Mohamed Emine Laarouchi



INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2020IPPAS010

Thèse de doctorat

A safety approach for CPS-IoT

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom Sud-Paris

École doctorale n°626 de l'Institut Polytechnique de Paris (ED IP Paris)
Spécialité de doctorat : Réseaux, information et communications

Thèse présentée et soutenue à Paris, le 10/11/2020, par

MOHAMED EMINE LAAROUCHI

Composition du Jury :

Guy Pujolle Professeur, Université Sorbonne	Président
Luigi Liquori HDR, INRIA	Rapporteur
Aomar Osmani HDR, LIP Paris 13	Rapporteur
Michael Mendler Professeur, Université de Bamberg	Rapporteur
Amel Mammar Professeur, Télécom Sud-Paris	Examinatrice
Andreia Cathelin HDR, ST Microelectronics	Examinatrice
Hakima Chaouchi HDR, Télécom Sud-Paris	Directrice de thèse
Daniela Cancila PhD, CEA	Co-directrice de thèse
Sebti Mouelhi Maitre de conférences, ESTACA	Invité

ECOLE
DOCTORALE

Titre : Une démarche de sûreté pour les CPS-IoT

Mots clés : Systèmes cyber-physiques, Internet des Objets, sûreté de fonctionnement, sécurité

Résumé : Depuis plusieurs années, nous assistons à une convergence entre les systèmes cyber-physiques (CPS) et l'Internet des Objets (IoT). Les CPS intègrent les systèmes embarqués avec leur environnement physique et humain en assurant une communication entre différents capteurs et actionneurs. L'IoT vise le réseau et les protocoles de communication entre les objets connectés. Cette convergence offre des perspectives d'applications diverses allant des véhicules connectés aux réseaux électriques intelligents ainsi qu'aux usines du futur.
Le but de cette thèse est d'assurer et garantir la sûreté de fonctionnement des systèmes CPS-IoT. Pour ceci, nous avons considéré un cas d'étude spécifique tout au long de la thèse qui est les drones.
Dans un premier temps, on s'est focalisé sur les différentes méthodes d'analyse de sûreté de fonctionnement qui sont déjà existantes. Ces méthodes ont fait leurs preuves pour la conception et la réalisation des systèmes embarqués. Tout au long de ce process, on a essayé de répondre à la question suivante: est-ce que ces méthodes existantes sont adéquates pour réaliser les analyses de sûreté de fonctionnement nécessaires pour les CPS-IoT ? On a conclu la nécessité de nouvelles approches pour analyser la sûreté de fonctionnement des systèmes CPS-IoT du fait de la complexité significative de ces systèmes.
Dans un second temps, on a proposé une méthodologie pour l'analyse prédictive de la résilience des CPS-IoT. La résilience est définie comme étant la capacité d'un système à tolérer les pannes, à continuer à fournir le service demandé tout en considérant les différentes contraintes internes et externes au système. On a différencié deux types différents de résilience qui sont la résilience endogène et exogène. La résilience endogène est la capacité inhérente du système à détecter et à traiter les défauts internes et les attaques malveillantes. La résilience exogène est la capacité permanente du système à maintenir un fonctionnement sûr dans son environnement ambiant.
La dernière partie de notre travail a consisté à investiguer l'impact de l'intelligence artificielle sur la sûreté de fonctionnement des CPS-IoT. Plus spécifiquement, on s'est intéressé à comment serait-il possible d'utiliser l'intelligence artificielle pour accroître la sûreté des drones lors de la phase de planification de chemin. Les résultats obtenus ont été comparés avec les algorithmes de planification existants.

Title : A safety approach for CPS-IoT

Keywords : Cyber-Physical Systems, Internet of Things, safety, security

Abstract : For several years now, we have been witnessing a convergence between cyber-physical systems (CPS) and the Internet of Things (IoT). CPS integrate embedded systems with their physical environment and human by ensuring communication between different sensors and actuators. The IoT targets the network and communication protocols between connected objects. This convergence offers application opportunities a variety of vehicles connected to the smart grid as well as the factories of the future.
The purpose of this thesis is to ensure and guarantee the operational safety of CPS-IoT systems. For this, we have considered a specific case study throughout the thesis which is the drones.
Initially, we focused on the different methods of analysis of operating safety that already exist. These methods have made it possible to their proofs for the design and implementation of embedded systems. All throughout this process, we have tried to answer the following question: is it that these existing methods are adequate for performing safety analyses necessary for the CPS-IoT? It was concluded that there was a need to new approaches to analyse the operational safety of systems CPS-IoT due to the significant complexity of these systems.
In a second step, a methodology for predictive analysis of the resilience of CPS-IoT. Resilience is defined as the ability to a system to tolerate failures, to continue to provide the requested service while considering the different internal and external constraints of the system. We have differentiated two different types of resilience which are endogenous resilience and exogenous. Endogenous resilience is the system's inherent ability to detect and treat internal defects and malicious attacks. Exogenous resilience is the system's permanent ability to maintain safe operation in its ambient environment.
The last part of our work consisted in investigating the impact of artificial intelligence on the operational safety of CPS-IoT. More specifically, attention was paid to how artificial intelligence could be used to increase the safety of drones during the path planning phase. The results obtained were compared with existing planning algorithms.

Institut Polytechnique de Paris
91120 Palaiseau, France

Résumé de la thèse en Français

Au cours de la dernière décennie, nous avons assisté à un déploiement croissant des Systèmes Cyber-Physiques (CPS) dans différentes disciplines. Les systèmes cyber-physiques sont des intégrations de calcul avec des processus physiques. Les ordinateurs et les réseaux embarqués surveillent et contrôlent les processus physiques, généralement par des boucles de rétroaction où les processus affectent les calculs et vice versa [1].

Les applications émergentes des systèmes cyber-physiques sont destinées à fonctionner dans une forme distribuée sur une plateforme qui combine le calcul à haute performance avec de grandes catégories de capteurs fournissant une énorme quantité de données. La Commission Européenne promeut le terme ACPS pour désigner ces derniers systèmes, où A signifie autonome, adaptable ou artificiel selon la caractéristique abordée [2]. Malgré cette tendance les questions de sécurité restent un défi important. Un problème majeur est de combiner la présence du système dans son environnement physique et sa caractérisation y compris la conception du logiciel intégré.

L'origine des systèmes cyber-physiques remonte aux systèmes embarqués. Les systèmes embarqués remontent aux années 1960 avec l'invention des circuits intégrés [3]. A partir de ce moment, les systèmes embarqués ont été fortement associés à des applications critiques telles que l'avionique et l'industrie automobile. Les systèmes embarqués ont toujours été tenus à des standards de fiabilité et de prédictibilité plus exigeants que l'informatique à usage général. La fiabilité est définie dans la norme ISO 8402 [4] comme *la capacité d'un élément à effectuer une fonction*

donnée, dans des conditions environnementales et opérationnelles et pour une certaine période de temps. De plus, les utilisateurs ont constaté un niveau de sécurité accru grâce à l'intégration des systèmes embarqués dans leur vie quotidienne. Par conséquent, lors du passage des systèmes embarqués vers les systèmes cyber-physiques, l'attente de fiabilité et d'efficacité ne fait qu'augmenter. En effet, sans amélioration de fiabilité et de prévisibilité, les CPS n'atteindront pas leur plein potentiel et ne seront pas déployés dans des applications critiques.

Problème de recherche

Dans cette thèse, on aborde le problème de la sécurité des systèmes cyber-physiques (autonomes) et on se focalise sur la manière d'inclure la sécurité dès les premières étapes de la conception du système. La problématique de recherche est ensuite déclinée en quatre grandes catégories, qui constituent également notre mode opératoire.

Tout d'abord, on étudie les approches de sûreté de fonctionnement traditionnellement utilisées pour les systèmes embarqués, ensuite on analyse si ces approches conviennent aux systèmes cyber-physiques et on identifie leurs limites - si elles existent. Ce problème de recherche vise également à déterminer s'il existe des paradigmes émergents qui sont plus adaptés à la sécurité des systèmes cyber-physiques.

Deuxièmement, on identifie les facteurs clés pour traiter la sûreté de fonctionnement des systèmes cyber-physiques.

Troisièmement, on propose une approche basée sur les contrats afin de traiter la sûreté de fonctionnement des systèmes cyber-physiques.

Enfin, on considère la tendance croissante qui intègre l'intelligence artificielle dans les systèmes cyber-physiques. On évalue son impact sur la sûreté de fonctionnement. Etant donné l'ampleur du sujet de recherche, on se limite aux systèmes de navigations des systèmes cyber-physiques autonomes.

Contributions

La première contribution de cette thèse porte sur les principales techniques d'analyse de la sûreté de fonctionnement des systèmes embarqués et examine si elles sont adaptées lorsqu'elles sont appliquées aux systèmes cyber-physiques. Bien que notre analyse ne soit pas exhaustive, on converge vers la conclusion que de nouveaux paradigmes devraient être introduits. Parmi ceux-ci, la notion de *résilience* semble plus appropriée lorsqu'on considère la sûreté de fonctionnement des systèmes cyber-physiques. Cette première contribution a été publiée dans l'Ada User Journal [5].

Dans la deuxième contribution, on étudie les facteurs clés de la sûreté de fonctionnement qui doivent être pris en compte lors des analyses de sûreté des systèmes cyber-physiques. On converge indépendamment vers les résultats fournis dans [6], où l'auteur présente une nouvelle façon d'aborder la sûreté de fonctionnement de systèmes aussi complexes que les systèmes cyber-physiques. En considérant les drones comme cas d'étude et en se concentrant sur les applications civiles de ces systèmes, on distingue trois facteurs principaux qui sont: le drone (le CPS), le(s) opérateur(s) humain(s) et l'environnement dans lequel le système est utilisé. Ces facteurs clés sont fortement liés et leur connexion doit être prise en considération lors de l'analyse de sûreté des systèmes cyber-physiques. Les résultats de cette contribution ont été publiés dans la conférence Digital Avionics Systems Conference (DASC) [7].

La troisième et principale contribution de cette thèse est la proposition d'une méthodologie (basée sur les contrats) pour l'analyse de la résilience des systèmes cyber-physiques autonomes. La résilience d'un système est définie comme étant sa capacité à résister aux perturbations externes de son environnement et à continuer à fournir le comportement attendu. En abordant la résilience des systèmes cyber-physiques, on identifie deux types différents qui sont: endogène et exogène. La résilience endogène se traduit par la capacité du système à traiter les défaillances internes et à résister aux cyber-attaques. La résilience exogène repose sur la capacité du système à fonctionner en toute sécurité dans son environnement ambiant. La méthodologie proposée est la suivante; on commence par définir des contrats afin de préciser le comportement souhaité du système. Ensuite, on représente

le système sous forme d'un réseau d'automates temporisés et on utilise l'outil UPPAAL pour vérifier les contrats spécifiés et valider la résilience endogène du système. En parallèle, la résilience exogène est validée en utilisant l'outil CAT (Contract Analysis Tool) pour vérifier les contrats sur des modèles 3D. Les résultats de cette contribution ont été publiés dans le journal IEEE Access [8].

La quatrième contribution de cette thèse aborde le problème de la sûreté de fonctionnement et de l'intégration de l'Intelligence Artificielle (IA) dans la navigation autonome des systèmes cyber-physiques autonomes. Plus en détail, on introduit les algorithmes génétiques dans la planification de chemins pour l'usage des drones dans un milieu urbain. Pour évaluer les résultats obtenus, on spécifie une base de référence mesurable basée sur l'algorithme bien connu A*. Nos résultats montrent que les algorithmes génétiques améliorent la sûreté de système par rapport à l'algorithme A*. Ces résultats ont été récemment publiés dans la International Conference on Cyber-Physical Systems (ICCPS) [9].

Structure de la thèse

Ce manuscrit de thèse est structuré comme suit. Chapitre 1 "Systèmes cyber-physiques et sûreté de fonctionnement" présente les concepts impliqués dans cette thèse. On commence par la définition bien connue des systèmes cyber-physiques, leurs principales caractéristiques et leurs applications. Ensuite, on présente l'état de l'art sur l'évolution des concepts de sûreté de fonctionnement. On montre les limites des analyses de sûreté de fonctionnement traditionnelles pour les systèmes embarqués et on identifie les facteurs clés de sûreté pertinents pour l'analyse de sûreté des systèmes cyber-physiques. Parmi ceux-ci, on souligne plus tard l'importance du concept de la résilience. Ensuite, on donne quelques notions et définitions de base sur les approches basées sur les contrats (CBD) et les réseaux d'automates temporisés. Ces définitions sont essentielles pour appréhender les chapitres suivants du manuscrit.

Chapitre 2 "Une nouvelle méthodologie pour l'analyse de la résilience des systèmes cyber-physiques autonomes" présente une méthodologie basée sur les approches par contrats pour l'analyse de la résilience des systèmes cyber-physiques autonomes. Deux types de résilience sont détaillés : endogène et exogène. Cha-

cune est analysée séparément. Une section est consacrée au retour d'expérience sur les travaux de développement et de réalisation des différents modèles. Le cas d'étude est l'application civile des drones dans des situations d'urgence médicale.

Chapitre 3 "Sûreté et navigation autonome" présente une étude de l'impact de l'intelligence artificielle et des algorithmes génétiques dans la sûreté de fonctionnement des systèmes cyber-physiques autonomes. Le cas d'étude présenté est la navigation des drones dans un milieu urbain avec une forte densité d'obstacles.

On termine le manuscrit par une conclusion générale et des perspectives des travaux de thèse.

Contents

List of Figures

General introduction

In the last decade, we have witnessed an increasing deployment of Cyber-Physical Systems (CPS) in different disciplines. Cyber-Physical Systems are integrations of computation with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa [1].

The emerging applications of cyber-physical systems are destined to run in a distributed form on a platform that combines high performance computing with broad classes of sensors providing a big amount of data. The broad majority of these new applications can be classified as "distributed sense and control systems" that go substantially beyond the "compute" or "communicate" functions, traditionally associated with information technology [1]. These applications have the potential to influence how we deal with a large range of problems today. For example, security and safety including surveillance, energy management and distribution, efficient and reliable transportation. Actually, CPS applications cover autonomous functionalities and include artificial intelligence [13]. The European commission promotes the term ACPS to refer to these latter systems, where A stands for autonomous, adaptive or artificial with respect to the tackled feature [2]. Despite this trend, safety issues remain a thorny challenge. One major problem is to combine the system's presence in the physical surroundings and its characterization, including the embedded software design.

The origin of cyber-physical systems goes back to embedded systems. Embedded systems date back to the 1960s with the invention of integrated circuits

(IC) [3]. From this point on, embedded systems have been strongly associated with critical applications such in avionic and automotive industry. Embedded systems have always been held to a higher reliability and predictability standards than general-purpose computing. Reliability is defined in ISO 8402 [4] as *the ability of an item to perform a required function, under given environmental and operational conditions and for a stated period of time.* Customers have witnessed an increased safety level with the integration of embedded systems in their daily life. For example, since anti-lock braking system (ABS) have been integrated in cars, the number of accidents caused by locked up wheels during braking has drastically decreased. Consequently, in the transition from embedded systems to CPS, the expectation of reliability and efficiency will only increase. In fact, without improved reliability and predictability, CPS will not reach their full potential and will not be deployed in critical applications such as traffic control and automotive safety.

Research Problem

In this thesis, I address the research problem of safety for (autonomous) cyber-physical systems and I focus on how to include safety since the early stages of system design. The addressed research problem is further declined into four main categories, which constitute also our *modus operandi*.

Firstly, I investigate if the safety approaches traditionally used for embedded systems can be suitable for CPS and I identify their limits - if any. This research problem equally addresses if there are emerging paradigms which are more suitable for CPS safety.

Secondly, I identify which are the key factors to deal with CPS safety.

Thirdly, I ask ourselves which are the approaches, techniques and tools that better meet CPS safety.

And finally, I consider the increasing trend, which integrates artificial Intelligence into CPS. I evaluate its impact on safety. Given the extensively research subject, I limit the study to autonomous navigation systems.

Contributions

The first contribution of this thesis addresses the main techniques for safety analysis of embedded systems and investigates if they are suitable when they are applied to CPS. Although my analysis is not exhaustive, I converge towards the conclusion that new paradigms should be introduced. Among them, the notion of *resilience* seems more appropriate when considering CPS safety. This first contribution has been published in Ada User Journal [5].

In the second contribution, I investigate the safety and reliability key factors that need to be taken into account during the safety analysis of cyber-physical systems. I independently converge towards the results provided in [6], where the author introduce a new way of tackling safety of systems as complex as CPS (Safety I, Safety II and Safety III).

To this end, I consider drones as a use case and I focus on the civilian applications of these systems. I distinguish three main key factors: the drone (the CPS), the human operator(s) and the environment in which is the system is operated. These key factors are strongly linked and their connection must be taken into consideration during the safety analysis of CPS. This contribution results were published on the Digital Avionics Systems Conference (DASC) [7].

The third and main contribution of this thesis is the proposal of a (contract-based) methodology for analysis of resilience in CPS. A system's resilience is defined as its capacity to withstand external disturbances from its environment and to continue to provide the required outcomes. When addressing CPS resilience, I identify two different types which are: endogenous and exogenous. Endogenous resilience is reflected in the system's capability of processing internal faults and resisting to cyber-attacks. Exogenous resilience relies on the system's ability to safely operate in its ambient environment. I start by defining contracts in order to specify the wished behaviour of the system. Then, I represent the system as a network of timed automata and I use the tool UPPAAL to verify the specified contracts and validate the endogenous resilience of the system. In parallel, exogenous resilience is validated using CAT (Contract Analysis Tool) to verify the contracts over 3D models. This contribution results were published in IEEE Access Journal [8, 14].

The fourth contribution of this thesis addresses the problem of safety and the integration of Artificial Intelligence (AI) into autonomous navigation for ACPS. More in detail, I introduce genetic algorithms to select the path navigation under safety constraints. To evaluate the obtained results, I specify a measurable baseline based on the well-know A* algorithm. My result shows that genetic algorithms improve the safety with respect to the baseline. This results has been recently published in the International Conference on Cyber-Physical Systems (ICCPS) [9].

Structure of the Thesis

This thesis manuscript is structured as follows.

Chapter 1 "Cyber-Physical Systems and Safety" introduces the concepts involved in this thesis. I start by the well-know definition of CPS, its main features and their applications. Then, I introduce the state of the art over the evolution of safety concepts. I show the limits of the traditional safety analysis for embedded systems and I identify the safety key factors relevant to the safety analysis of CPS. Among these later, I highlight the importance of resilience. Afterwards, I provide some basic notions and definitions about Contract-Based Design (CBD) and networks of timed automata. These definitions are essential in order to apprehend the following chapters of the manuscript.

Chapter 2 "A new methodology for analysis of resilience in ACPS" introduce the promoted contract-based methodology for the analysis of resilience in CPS. Two types of resilience are detailed: endogenous and exogenous. Each one is analyzed separately. A section is dedicated to the feedback of the tool implementation and the model realization. The studied use case is drone civilian application in urgent medicine situations.

Chapter 3 "Safety and autonomous navigation" investigates the impact of artificial intelligence and genetic algorithms in the safety of autonomous cyber-physical systems. The studied use case is drone navigation in urban environment.

I end the manuscript with a general conclusion and the thesis's prospectives.

Chapter 1

Cyber-Physical Systems and Safety

Contents

1.1 Introduction

In this chapter, I introduce the definitions and fundamentals related to cyber-physical systems and safety. I also introduce the state of the art of existing methodologies involved in safety analysis. This fundamentals understanding is necessary for comprehending the contributions presented later on in this manuscript.

Thus, in this chapter, I start by defining what is a cyber-physical system. After outlining the main features of these systems as well as their applications, I address the discipline of safety engineering. I give a brief history and background of safety, then I discuss the different processes for system safety analysis. I also identify key factors for CPS safety by considering drones as a use case. Finally, I provide some basic notions and definitions about Contract-Based Design and networks of timed automata which are essential to understand the rest of the manuscript.

1.2 Cyber-Physical Systems

1.2.1 Definition of a system

Oxford dictionary defines a system as "a set of things working together as parts of a mechanism or an interconnecting network; a complex whole".

The International Council on System Engineering (INCOSE) proposes another definition: "A system is a construct of different elements that together produce results not obtainable by the elements alone."

In [15], the author defines a system as "any group of interacting, interrelated, or interdependent parts that form a complex and unified whole that has a specific purpose."

The common point between all the definitions above is that all systems are associations of elements (components) in interaction in order to bring out new functionalities. It can be concluded that interactions are the very essence of systems. As a consequence, new properties appear, beyond those specific to the simple components.

1.2.2 Definition of a Cyber-Physical System

The original definition dates back from 2006 when a group of academics from the United States realized that embedded systems were evolving into systems where physical aspects played a fundamental role. The emergence of new and complex systems such as smart electricity grids and autonomous vehicles has been a source of interest for researchers. These systems are characterized by their distributed aspect where both physical and software sub-systems are connected in order to provide a particular service. The interaction between the intelligence provided by distributed processors that were interconnected with networks of growing complexity AND the physical world has become necessary to be taken into account. Deliverable "Characteristics, capabilities, potential applications of Cyber-Physical Systems: a preliminary analysis" [16] gives a well-summarized overview on how the term "cyber-physical system" has arised. The first definition proposed back in 2006 by the group of academics is the following: "*The integration of physical systems and processes with networked computing has led to the emergence of a new generation of engineered systems: Cyber-Physical Systems (CPS). Such systems use computations and communication deeply embedded in and interacting with physical processes to add new capabilities to physical systems. These CPS range from minuscule (pace makers) to large-scale (the national power-grid).*"

The same deliverable [16] proposes a simple and yet general definition of cyber-physical systems: "*A CPS consists of computation, communication and control components tightly combined with physical processes of different nature, e.g., mechanical, electrical, and chemical.*" The authors consider that a simpler definition would be more effective to convey to the public and to the policy makers what CPS are all about.

Edward A. Lee in [17] defines cyber-physical systems as "*an integration of computation with physical processes. Embedded computers and networks monitor and control the physical processes, usually with feedback loops where physical processes affect computations and vice versa.*" The cyber part includes all the software, the code and the executed algorithms. As an intellectual challenge, CPS is about the intersection, not the union, of the physical and the cyber. It is not sufficient to separately understand the physical components and the computational components.

Instead, it is essential to understand their interaction [17].

Lately, a new concept known as Internet of Things (IoT) have emerged and have been commonly associated with CPS. This concept emerged primarily from a networking and information technology perspective. "*The term Internet of Things is used as an umbrella keyword for covering various aspects related to the extension of the Internet and the Web into the physical realm, by means of the widespread deployment of spatially distributed devices with embedded identification, sensing and/or actuating capabilities*" [18]. In [19], the authors provide a comparison between these two concepts by highlighting the similarities as well as the differences.

Broadly, a CPS corresponds to a system integrating electronics and software, sensors and actuators and equipped with communication capabilities. A CPS interacts with its environment in which in takes data, processes it and, through a feedback loop, controls or influences the process in which it is associated. Through its communication capabilities, a CPS can act in collaboration with other systems and/or exchange data with remote systems. The communication can be either wired or wireless. A CPS is characterized by a high degree of complexity that is partly intrinsic and mainly due to interconnection and dynamic interactions with other systems.

1.2.3 Autonomous Cyber-Physical Systems

During these last years, a trend towards integrating Artificial Intelligence (AI) with CPS has been noticed. AI is the simulation of human intelligence processes by computing systems. These processes include the acquisition of information, its processing and reaching conclusions based on the processing carried out. Remarkable success has been achieved by AI and machine learning algorithms in solving complex tasks previously thought to require human intellect. The thrust to achieve trustworthy autonomous systems, which can attain goals independently in the presence of significant uncertainties and for long periods of time without any human intervention, has always been enticing. Significant progress has been made in the domains of both software and hardware in order to meet these objectives. This emergence has led to a concerted effort to utilize AI in embedded software for CPS applications giving place to what is called Autonomous Cyber-Physical

Systems (ACPS) [20].

In 2014, the Cyber-Physical European Roadmap and Strategy (CyPhERS) [16, 21] pointed out AI as a distinctive characteristic of ACPS . In 2018, the Platform4CPS [22] European project introduced a list of recommendations together with the main scientific topics and business opportunity for ACPS markets [23]. Among the main topics, the project highlights the importance of ACPS (especially those including AI) and their impact on the incoming period with respect to several aspects and disciplines.

The design and development of ACPS requires the convergence of the cyber side (computing and networking) with the physical side (sensing and actuating) and AI. This convergence is extremely challenging especially when it comes to decision making under uncertainty. In an autonomous system, uncertainties can arise from the operating environment, adversarial attacks, or within the system itself.

ACPS technologies are expected to bring large-scale improvements through new products and services across a myriad of applications ranging from healthcare to logistics through manufacturing, transport and more. The technical foundations and assumptions on which traditional safety engineering principles are based worked well for human-in-the-loop systems where the human was in control, but are inadequate for ACPS, where autonomy and AI are progressively more active in this control loop. Incremental improvements in traditional safety engineering approaches over time have not converged to a suitable solution to engineer ACPS, having increased levels of autonomy and adaptation. In addition to physical integrity, safety in ACPS is tightly related to ethical and legal issues such as trustworthiness, responsibility, liability and privacy. This aspect becomes more pronounced in the cases where the ACPS fails to accomplish its mission or if there is an accident. Determining the responsible of the incident becomes a challenging task.

1.2.4 CPS features

As I exposed earlier, a CPS corresponds to a system integrating electronics and software, sensors and actuators and equipped with communication capabilities. In

this section, I highlight the main distinctive aspects of CPS which are autonomy levels, uncertainty, networking and complexity. Is is important to mention that this list is not exhaustive. The goal of this section is to expose the most relevant features of CPS in order to comprehend what makes them so particular compared to other systems.

1.2.4.1 Autonomy levels

The first feature is the **autonomy levels**. CPS are typically designed to act more or less independently of humans, even if they may be triggered by human inputs, including shared control. Shared control is used when the CPS performs a complex function that can be divided into several sub-functions. Each sub-function is assigned to a particular operator that can be either human or automated. A great example of this concept is an airplane with remote control systems. The remote control system manages certain functions of the plane in order to let the human operator (the pilot) to make decisions such as direction and altitude. The remote control system is essential to keep the plane flying, without it the pilot would need plenty of adjustments. Shared control has also its challenges: it is crucial to clarify who is in control at any time to make sure that unintended control does not take place. Traditionally, systems are designed to be totally controlled by human operators. The tasks performed by the system are well defined, so the system's behavior is delimited in a certain perimeter.

In [24] and [25], the authors associate robots autonomy with Human-Robot Interaction (HRI) concept. They introduce a correlation between the level of HRI and the autonomy level of the robot. As showed in figure 1.1, the Autonomy Levels of Unmanned Systems (ALFUS) is illustrated on a scale from zero to ten; zero being associated to a system fully controlled by a human operator and ten being associated to a fully-autonomous system.

Recent cyber-physical systems are endowed with different levels of autonomy. For simplification reasons, only three levels of autonomy are considered in this thesis; fully human-controlled, semi-autonomous and fully-autonomous CPS. Semi-autonomous systems are controlled by human operators, but not completely. Certain tasks, mainly the critical ones, are usually performed by the human operator

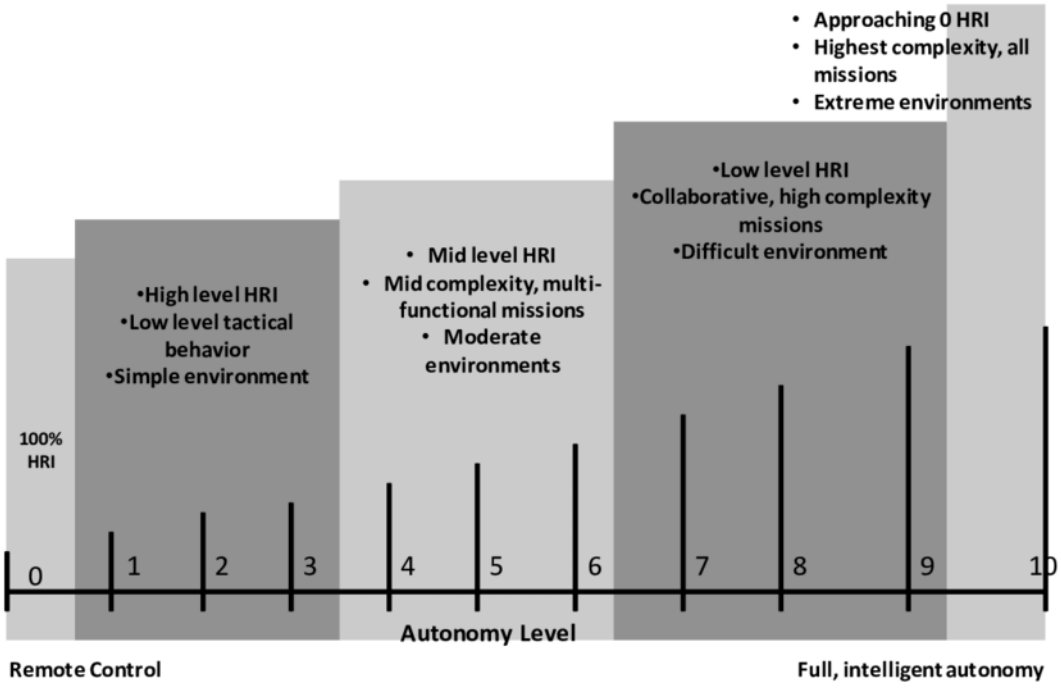


Figure 1.1: Autonomy Levels for Unmanned Systems (ALFUS) (figure from [25])

while less critical tasks are performed autonomously. One example illustrating semi-autonomous systems is the civil airplanes used nowadays. Critical tasks such as take-off or landing are performed by the pilot while cruise control is performed autonomously. For semi-autonomous cyber-physical systems, the human operator (the pilot in this case) always have the priority. He can intervene at any moment to stop the autonomous task and take over the control of the system.

Fully-autonomous systems are totally selfsustaining. All tasks (critical and not critical) are performed autonomously. The human operator only plays the role of a supervisor and can intervene when he considers that something is wrong. One example of the fully-autonomous cyber-physical systems is the autonomous vehicle. Fully autonomous vehicles are designed to be totally stand-alone. The driver only indicates the destination location and supervises the autonomous car driving itself. The autonomy feature complicates the safety analysis of such systems, because their behavior tend to be unpredictable and non-deterministic.

1.2.4.2 Uncertainty

The second important feature of CPS is the **uncertainty**. Uncertainty is intrinsic in CPS due to novel interactions of embedded systems, networking equipment and human operators. Uncertainty is a term that has been used in various fields such as philosophy, physics, statistics and engineering to describe a state of having limited knowledge where it is impossible to exactly tell the existing state, a future outcome or more than one possible outcome [26]. Various uncertainty models have been proposed in the literature from different perspectives for various domains. The authors of [27] review uncertainty from an ethics perspective claiming that "*Uncertainties challenge the central claim of science: that all problems are presumed to be solvable by research*". From this perspective, uncertainties are classified as objective uncertainty and subjective uncertainty, both of which are further classified into subcategories to support decision-making. In healthcare, uncertainty has often been defined as "the inability to determine the meaning of illness-related events" [28].

Uncertainty is progressively receiving attention in recent years in both system and software engineering, especially for CPS, which are required to be more and more context aware [29–31]. Moreover, CPS inherently involves tight interactions between various engineering disciplines, information technology, and computer science. In [32], the authors confirm that uncertainty is unpreventable in the behavior of a cyber-physical system given its close interaction with its physical environment. Predicting the exact behavior of the physical environment of a CPS is not viable, and a common practice is to make assumptions about the physical environment during the design and testing phases of the CPS. The correct behavior of a CPS is only guaranteed when such assumptions prove to be true. Given the complexity of problems being solved by CPS in critical domains, these systems must function safely even when experiencing uncertainty in their physical environment to avert any harm.

To understand what uncertainty is in the context of software engineering, a conceptual model, named as U-Model [33], has been proposed to define uncertainty. The authors of this model expressed facing several challenges when they addressed uncertainty both on the understanding and the quantifying levels. Due

to the interdisciplinary nature of CPS, it is quite difficult to precisely understand uncertainties. It is mainly because uncertainties not only exist in software, but also in hardware, communications, humans and the interaction among them. Comprehending uncertainty requires a wide range of knowledge across different disciplines and also requires the knowledge of various components of CPS, their interactions and the overall functionalities of CPS as a whole.

1.2.4.3 Networking

The third feature of CPS is **networking**. CPS are often complex and composed of several components continuously exchanging data. These components use different networks to communicate and some components may even be cloud-based. Networking feature is strongly related to **security**. A number of security-related aspects such as intrusion detection and prevention, privacy, anonymity, and so on need to be handled in CPS [34]. Handling these aspects is specially challenging in the context of CPS. For example, a distributed attack may not exploit the weaknesses of the separate components of the system, while combined together, may have catastrophic consequences. The problem of defining secure control and the challenges in securing CPS are further outlined in [35]. The authors outline the manner in which the developments from the fields of information security, sensor network security and control theory can be utilized to ensure survivability of CPS. In [36], the authors further discussed the possible threats and their consequences. Compared with traditional IT security, security in CPS poses different challenges, since installing new software patches is not straightforward due to the time-critical and heterogeneous nature of operation of CPS. Most CPS are designed to operate in dynamic environments with many variables. Few works actually took advantage of the CPS environments in order to make them more secure. In [37], the authors discussed a method to generate secret keys in smart homes. In CPS, the unpredictable and erratic nature of physical environments present a rich source of randomness. By leveraging it, the secret key generation algorithm can be made smarter and can be used to make secure wireless communications. This example illustrates how it is possible to take advantage of the randomness aspect of the physical environment of CPS in order to provide more secure communication.

1.2.4.4 Complexity

As well described in [38], complexity for CPS is highly multifaceted, arising from the CPS itself, its environment and its design process. The authors discuss CPS complexity by identifying four different facets which are:

- The environment in which the CPS is operating
- The software components of the CPS, where software-defined behaviors lead to very large state-spaces. This means that the system behavior tend to get unpredictable and not understandable due to the number of states in which the system can be.
- The physical components of the CPS (such as sensors and actuators), where an important source of complexity arises from side effects (e.g. friction-induced thermal effects between surfaces in contact) [39].
- Interactions between the cyber and physical components. Combining cyber and physical components enables the system to perform hard tasks and offers unprecedented possibilities for executing a wide range of missions. On the other hand, such systems are characterized by and increasingly complex behaviors including a multitude of possible faults and failure modes.

1.2.5 CPS applications

"The potential of CPS to change every aspect of life is enormous , concepts such as autonomous cars, robotic surgery, intelligent buildings, smart electric grid, smart manufacturing, and implanted medical devices are just some of the practical examples that have already emerged [40]". CPS are enabling a new generation of "smart systems" and the economic impact could be huge. The innovative technologies emerging from the combination of the cyber and physical worlds could provide an engine for multiple domains. In this section, I detail a non-exhaustive list of CPS key application areas.

1.2.5.1 Smart manufacturing

Manufacturing constitutes a broad domain, encompassing many levels and processes, from low level material shaping processes via production machines and cells, to operations management and virtual factory planning, sometimes also encompassing logistics [41]. Today, the manufacturing industry is aiming to improve competitiveness through the convergence with Information and Communication Technologies (ICT). Combining CPS capabilities with ICT may lead to the 4th industrial revolution, frequently noted as Industry 4.0 [42]. According to the Federal Ministry of Education and Research of Germany: "Industry is on the threshold of the fourth industrial revolution. Driven by the Internet,the real and virtual worlds are growing closer and closer together to form the Internet of Things. Industrial production of the future will be characterized by the strong individualization of products under the conditions of highly flexible (large series)production,the extensive integration of customers and business partners in business and value-added processes,and the linking of production and high-quality services leading to so-called hybrid products [42]".

Smart manufacturing refers to the use of embedded software and hardware technologies to optimize productivity in the manufacture of goods or delivery of services [43]. The National Institute of Standards and Technology (NIST), which is an agency of the U.S Department of Commerce, defines smart-manufacturing as "fully-integrated and collaborative manufacturing systems that respond in real time to meet the changing demands and conditions in the factory, supply network, and customer needs [44]."

The main drivers for smart manufacturing are:

- Improving products quality
- Improving safety by reducing accidents caused by human errors
- Improving industry's competitiveness

Manufacturing is strongly characterized by and influenced by CPS technologies. As such manufacturing already provides a domain featuring advanced CPS technologies with automation provided by industrial robots, mixed continuous and

Figure 1.2: Smart manufacturing (Image: blog.integral-system.fr)

discrete control, and hierarchical distributed control systems [41]. The strong potential for growth in relation to CPS is clearly indicated by research and innovation roadmaps such as the Factory of the future roadmap [45], and studies on potentially disruptive technologies [46]. In the latter study on potentially disruptive technologies and their potential impact on business, many of the key covered technologies relate closely to manufacturing including Internet of Things, Cloud Technology, Advanced Robotics, 3D printing and advanced materials.

1.2.5.2 Transportation and mobility

CPS in the transportation industry are strongly related to our daily life and play an important in society. The transportation domain includes various sectors: automotive, aerospace and railway. Each sector is a manifestation of a CPS, and includes both vehicles and infrastructural components. In this section, we will focus on the automotive sector and more specifically the autonomous vehicle.

Automated driving is seen as one of the key technologies and major technological advancements influencing and shaping the future mobility of people.

The main drivers for higher levels of automated driving are [47, 48]:

- Improving safety by reducing accidents caused by human errors.

- Contributing to the optimization of the traffic flow by increasing transport system efficiency and reducing time in congested traffic.
- Reducing fuel consumption and CO_2 emissions.
- Improving user's comfort by enabling his freedom for other activities when automated driving is enabled.
- Ensuring mobility for all, including elderly, impaired and non-confident users.

Several forecasts predict a limited availability for automated driving functions in 2020 (with different levels of autonomy as discussed in Section 1.2.4.1) and a wide availability by 2040 including high and full automation. Today's Advanced Driver Assistance System (ADAS) like Automatic Cruise Control (ACC), Lane Departure Warning (LDW), or Pedestrian Detection (PD) will form the backbone of tomorrow's mobility. In this regard, the Deserve European Project [49], between 2012 and 2015, designed and developed a Tool Platform for embedded ADAS. This platform provided an environment for ADAS design, development, pre-validation and pre-certification of software and hardware modules to be integrated in ADAS applications.

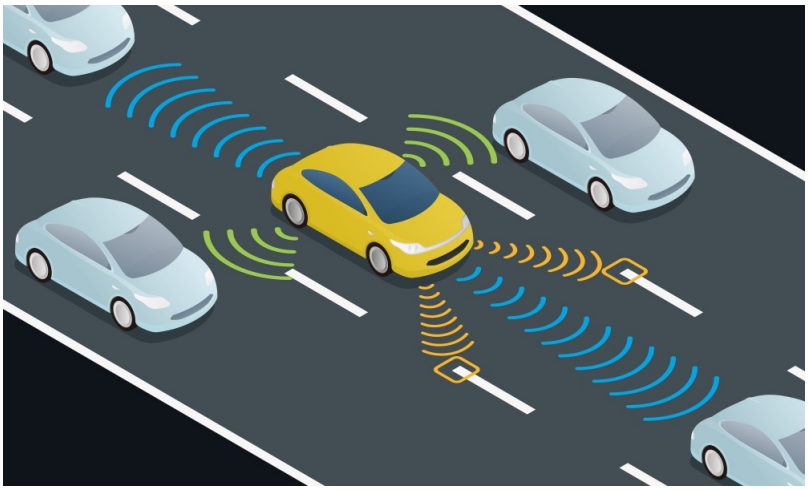


Figure 1.3: Autonomous car (Image: shutterstock.com)

Vehicles are designed to communicate with each other as well as with the infrastructure. Vehicle-to-Vehicle (V2V) communication allows vehicles to exchange

relevant information like local traffic data (e.g. nearby accidents) and about their driving intention. Vehicle-to-Infrastructure (V2I) communication will be used to optimize the road network usage and thereby helps to reduce environmental pollution.

Safety-critical applications in cooperative vehicular networks require authentication of nodes and messages. Yet, privacy of individual vehicles and drivers must be maintained. Pseudonymity can satisfy both security and privacy requirements. In survey [50], the authors detail the challenges and requirements for such pseudonym mechanisms, propose an abstract pseudonym lifecycle, and give an extensive overview and categorization of the state of the art in this research area. In [51], the authors present SEROSA, a service-oriented security and privacy-preserving architecture for vehicular communication. By synthesizing existing vehicular communication standards and web services, the proposed architecture provides comprehensive identity and service management while ensuring interoperability with existing service providers.

The role allocation between human drivers and automated driving systems is specified by six levels of driving autonomy: no automation, driver assistance, partial automation, conditional automation, high automation and full automation.

The option to switch to "automated driving mode" will give drivers more freedom in terms of individual mobility. With the market introduction of highly automated vehicles by 2020-2025, drivers will be able to manage their driving times better [47]. At the same time, an automatically controlled vehicle will be even safer thanks to the increased interaction with itself and its environment. Futhermore, the energy management and driving characteristics of the vehicle will be optimized enabling more energy-efficient driving. Highly automated road transport will have a significant impact on our mobility behavior, road safety and traffic efficiency.

An autonomous vehicle will need to be able to carry out tasks such as:

- Understanding complex and dynamic unknown environments and avoiding obstacles that can be either static or mobile.
- Understand road signs and enforce traffic regulations such as speed limit and blocked roads.

- Communicating and exchanging data with road infrastructure and with other vehicles.
- Execute commands in real time.

1.2.5.3 Unmanned Aerial Vehicle

Unmanned Aerial Vehicles (UAV) are getting increasingly used for different applications. In the near future, millions of unmanned aircrafts are expected to be rapidly deployed in diverse sectors of our daily life performing wide-range activities from delivering a package to surveillance and reconnaissance or environmental monitoring [52]. In [53], the author discusses a market analysis for commercial UAV, stating that consumer drone shipments will hit 29 million with a Compound Annual Growth Rate (CAGR) of 31.3% by 2021 and enterprise drone shipments will reach 805,000 in 2021 with a CAGR of 51% [53].

In 2013, Amazon founder Jeff Bezos predicted that drone delivery would be a reality within five years. The billionaire entrepreneur had been right but had probably not anticipated the fact that the first company to offer this new service would be one of its direct competitors: Google. No later than April 2019, Google's subsidiary dedicated to UAV delivery, Wing, has obtained the status of "air carrier" by the Federal Aviation Administration (FAA), the American government agency responsible for civil aviation regulations. In concrete terms, this new status, coveted by other companies such as Uber or UPS, allows Google to bill customers for UAV delivery. A first. "This is an important step forward in testing and safely integrating drone into our economy", said Elaine Chao, U.S Secretary of State for Transportation [54]. The FAA has already granted temporary authorizations to several UAV delivery companies for demonstration or short distance delivery. On the other hand, this is the first time that the American regulator has granted a company specializing in UAV the same status as charter companies or air cargo carriers.

Moreover, the ample availability of affordable drones is leading to large amounts of drones being sold for civilian uses, especially when drones are being equipped with high quality cameras and many other sensors which make them adaptable to a variable set of civilian applications [7].

Figure 1.4: Unmanned Aerial Vehicle (UAV) (Image: futura-sciences.com)

For example, drones were a key tool to help Paris firemen fighting the fire that spread in Notre-Dame church on 15 April 2019. The goal of using drones in similar situations is to better orient fire hoses according to the different sources of fire. Drones allowed the establishment of a flame strategy with a view of the sky offered by an aerial vehicle that is more flexible and economical than a helicopter. The French Ministry of the Interior's air force drones, as well as those of the French Ministry of Culture ,were deployed. The Paris Fire Brigade does not (yet) have its own fleet of remotely operated drones, although they are very useful for operating at such high sites. The Chinese manufacturer DJI confirmed that it was two of their drone, Mavic Pro, that had come into action under the guidance of trained pilots. The drones are equipped with a 4K camera and a thermal camera that produce a double view in order to locate from the sky the still hot spots after the fire is extinguished. In theory, any flight over Paris is prohibited and even technically impossible because of the geo-fencing system that uses the GPS of the quadricopters to block take-offs from and to no-fly zones. However, the manufacturer can unblock these banned flight zones at the request of the authorities as it seems to have been the case of Notre-Dame fire [55].

A drone will need to be able to carry out tasks such as:

- Understanding complex and dynamic unknown environments and avoiding obstacles that can be either static (such as buildings) or mobile (such as people or other unmanned aircraft).
- Understanding where the aircraft is exactly positioned within such environments.
- Ensuring failure containment including the sensors failure (wrong data or data loss), weak signal or total loss of communication. This also include the failure of the hardware components of the drone. This failure containment can be resolved using the redundancy of the sensors and the validation of the data provided by the sensors (data validation).
- Executing commands in real time.
- Embedding sufficient power to maintain movement, to implement the controls, and to operate sensors and data-feeds, for the duration of the flight.
- Ensuring sufficient physical robustness to withstand threatening events, such as wind-shear, lightening and turbulence.

1.3 Safety

1.3.1 History and background

The need for safety has always been a part of the human life. One of the earliest written references to safety is from the Code of Hammurabi, around 1750 BCE. His code stated that if a house was built and then fell due to poor construction, resulting in the death of the owner, then the builder himself would be put to death. Afterwards, different notions of safety have succeeded each other notably in the maritime domain. Some of the first maritime safety regulations, came about around 1255 in Venice, stating that a ship's draught could not be exceeded and must be verified by visual inspection before release. Around 1834, Lloyd's Register of British and Foreign Shipping was created, institutionalizing the concept of safety and risk analysis. In response to the sinking of the Titanic, the

International Convention for the Safety of Life at Sea treaty was passed in 1914, stipulating that the number of lifeboats and other safety equipment must be commensurate with the number of passengers on board the ship. The German safety certification company, TUV Rheinland, was founded in 1872, providing technical safety certification services. In 1877, the U.S. Commonwealth of Massachusetts passed a law to safeguard machinery and also created employers liability laws. Underwriters Laboratory was founded in Illinois, United States, in 1894, creating one of the most recognized product testing, certification, and standard bodies in the world [56].

Around the 1920s, private companies started to create formalized safety programs. The early 1930s was the beginning of the implementation of accident prevention programs across the United States. By the end of the decade, the American National Standards Institute had published hundreds of industrial manuals. It was in the 1930s that the first collection of statistical information on engines and aircraft accidents has been conducted in the air transport sector. Between 1939 and 1942, the very first quantified objectives given by Canadian Infantry Brigade were evaluating rates of failure up to 10^{-5}/h for aircraft and 10^{-7}/h for their structures. Later in the 1940s, reliability techniques began to develop. One of the earliest concept definition for system safety (looking at safety from a system perspective) first appeared at the *Fourteenth Annual Meeting of the Institute of Aeronautical Sciences* in New York City in January 1946.

In the 1950s, the concept of maintenance [57] appeared. The first-ever human reliability studies have been carried out for the new nuclear power plants. At the same time, collecting data on electronic reliability was getting increasing attention.

From 1960 onwards, the aeronautics and space industries carried out analyses of components failures. The US Department of Defense (DoD) promoted the first real requirements for Operational Safety following accidents on missiles. In 1961, Bell Laboratories used the new concept of cause tree on the Minuteman missile project [58].

In 1962, the French Academy of Sciences welcomed the word “reliability” in its terminology.

From 1970, the first studies on software reliability [59] have been carried and most of them were in the nuclear field. Then, gradually, safety techniques widely

spread and were extended to more and more areas: chemicals, railways, automobiles, and all types of major industrial sectors.

To conclude this brief summary of safety's history and background, it is relevant to say that nowadays the safety of property and people has never been more important, phenomenon increased by media and ecological pressure around major accidents. Moreover, from a marketing point of view, ensuring the safety of a product is essential in order to promote it and a simple accident may endanger the whole process. For example, the Air France Concorde went from a 27-year record of zero crashes to a single crash in July 2000, killing 100 passengers and 9 crew members, becoming one of the worst aircraft-type safety records (due to its low flights frequency). Another example is the crash that occurred at March 2018, where a Tesla model X crashed into a roadside and caught fire resulting in the death of the driver. Autopilot was engaged at the time of the accident. The company later published a report stating that the driver had recieved several visual and audible warnings in the drive and the driver's hands were not detected on the wheel for six seconds prior to the collision. This accident rose several questions about self-driving cars and caused the company to temporarily suspend all self-driving tests in North America [60].

1.3.2 Definition

Safety analysis is a generic term for study of the system, identification of dangerous aspects of the system, and their correction. Its purpose is to maintain the proper functioning of a system, a product or one of its components, in the time, throughout its life cycle. Safety study has become fundamental for critical systems, where a malfunction can create a significant human or financial losses. It also applies to software, where again a malfunction can cause financial or social risks.

The safety norms CENELEC 50126 [61] introduces safety as part of RAMS, which is a combination of Reliability, Availability, Maintainability and Safety. More in general, RAMS contributes to what is called **dependability** [10,62]. The dependability of a system is its ability to deliver a service that can be justifiably trusted. This definition stresses the need for justification of trust. Addressing

dependability in a structured and comprehensive manner is achieved on the basis of the dependability tree [62].

Dependability is an integrating concept that encompasses many attributes. I capitalize on the work done in [62] which included the following list of attributes:

- **availability**: readiness for correct service.
- **reliability**: continuity of correct service.
- **safety**: absence of catastrophic consequences on the user(s) and the environment.
- **integrity**: absence of improper system alternations.
- **maintainability**: ability to undergo modifications and repairs.
- **confidentiality**: ability to keep information secret from unauthorized users.

In [11], the authors investigated the dependability of the Internet of Things (IoT). They considered the list defined above and added two new attributes related to dependability which are scalability and privacy. Scalability is defined as the ability of the system to be extended by further components while privacy is defined as the ability of someone to (i) assess his personal privacy risks, (ii) take appropriate action to protect his privacy, and (iii) to be assured that it is enforced beyond his immediate control sphere [63].

I consider the previous list of dependability attributes and I extend it by adding **security** as showed in figure 1.5. Security is a highly significant feature for dependability when addressing cyber-physical systems. As discussed in section 1.2.4.3, networking is an important feature for CPS. CPS are composed of several component continuously exchanging data. Therefore any security compromise of the CPS can have severe consequences. Moreover, cyber-physical systems suffer from specific vulnerabilities which do not affect classical control systems, and for which appropriate detection and identification techniques need to be developed. For instance, the reliance on communication networks and standard communication protocols to transmit measurements and control packets increases the possibility of intentional and worst case attacks against physical plants. On the other

hand, information security methods, such as authentication, access control, and message integrity, appear inadequate for a satisfactory protection of CPS. Indeed, these security methods do not exploit the compatibility of the measurements with the underlying physical process or the control mechanism, and they are therefore ineffective against insider attacks targeting the physical dynamics [50].

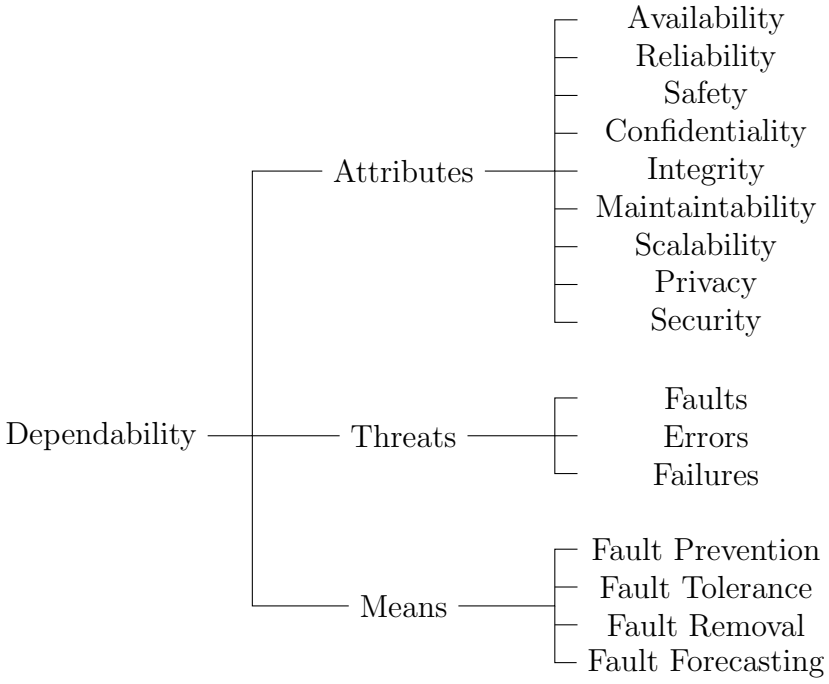


Figure 1.5: Dependability tree (adapted from [10] and [11])

The main objective of dependability is to reduce as much as possible system failures and to ensure that its attributes are effective. In other words, dependability ensures that the system delivers to the maximum the services that are normally provided, without deviating from their respective objectives. However, it is possible that errors could disrupt the proper functioning of the system and cause its failure. A system may fail either because it does not comply with the specification, or because the specification did not adequately describe its function. The threats to dependability are threefold: faults, errors and failures.

In any computing system, the root cause of a malfunction is called **fault**. A fault is active if it produces an error; otherwise it is dormant. According to [62], it is possible to classify faults that may affect the system during its life cycle

according to eight basic viewpoints which are:

- conception phase: development faults or operational faults.
- system boundaries: internal or external faults.
- phenomenological cause: natural or human-made faults.
- dimension: hardware of software faults.
- objective: malicious or non-malicious faults.
- intent: deliberate or non deliberate faults.
- capability: accidental or incompetence faults.
- persistance: permanent or transient faults.

An **error** is the part of the system state that may cause a subsequent failure. Depending on the nature of the system, an error may be detected and corrected before it manifests as a failure. If a fault is activated and no measures are taken to correct the subsequent error, the system may deviate from its specified and intended behavior. This lead to what is called **failure**. A failure is an event that occurs when the delivered services deviate from correct and expected service. The relation between, fault, error and failure is illustrated in figure 1.6.

Figure 1.6: Fault-error-failure relation

The way today's dependable systems are designed is mainly motivated by four different techniques [64]:

- **Fault prevention:** to prevent the occurrence or introduction of faults, including different techniques from system engineering and good practices from design both hardware and software.

- **Fault tolerance:** to avoid service failures in the presence of faults using different techniques such as redundancy, error detection, etc.
- **Fault removal:** to reduce the number of severity of faults mainly using validation and verification techniques.
- **Fault forecasting:** to estimate the present number, the future incidence, and the likely consequences of faults. This includes risk analysis methods.

1.3.3 Limits of existing safety approaches for CPS

Integration of physical processes and computing is not new. The term "embedded system" has been used for quite some time to describe engineered systems that combine physical processes with computing. An embedded system is a computational system embedded in physical system. Any cyber-physical system contains and embedded system. The main distinction is that the term embedded system reflects a primary focus on the computational component. The CPS view emphasizes the importance of taking into account the physical context of the computational system which is often necessary to design, test and verify the functionality that is being developed [65].

Historically, embedded systems (especially critical ones) have been based on (variations of) the V-Schema. A V-Schema includes the following phases: requirement, design, development (or implementation), integration and validation. The V-Schema has been introduced by the safety standard IEC 61508 [66] to deal with software and system dependability. This standard constitutes an umbrella for safety-related domain standards, such as nuclear [67–69], railway [61, 70, 71] and more recently automotive [72] application domains.

Among the main methodologies that have had a crescendo in their success in the scientific and industrial community for the safety analysis of embedded systems, I recall the following ones:

- **Correction-by-construction** was first introduced to address correction for code [73], and then extended to system's components [74]. At system level, correction by construction means a design of system's components such that it is possible to automatically generate a code, which is correct with respect to

a given specification. Correction by construction inherently involved *composability* and *compositionality* [74], i.e. the ability for a component to preserve properties during its integration in another system, and to extend properties from a set of components to the system which includes them.

- **(Semi)-Automatic generation of code** is an issue, strictly related to correction-by-construction approach. It allows engineers to work with an abstraction level higher than code. It is expected to better manage complexity and therefore reduce human errors.

- **Modeling** is more and more deployed on industries. Among the most adopted standards is UML and its profiles (i.e. SysML) promoted by the OMG, AADL [75] and Simulink models.

- **Modular pre-certification** allows engineers to anticipate and structured the argumentation needed to the certification process. The underlying idea is to decrease the (high) cost linked to the certification of the whole system whenever a single component is modified. GSN (Goal Structural Notation [76]) is a standard which aims to improve the certification process. GSN exploits graphical notations and models (1) by specifying safety objectives of a system and (2) structuring the strategy in blocks to achieve the safety objectives.

1.3.4 Resilience

1.3.4.1 Background and definition

Resilience (from the Latin etymology resilire, to rebound) is literally the act or action of springing back. Historically, the notion of resilience have been elaborated in many domains before being introduced to system engineering. [77] investigated the resilience and stability of ecological systems and defined resilience as “moving from a stability domain to another one under the influence of disturbances”. [78] used resilience in child psychology and psychiatry and referred to resilience as “living and developing successfully when facing adversity”. The authors of [79] elaborated resilience in business and referred to it as “the capacity to reinvent a

business model before circumstances force it". In 2006, [6] introduced used the term resilience in industrial safety and defined it as "anticipating risk changes before damage occurrence".

1.3.4.2 From dependability to resilience

A total change of scale is needed when moving from embedded systems to ACPS with complex information infrastructure and myriads of components communicating together and continuously exchanging data. With such systems, what is at stake is maintain dependability, i.e. the ability to deliver service that can justifiably be trusted in spite of continuous changes [62], as discussed in section 1.3.2.

The author of [80] considers that the notion of dependability is in the process of evolving to resilience when it is addressed to complex systems such as CPS. He defines resilience as "the persistence of service delivery that can be justifiably be trusted, when facing changes". This definition is built on the initial definition of dependability, which emphasizes justifiably trusted service. A shorthand definition of resilience would be then "the persistence of dependability when facing changes" [80].

1.3.4.3 Resilience engineering

"A system cannot be resilient, but a system can have a potential for resilient performance" [81]. Dr Erik Hollnagel has an innovative vision of system resilience. According to him, a system is said to perform in a manner that is resilient when it sustains required operations under both expected and unexpected conditions by adjusting its functioning prior do, during, or following events. The main idea that Dr Hollnagel proposes is to change the classical safety analysis process that focuses mainly on reducing the number of adverse outcomes by taking into account the success stories that tend to become invisible and insignificant, because they are considered as normal, i.e. as planned. Even if the idea is not new, and has been the focus of several pieces of research and PhD theses on numerous topics (e.g. safety, situation awareness, sense-making, resilience, feedback of experience, etc.).

A system is traditionally considered to be safe if the number of adverse outcomes is acceptably low. Such outcomes are typically errors, faults and failures

as discussed in section 1.3.2. The level of safety corresponds to the number of such outcomes, and the common interpretation is that a higher level of safety corresponds to a lower number of adverse outcomes. One example of that is the definition of safety by the International Civil Aviation Organization as: "the state in which the risk of harm to persons or of property damage is reduced to, and maintained at or below, an acceptable level through a continuing process of hazard identification and risk management" [82].

However, Dr Hollnagel's vision of safety goes beyond reducing the number of adverse events. According to him, Resilience Engineering (RE) defines safety as the ability to succeed under varying conditions [81]. This definition encompasses the traditional meaning of safety, since the ability to succeed under varying conditions will lead to fewer adverse outcomes. In order to distinguish the two definitions, they have been called Safety-I and Safety-II, respectively [83]. Where the focus of the Safety-I definition is on protection and prevention against harmful events (*protective safety*), the focus of the Safety-II definition is more broadly on the system's ability to function in a way that produces acceptable outcomes (*productive safety*). Resilience engineering is about what a system needs for its continued existence and growth, hence addresses both safety and core business processes such as productivity, quality and effectiveness [81]. This vision revolutionize completely how safety is understood or defined, how it is measured, and how it is managed.

1.4 Formalization of CPS

The previous sections have introduced our research context related to two domains that are cyber-physical systems and safety. The safety analysis of CPS requires the establishment of cyber-physical system architecture in which the services of CPS can be guaranteed by a small subset of modules and their interactions; the design of this subset will have to be **formally** specified and verified. The assumptions made about the physical environment should be fully tested, and furthermore, there is a need to develop advanced and integrated static analysis and testing technologies to ensure that 1) the software code is compliant with the design, and that 2) the assumptions regarding external environment are sound. The verification and validation of a CPS is not a one-time event; it should be life cycle process

that produces an explicit body of evidence for the certification of safety critical services [84].

In order to analyze CPS safety and resilience, it is important to represent these systems and to verify their safety features. The following section is dedicated to existing representation and verification methods for CPS.

1.4.1 Representation of cyber-physical systems

The most important representation methods for CPS are discussed below:

1.4.1.1 State-based modeling

State-based models are mathematical constructs that represent the behavior of the CPS in term of discrete modes of controller software and continuous state variables of physical system. In each discrete mode a certain control decision is evaluated based on the values of certain continuous state variables in CPS. The variation in state variables is evaluated using differential equations which represent the behavior of CPS [85]. Hybrid automata [86] are popularly used to represent the discrete computing models and continuous variables in a single mathematical construct. Hybrid automaton also enables transitions between discrete modes, which are decided associated guard conditions on state variables.

There are several variants of hybrid automata. The most commonly used are timed automata [87] and linear hybrid automata. Timed automata are a subclass of hybrid automata where continuous variables are clocks, that is, continuous variables that have constant slopes equal to 1 (counting time), values of clocks are compared to constants, and the only updates allowed are resets to 0. A linear hybrid automata assumes that the dynamic equations can only be of the form of linear first-order differential equations. The models provide limited support to represent non-linear and spatio-temporal nature of aggregate effects in cyber-physical systems [85].

1.4.1.2 Multiagent representation

Another way of representing networked CPS is by using the concept of multi-agent systems. A Multi-Agent System (MAS) is a system that contains a set of

agents that interact via communications protocols and are able to act on their environment. Different agents have different spheres of influence, mainly because of their control (or at least an influence) on different parts of the environment [88]. The collaboration between multiple agents resulting in group fits nicely with the concept of CPS complexity, which is also the result of communication between multiple agents.

From the above definition, it is obvious that both MAS and CPS operate in a distributed heterogeneous environment. The major component integrated into the CPS includes observation, communication and control aspects [89]. This integration can easily be modelled in a multi-agent based system where agents can have the ability to observe any changes in their environment, act (send control commands) and also to exchange data with another agent via communication [90].

1.4.2 Verification of cyber-physical systems

There has been substantial use of formal methods within the context of system design within the literature. Formal methods are defined within the literature broadly as "mathematical techniques, often supported by tools, for developing software and hardware systems" [91]. Some of the most important CPS verification techniques are discussed below:

1.4.2.1 Model-checking

To verify the correctness of CPS with aggregate effects, model checking on CPS properties (specified using temporal logic formulas or first-order logic formulas) is performed [92]. Model-checking checks if the CPS model satisfies the specified property [85]. Semantically, if M is the CPS model and ϕ is the property, model-checking methods verifies whether M entails ϕ. Model-checking for CPS uses reachability analysis technique to see if the specified property holds for all system reachable states. Such exhaustive exploration of state space can be computationally intensive and time consuming.

Since its development in the early 1980's, model checking has been applied to a large number of problems, such as complex sequential circuit designs and communication protocols. Model checking overcomes a number of problems that other

approaches based on simulation, testing, and deductive reasoning suffer from. To mention a few, approaches based on testing are not complete, and deductive reasoning using theorem provers is generally not fully automated since it has much higher complexity [93]. On the other hand, model checkers are 'push-button' software tools, they do not require any proofs, and they can provide diagnostic counterexamples when a universally path-quantified specification is found to be false. Thanks to these and other features, model checkers have become very popular for (hardware) verification, and are also often used for debugging purposes.

A model checker is usually composed of three main parts:

1. a property specification language based on a temporal logic.

2. a model specification language which is a formal notation for encoding the system to be verified as finite-state transition system

3. a verification procedure which is an intelligent exhaustive search of the model state space that determines whether the specification is satisfied or not. In the latter case, the procedure provides a counterexample path exhibiting the violation of the specification.

Model-checking is a powerful framework for verifying specifications on finite-state systems. One of the main advantages of model-checking is that it is fully automated. No expert is required in order to check whether a given finite-state model conforms to a given set of system specifications. Model-checking also works with partial specifications, which are often troublesome for techniques based on theorem proving. When a property specification does not hold, a model checker can provide a counterexample (an initial state and a set of transitions) that reflects an actual execution leading to an error state. This is the reason why tools based on model-checking are popular for debugging.

However, the main drawback for model-checking is the state-explosion problem. If the number of states is too large, then the complexity of the verification procedure may render the technique unusable [93].

1.4.2.2 Theorem proving

Theorem proving is widely used for CPS verification by providing mathematical reasoning on the correctness of system properties [94]. Unlike model-checking, theorem proving requires less time as it reasons about the state space using system constraints only, not on all states on state space. However, fully automated techniques are less popular for theorem proving as automatically generated proofs can be long and hard to understand [85].

1.4.2.3 Simulation

Simulations of various network components and their interaction with the physical world are widely used for designing wireless networks. Thus, they can be readily used to consider aggregate effects in CPS. In [95], the authors developed the Wireless Cyber-Physical Simulator (WCPS), an integrated environment that combines realistic simulations of both wireless sensor networks and structures. WCPS focuses on simulating wireless civil infrastructural control systems and the effects of network delays and data loss on control. In [96], the authors evaluate Jitterbug and Truetime which are Matlab-based simulators used for simulating the effects of network performance on continuous control systems. However, a common disadvantage of these simulators is that they do not consider events from continuous systems. These events can change parameters of the control system and hence change the nature of continuous dynamics that govern the system variables. Further, they do not consider any form of time refinement to accurately estimate events timing and hence take the fixed time step approach towards simulation which can result in approximate estimation of aggregate effects [85].

1.4.2.4 Symbolic execution

Another way of verifying CPS safety is to analyze the cyber part which covers all the software involved in the CPS. Symbolic execution is a technique to analyze CPS code and automatically generate test case inputs that might cause errors in the software. In this technique, the variables in a program are represented using symbols and the steps of the program are executed to track the values of the variable in terms of expressions on the symbols. For any branch statement, the

symbolic equation is compared with a threshold to generate input data that can result in changes in program sequence. This methodology can be used to generate test cases for CPS software. Such test cases can then be simulated using physical system simulators to analyze the effects of different inputs on the CPS [85].

1.5 Key factors for CPS safety

In order to analyze CPS safety, it is important to identify the key factors that are involved. These factors will need a special focus during the verification process of CPS. This section is dedicated to the identification of these factors. For this purpose, I start by specifying a CPS use case which is drones. Then, I focus on the features that are relevant to the safety analysis of CPS.

1.5.1 Drones as a use case

As discussed in section 1.2.5.3, Unmanned Aerial Vehicles (commonly called drones) are an interesting application of CPS. They are a representative use case as they involve a cyber part covering all the flying-related computation and a physical part involving a set of sensors and actuators. The networking part is also present in the drone use case. The control part is done remotely either by a human operator on the ground or by a Ground Control Station (GSC). The GCS can be considered a part of the whole system. The whole system including the flying vehicle and the GCS is called Unmanned Aerial System (UAS). In [97], the author projects that global civil drones fleet will increase 12.6 % a year, raising from 4.9 billion USD in 2019 to an expected 88.3 billion USD in 2028. Drones have been introduced for military applications, but the ample availability of affordable drones is leading to large amounts of drones being sold for civilian applications, including the industrial ones, especially when drones are being equipped with high quality cameras and many other sensors which make them adaptable to a variable set of civilian applications. The agriculture domain provides us a capital example of that innovation where drones are mainly exploited for the digital territory supervising of the status of an orchard or a vineyard. Another civil application is humanitarian interventions where drones are being used to transport blood samples or providing

supplies to highly contagious areas.

The drone number entering the airspace is increasing and is leading to real concerns about safety and security issues; more and more small incidents are occurring making safety issues for civil UAV a key feature in order to obviate serious incidents.

Because they are unmanned, UAV are less well maintained and subsequently less reliable than manned aircrafts. UAS suffer accident rates multiple times higher than manned aircrafts [98, 99]. It is an expected result since the civilian drones are piloted by amateurs with no particular requirements.

The traditional users of the airspace are also concerned because they are expecting new aircrafts and other flying objects entering the airspace, without traditional safety measures and procedures being followed. Small drones cannot reach a high altitude and yet their collision with manned aircraft is still probable. Even below 500 feet, there is still a lot of air traffic especially in the airspace next to airports, where the landing and departing aircrafts are due.

All these concerns lead to one conclusion: the importance of ensuring safety for UAS in civilian applications.

1.5.2 Key factors for drone safety

1.5.2.1 Drone airworthiness

For civilian applications, a flight is considered safe if, for a given mission, the drone is capable of accomplishing its goal without any damage or accident. The term "airworthiness" has been first introduced for manned aircraft to refer to an aircraft's suitability for a safe flight [100], and is now commonly used for drones. In order to ensure drones airworthiness, a set of given factors have to be satisfied. I strategically capitalize on the work done by Filippo Del Florio [101] in the field of manned aircrafts and I adapt it to the UAS. Del Florio identified three main conventional flight safety factors which are the pilot, the environment and the machine [101]. As shown in Figure 1.7 , I take into account this identification and I consider three major **flight safety factors for UAS** which are: the human, the drone and the environment. These factors are strongly linked in order to ensure drone's safety. The failure of a single link is sufficient for and accident to occur.

An error made by the pilot can lead to the crash of the best aircraft and the best pilot would not be able to compensate for a severe aircraft failure. Accidents are often caused by a combination of multiple factors. Nevertheless, the accident always begin with the failure of one of the above-mentioned factors.

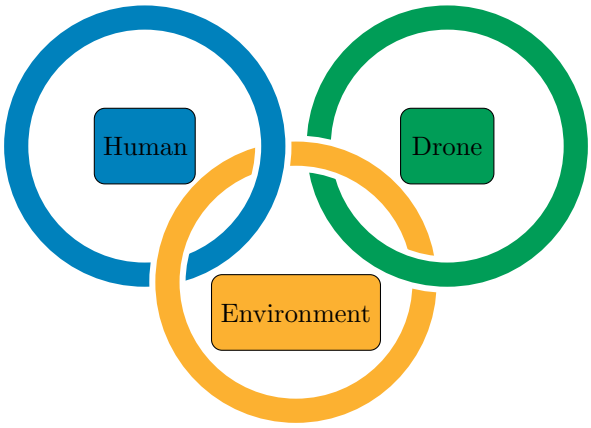


Figure 1.7: Flight safety factors for UAS

1.5.2.2 Human factor

The *human factor* regroups all the human operators included for the drone flight. Some missions are complex and require the coordination between multiple human operators in order to ensure the accomplishment of the given mission in safe conditions. The main human operators involved for drone operations are: the aircraft operator(s), the payload operator and the control station technician [102]. The aircraft operator is the person who operates the aircraft from the engine start to shut down. The aircraft operator is the main part of the UAS flight crew. He plans the flight in advance, and executes the flight operation. He also provides coordination between the crew members and gives the final decisions related to the aircraft operation. The payload operator operates the payload when the aircraft is on the ground, at the time of takeoff and landing, and in all phases of flight operation. The control station technician performs the pre-flight and post-flight checks, executes the required procedures of control station to maintain the aircraft's functionality during the flight. The significant difference between UAS and manned

ones is the skill of the human operator. In this regard, the pilots of manned aircrafts need to have specific flying licences and satisfy certain requirements in order to be authorized to operate the aircrafts. On the other hand, civilian drones are operated by people without any specific background and licences. This difference of skill and experience is vitally important for safety reasons because inexperienced pilots can lead to serious accidents. In order to solve this issue, the European Aviation Safety Agency (EASA) is leading the European initiative by providing a regulatory framework for drone operations [103]. This highlights the importance of the human operator and the need to take it into consideration during the safety analysis of the UAS.

1.5.2.3 Environment

The *environment* covers all the external elements that can eventually impact the flight. This includes the meteorological conditions, the communication between the drone and the remote control and also physical obstacles. There are two different types of environments: static and dynamic. An environment is considered static when all the external elements are determined before the flight and their respective behavior is predictable and controlled, no further elements can be added during the flight. This type of environment is mainly used for tests and simulations to highlight predefined aspects of the UAS. Unlike static environments, the dynamic ones are more realistic. Unexpected events may occur during the flight and may impact the drone's behavior and the flight's safety making the safety analysis even more complex. Representing the external elements, especially when they are unpredictable, is a challenge. Simulations that includes dynamic environments are more complex and more accurate because they are closer to the real world.

The environment is of a capital importance when addressing the safety for drones and CPS in general because it has a significant impact on the system's behaviour. Taking into consideration the environment during the safety analysis of these systems is essential.

1.5.2.4 Drone

The *drone* is the unmanned aerial vehicle. Drones can be equipped with information gathering tools such as cameras and sensors which allows it to extract data from its environment. This feature gives the UAS a great flexibility making it able to adapt to a wide range of applications. This equipment that can be associated with a drone embrace from high resolution cameras to radiation detectors, from night vision cameras to air sampling devices and from heat sensors to mobile-phone jammers.

In order to ensure the drone airworthiness, some safety constraints must be maintained. In order to determine these constraints, I capitalize on the work done by Clarke [100], and I extend the list that he established. The key attributes that enables the drone survival are [7]:

- Awareness of the drone's location within the operational space and also its direction and acceleration.
- Sensors and/or remote data-feeds that enables maintenance of the awareness of location, attitude and movement in a sufficiently timed manner.
- Sufficient set of controls over the drone's altitude, direction and acceleration, to enable flight to be sustained under a wide variety of atmospheric conditions.
- Secure and stable communications between the drone and the remote control, to provide a sufficiently rapid response to the controls (manoeuvrability) and to ensure that no external actor can interfere into this connection causing the loss of data, crashing the drone or even taking control of the flying system.
- Failure containment including the sensors failure (wrong data or data loss), weak signal or total loss of communication. This also includes the failure of the hardware parts of the drone. The failure containment can be resolved by using sensors redundancy and the validation of the data provided by the sensors (data validation). Some researchers have elaborated a solution for a quad-rotor to maintain the flight and avoid the crash despite having lost one or many propellers [104].

- Sufficient power to maintain movement, to implement the controls, and to operate sensors and data-feeds, for the duration of the flight.
- Collision avoidance and threat detection. This attribute can be realized by the analysis of the data flow provided by the drone's sensors and cameras. The analysis must be done in real-time in order to avoid accident before their occurrence.
- Sufficient physical robustness to withstand threatening events, such as wind-shear, turbulence, lightning and bird-strike.

It is important to note that this section highlights the importance of considering the system itself (the drone is this case) when addressing safety issues. The above-mentioned list of constraints is not exclusive to drones, it covers other ACPS such as autonomous vehicles.

1.5.3 Autonomy levels and safety

As discussed in section 1.2.4.1, autonomy is an important feature of CPS. The autonomy is defined as the quality of being independent and self-governing. An autonomous system is capable of sensing, analyzing, communicating, planning, decision-making and acting during the mission (online) as assigned by it was programmed by its human operator (offline). An Unmanned Aerial System is usually designed to be directly operated by humans. In this regard, the autonomy in UAS is relative to a given mission. A UAS is fully autonomous if it accomplishes its assigned mission successfully without any intervention from human or any other external system while adapting to operational and environmental conditions [105]. The implementation of autonomy is an essential step toward the road-map for the development of unmanned aerial systems. This capability would not only allow performing missions with better efficiency and effectiveness, but also with improved system safety. [106] was one of the pioneers who introduced the autonomy and human-computer interactions. He proposed a 10-level scale of degrees of autonomy based on the entity responsible of decision-making (human or computer) and how to execute those decisions. Autonomy evaluation for unmanned systems is generally associated with measuring its level of autonomy.

The NASA developed an autonomy assessment tool based of the OODA (Observe, Orient, Decide and Act) loop and uses an 8-level scale to measure the autonomy of each OODA category [107]. Concerning the Unmanned Aerial Systems, the US Air Force Research Laboratory (AFRL) presented the results of a research study on how to measure the autonomy level of an unmanned aerial vehicle (2002) [108]. The result of this study is summed in the Autonomy Control Level (ACL) chart where 11 autonomy levels have been introduced. The autonomy level is determined using OODA concept: perception and situational awareness (observe), analysis and coordination (orient), decision making (decide) and capability (act). A more generic framework has been described later for defining Autonomy Levels For Unmanned Systems (ALFUS) and later renamed Contextual Autonomous Capability (CAC) [24]. In this framework, the autonomy level is determined by measuring various metrics of three aspects (or axes) which are human independence (HI), mission complexity (MC) and environmental complexity (EC) (Figure 1.1 in Section 1.2.4.1).

1.5.4 Integrating manned and unmanned CPS

How to integrate manned and unmanned CPS ? What would happen if a collision occur involving a manned and an unmanned CPS (such as drones or autonomous vehicles) ? These questions make sense with the rise of concepts such as Smart Cities where manned and unmanned CPS are supposed to coexist to share the same infrastructures. In order to answer these questions, I consider the drone use case and I reason about how to integrate these unmanned systems with manned ones in a safe way.

Integrating manned and manned aircrafts is challenging, especially when only few and unconfirmed midair collisions have been seen so far, and only few tests have been conducted to show the results of these collisions. It is possible to consider the drone/plane collision similar to bird strikes, a well documented and studied topic [109]. But this is not totally exact since the drone components are metallic and cause different damages to the planes engines than bird bones. A team of engineers at the Crash-worthiness for Aerospace Structures and Hybrids (CRASH) Lab at Virginia Tech [110,111] has delved into this question in an effort

to understand how the consequences of a drone strike might be different depending on where the strike took place on the aircraft. According to the CRASH Lab team: "once an engine has digested a drone, it will suffer from a minimum of operational stability issues, to a maximum of thrust loss due to catastrophic failure".

Many efforts are being conducted in order to provide eventual solutions that might reduce or prevent future incidents involving drones in the airspace. I strategically capitalize on the work done in [109] to list the following solutions:

- **Geo-fencing:** Geo-fencing is defined as the use of software to limit the areas where UAS can fly. DJI, the worlds leading manufacturer of small drones for civilian use, has installed geo-fencing software in its unmanned aircrafts. This system restricts user's ability to fly within five miles of an airport or within certain restricted airspaces such as Washington, D.C [112]. Geo-fencing allows manufacturers to govern where their products fly, proving help to people with good intentions but who don't necessarily read and follow instructions. Nevertheless, it is relatively easy to defeat. So this solution is useful with cooperative users, but a non-cooperative user who doesn't want to use the geo-fencing system can get around it. With some software programming skills, it is possible to modify the code executed on the drone and to fly over restricted areas.

- **Sense-and-avoid:** Sense-and-avoid systems allows unmanned aircrafts to autonomously detect a potential collision with another aircraft and take evasive action, just as a human pilot would. In [113], the authors develop control designs and vehicle models to analyze collision avoidance between quadrotors and helicopters.

- **Traffic management:** One way of keeping drones and manned aircraft away from each other is to implement an air traffic control system similar to the one currently in use for manned aircrafts. NASA's Unmanned Aircraft Systems Traffic Management initiative is looking to build a management system that would provide drone operations with "airspace design, corridors, dynamic geo-fencing, severe weather and wind avoidance, congestion management, terrain avoidance, route planning and re-routing, separation management, sequencing and spacing, and contingency management" [114].

- **Education:** There is a high level of consensus among experts that lack of awareness of airspace rules and guidelines on the part of drone operators is a major contributing factor to incidents involving drones in the national airspace. Therefore, educational campaigns have been identified as a potential means of reducing the rate of such incidents [109].

It is important to mention that the above list of solutions is not exhaustive, and is not restricted to unmanned aircrafts. The proposed solutions remain valid for other CPS such as autonomous vehicles.

1.5.5 Communication technologies reliability

As mentioned in section 1.2.4.3, networking is an important feature of CPS. Ensuring a secure communication between the multiple parts of the CPS is essential in order to guarantee its safety. In this section, I consider the same CPS use case as the previous sections which is drones, and I highlight how important is communication technologies reliability for the safety analysis of these systems.

The control/command system embedded in the UAS usually pre-program a safety procedure that the UAV should follow in case of lost connection with the GCS such as flying back to the departure location or flying on a static shape until the connection with the GCS is recovered, and unfortunately this loss of connectivity might be a source of different incidents causing the unsafe use of the UAS. Most of the civilian UAS are connected to the GCS via wireless connections similar to WiFi. This requires a clear and interference-free flying coverage for the UAS to be authorized to fly. In the context of civilian UAS applications, it is important to evaluate the communication's reliability in order to predict the behaviour of the system when it looses its connection. The energy consumption is also a constraint that needs to be taken into account. Highly reliable communications are often very energy consuming. So a balance must be struck between communication reliability and energy consumption.

1.6 Basic notions and definitions

In this section, I first recall the basic notions and definitions about contract-based design and networks of timed automata, which are essential to apprehend the proposed methodology in the next chapter (Chapter 2). I also discuss the state of the art of these techniques and I discuss howthey can be applied to the analysis of resilience in ACPS.

1.6.1 Contract-Based Design

As mentioned in [115], it is difficult to write a comprehensive bibliography on the general aspects of Contract-Based Design (CBD). The topic is multi-faceted and has been addressed by several communities: software engineering, language design, system engineering, and formal methods in a broad sense. CBD was used to deal with platform [1], and was successfully applied at system level (e.g. [116]). CBD is a methodology that allows engineering to rigorously specify component's interfaces [117].

The framework of contracts developed in the area of Software Engineering have proved useful paradigms for component-based software system development. For cyber-physical systems, *model-based development* (MBD) is generally considered as a key enabler due to its capabilities to support early validations and virtual system integration. MBD-inspired design languages and tools include SysMl [118] or AADL [119] for system level modeling, Modelica [120] for physical system modelling, Matlab-Simulink [121] for control-law design, and Scade [122] for detailed software design. UML-related standardization efforts also include the MARTE UML [123] profile for real-time systems.

In order to understand CBD, it is important to introduce the notion of component. A *component* is a hierarchical entity that represents a unit of design. Components are connected together by sharing and agreeing on the values of certain ports and variables.

A contract $\mathcal{C}$ for a component $\mathcal{M}$ is a pair of assertions (A, G), called the *assumptions* and the *guarantees*, each representing a specific set of behaviors over the component variables [124]. An implementation M satisfies an assertion B when-

ever M and B are defined over the same state of variables and all the behaviors of M satisfy the assertion, i.e when $M \subseteq B$.

An implementation of a component satisfies a contract whenever it satisfies its guarantee, subject to the assumption(s). Formally, $M \cap A \subseteq G$, where M and $\mathcal{C}$ have the same variables. Such a *satisfaction* relation is denoted by writing $M \vDash \mathcal{C}$. An implementation E is a legal *environment* for $\mathcal{C}$, i.e $E \vDash_E \mathcal{C}$ whenever $E \subseteq A$. Two contracts $\mathcal{C}$ and $\mathcal{C}'$ with identical variables, identical assumptions, and such that $G' \cup \neg A = G \cup \neg A$, where $\neg A$ is the complement of A possess identical sets of environments and implementations. Such two contracts are then equivalent. In particular, any contract $\mathcal{C} = (A, G)$ is equivalent to a contract in *saturated form* (A, G'), obtained by taking $G' = G \cup \neg A$. A contract is consistent when the set of implementation satisfying it is not empty, i.e. it is feasible to develop implementations for it [124].

Contracts often appear in the form of an Interface Theory. Interfaces have been the subject of considerable literature. [117] introduced *Interface Automata*, where interfaces are seen as games between the component and its environment. Since then, Interface Automata have often been considered as *the* theory of reference regarding interfaces [115].

A widely accepted approach to deal with complexity of systems in several domains is to structure product development processes along variations of the V diagram. Its characteristic V-shape splits the product development process into two different phases: *design* and *integration*. Usually, safety analysis methodologies intervene in the later phases of the V diagram. But it is very costly (both in time and money) to correct a bug detected in a later phase of the V diagram. So it would be very efficient to include the safety from the early phases of the V diagram. This is one of the main advantages of the contract-based design since it is integrated since the phases of requirement expressing and design.

An extensive trace-based theory of Assume/Guarantee reasoning in the form of A/G-contracts has been proposed in the SPEEDS [125] project with explicit handling of multiple-viewpoint contracts. By explicitly relying on the notions of Assumptions and Guarantees, A/G-contracts are intuitive, which makes them appealing for engineers [115].

The wealth of results in temporal logic and model checking can provide a sub-

stantial basis for requirement analysis for *discrete time (discrete-event) discrete-state* system abstractions [126]. Both assumptions A and guarantees G of a contract $\mathcal{C}$ can be specified as temporal logic formulas [124]. In this case, a component $\mathcal{M}$ satisfies the contract $\mathcal{C}$ if it satisfies the logical implication $A \rightarrow G$, while it is a legal environment for $\mathcal{C}$ if it satisfies the formula A. Contract satisfaction can thus be reduced to two specific instances of model checking [127]. Composition and conjunction of contracts $\mathcal{C}_1$ and $\mathcal{C}_2$ can be represented by appropriate Boolean combinations of the formulas A_1, A_2, G_1 and G_2. *Refinement* is an instance of *validity checking*. Checking that $\mathcal{C}_1$ refines $\mathcal{C}_2$ can be translated into checking that $A_1 \rightarrow A_2$ and $G_2 \rightarrow G_1$ are valid formulas. Contract *compatibility* and *consistency* checking are, instead, less immediate, since they may either translate into checking *satisfiability* or *realizability* of formulas, depending on the specific temporal logic used and the semantics adopted for implementation and environments [127].

1.6.1.1 Contribution to the State of Art

In section(ref), I discussed how safety notion is evolving by introduction the concepts of Safety I and Safety II. So far, CBD has been used to represent risk analysis (Safety I). My contribution to the state of the art is the application of CBD to Safety II (i.e. the wished behaviour) and to resilience more in general.

1.6.2 Networks of timed automata

No better than the UPPAAL tutorial [128] to recall the common definition of timed automata: finite-state machines extended by synchronous-evolving *clocks* used to abstract and reason about the real-time behaviors of systems. UPPAAL extends them by bounded discrete integer variables. In this section, I slightly revisit the succinct and intuitive definitions of the formalism and its semantics given in [128] for more precision.

The universal set of discrete variables is denoted by $\mathbf{X}$. Given a set $X \subset \mathbf{X}$, I define by $\mathbb{T}[x]$ the type (possible values) of $x \in X$, written $x : \mathbb{T}[x]$ for short. $\mathbb{T}[\![X]\!]$ is the type of X (the union set of cartesian products $\prod_{x_i \in X} \mathbb{T}[\sigma(x_i)]$ for all permutations $\sigma : X \rightarrow X$). I write $\mathbb{T}[\![X_1, ..., X_n]\!]$ for $\mathbb{T}[\![\bigcup_{1 \leq i \leq n}(X_i)]\!]$ with $X_i \subset \mathbf{X}$. I denote by $\mathbf{C}$ the universal set of *clocks*. I have $\mathbb{T}[c] = \mathbb{R}^+$ for any $c \in \mathbf{C}$, and

$\mathbb{T}[C] = (\mathbb{R}^+)^n$ for any $C \subset \mathbf{C}$ with a cardinality $|C| = n$. The left shifted (or the *next*) version of $x \in X$ with a one logic step is denoted by $x' : \mathbb{T}[x]$ *e.g.,* let us consider two logic successive states s_1 and s_2, if $x = x_1$ at s_1 and $x = x_2$ at s_2, then $x' = x_2$ at s_1 and so on. I generalize the notation for sets, X' is the set of next versions x' of all $x \in X$.

Conditions are predicates of the First-Order Logic. Given $X = \{x_1, ..., x_n\} \subset \mathbf{X}$, a *predicate* p on $x \in X$ represents a subset of the possible values of x *i.e., p* is a sub-type of $\mathbb{T}[x]$. A predicate Q on X is then a sub-type of $\mathbb{T}[\![X]\!]$. The *projection* of Q on $Z = \{z_1, ..., z_k\} \subseteq X$ is written $Q[\![Z]\!]$. The syntax of predicates, functions, operators and constants are defined according to variable types under specific theories (equality, linear arithmetic, etc). $Q\langle x/x'\rangle$ is Q by substituting $x \in X$ by its primed version x'. I generalize the notation for sets: $Q\langle\!\langle X/X'\rangle\!\rangle = Q\langle x_1/x'_1\rangle\langle x_2/x'_2\rangle ... \langle x_n/x'_n\rangle$.

Definition 1. *A* timed automaton (TA) *A on $C, X \subset \mathbf{C}, \mathbf{X}$ is a tuple $(\Upsilon, \imath, \Sigma, \Psi, \mathcal{G}, \mathcal{E}, \mathcal{J}, \mathcal{I})$ consisting of:*

- *a set of* locations Υ *with $\imath \in \Upsilon$ is the* initial *state;*
- *a set of* actions Σ*;*
- *a* transition function $\Psi \subseteq \Upsilon \times \Sigma \to \Upsilon$*;*
- *a* guard *function* $\mathcal{G} \subseteq \Psi \to \mathbb{T}[\![C, X]\!]$*;*
- *an* update *function* $\mathcal{E} \subseteq \Psi \times \mathbb{T}[\![C, X]\!] \to \mathbb{T}[\![C', X']\!]$*;*
- *an* initialization *function* $\mathcal{J} \subseteq \{\imath\} \to \mathbb{T}[\![C, X]\!]$*;*
- *an* invariant *function* $\mathcal{I} \subseteq \Upsilon \to \mathbb{T}[\![C, X]\!]$.

For all $c \in (\mathbb{R}^+)^n = \mathbb{T}[\![C]\!]$ where $n = |C|$, I denote by $c \oplus \tau$ the tuple $(c_1 + \tau, ..., c_n + \tau)$. To save space, I write $e_1...e_k$ instead of $(e_1, ..., e_k)$ with $k \in \mathbb{N}^+$. I write $A.K$ for any component $K \in \{\Upsilon, \imath, \Sigma, \Psi, \mathcal{G}, \mathcal{E}, \mathcal{J}, \mathcal{I}\}$ of A.

Definition 2. *The* semantics $\mathfrak{S}(A)$ *of A is a labeled transition system LTS $(S, s_0, \mathcal{R})$ where $S \subset \Upsilon \times \mathbb{R}^{|C|} \times \mathbb{T}[\![X]\!]$ is the set of* states *with $s_0 = (\imath, \mathcal{J}(\imath)[\![C]\!], \mathcal{J}(\imath)[\![X]\!]) \in S$ is* initial*, and $\mathcal{R} \subseteq S \times (\mathbb{R}^+ \cup \Sigma) \to S$ is a* transition function *such that:*

- $\mathcal{R}(lcx,\tau) = l(c\oplus\tau)x$ *if* $(\forall t\in[0,\tau])\, c\oplus t\in\mathcal{I}(l)[\![C]\!]$;
- $\mathcal{R}(lcx,a) = l'c'x'$ *such that*
  - $l' = \Psi(la)$,
  - $c\in\mathcal{I}(l)\wedge\mathcal{G}(la\mapsto l')[\![C]\!]$, $c'\in\mathcal{E}(la\mapsto l',c)[\![C']\!]$,
  - $x\in\mathcal{I}(l)\wedge\mathcal{G}(la\mapsto l')[\![X]\!]$, $x'\in\mathcal{E}(la\mapsto l',x)[\![X']\!]$,
  - $c'\in\mathcal{I}(l')\langle\!\langle C/C'\rangle\!\rangle$, *and* $x'\in\mathcal{I}(l')\langle\!\langle X/X'\rangle\!\rangle$.

In Uppaal's graphical language, a system is modeled by a Network of Timed Automata (NTA) with concurrent behavioral semantics over sets of common clocks and variables $C,X\subset\mathbf{C},\mathbf{X}$, and actions Σ split into two disjoint subsets Σ^{asy} and Σ^{sy} of resp. *asynchronous* and *synchronous* actions. This NTA is written $A_1\|...\|A_n$ (or $A_{1..n}$), and consists of n TAs A_i for $i\in\{1,...,n\}$. I write K_i, $\bar{\Upsilon}$ and $\bar{\imath}$ resp. for $A_i.K$, the product $\prod_i\Upsilon_i$, and the vector $\imath_1...\imath_n$. I define $\mathcal{I}\subseteq\bar{\Upsilon}\to\mathbb{T}[\![C,X]\!]$ and $\mathcal{J}\subseteq\{\bar{\imath}\}\to\mathbb{T}[\![C,X]\!]$ resp. such that $\mathcal{I}(\bar{l})=\bigwedge_i\mathcal{I}_i(l_i)$ with $\bar{l}=l_1...l_n$ and $\mathcal{J}(\bar{\imath})=\bigwedge_i\mathcal{J}_i(\imath_i)$. I write $\bar{l}[l_i/l_i']$ to denote $\bar{l}$ with l_i replaced by l_i'.

Definition 3. *The semantics* $\mathfrak{S}(A_{1..n})$ *of* $A_{1..n}$ *is a LTS* $(S,s_0,\mathcal{R})$ *consisting of a set of states* $S\subset\bar{\Upsilon}\times\mathbb{R}^{|C|}\times\mathbb{T}[\![X]\!]$ *with* $s_0=(\bar{\imath},\mathcal{J}(\bar{\imath})[\![C]\!],\mathcal{J}(\bar{\imath})[\![X]\!])\in S$ *initial, and a transition relation* $\mathcal{R}\subseteq S\times(\mathbb{R}^+\cup\Sigma)\to S$ *defined such that:*

- $\mathcal{R}(\bar{l}cx,\tau)=\bar{l}(c\oplus\tau)x$ *if* $(\forall t\in[0,\tau])\, c\oplus t\in\mathcal{I}(\bar{l})[\![C]\!]$;
- $\mathcal{R}(\bar{l}cx,a) = \bar{l}'c'x'$ *such that*
  - $a\in\Sigma^{\text{asy}}$, $\bar{l}'=\bar{l}[l_i/\Psi_i(l_ia)]$,
  - $c\in\mathcal{I}(l)\wedge\mathcal{G}_i(l_ia\mapsto l_i')[\![C]\!]$, $c'\in\mathcal{E}_i(l_ia\mapsto l_i',c)[\![C']\!]$,
  - $x\in\mathcal{I}(l)\wedge\mathcal{G}_i(l_ia\mapsto l_i')[\![X]\!]$, $x'\in\mathcal{E}_i(l_ia\mapsto l_i',x)[\![X']\!]$,
  - $c'\in\mathcal{I}(\bar{l}')\langle\!\langle C/C'\rangle\!\rangle$, *and* $x'\in\mathcal{I}(\bar{l}')\langle\!\langle X/X'\rangle\!\rangle$;
- $\mathcal{R}(\bar{l}cx,b) = \bar{l}'c'x'$ *such that*
  - $b\in\Sigma^{\text{sy}}$, $\bar{l}'=\bar{l}[l_i/\Psi_i(l_ib)][l_j/\Psi_j(l_jb)]$,
  - $c\in\mathcal{I}(\bar{l})\wedge\mathcal{G}_i(l_ib\mapsto l_i')\wedge\mathcal{G}_j(l_jb\mapsto l_j')[\![C]\!]$,
    $v\in\mathcal{I}(\bar{l})\wedge\mathcal{G}_i(l_ib\mapsto l_i')\wedge\mathcal{G}_j(l_jb\mapsto l_j')[\![X]\!]$,

– $c' \in \mathcal{E}_i(l_i b \mapsto l'_i, c) \wedge \mathcal{E}_j(l_j b \mapsto l'_j, c)[\![C']\!]$,
$x' \in \mathcal{E}_i(l_i b \mapsto l'_i, x) \wedge \mathcal{E}_j(l_j b \mapsto l'_j, x)[\![X']\!]$,
– $c' \in \mathcal{I}(\bar{l'})\langle\!\langle C/C' \rangle\!\rangle$, *and* $x' \in \mathcal{I}(\bar{l'})\langle\!\langle X/X' \rangle\!\rangle$.

From a given state of the resulted LTS, every A_i may 1) fire a transition separately by elapsing time, or by enabling an asynchronous action and applying its individual update on variables and clocks, or 2) synchronize with another TA A_j through transition(s) labeled by a synchronous action a enabled as output (depicted $a!$ in UPPAAL) in one of them and as input (depicted by $a?$ in UPPAAL) in the other such that $a!$ transition update applies before that of $a?$ transition.

The toolbox UPPAAL offers additional design features like time urgency, synchronous and broadcasting (asynchronous) channels, data types (arrays and structures), etc. It has a query language to specify CTL properties for model-checking.

1.7 Conclusion

This first chapter constitutes a state of the art of the research problem of this thesis which is the safety of Cyber-Physical Systems. First, Cyber-Physical Systems and Autonomous Cyber-Physical Systems were defined. The properties of these systems were listed and different application domains were detailed. Second, safety was defined. The challenge of ensuring safety for CPS was highlighted by describing how existing safety analysis methodologies are insufficient to deal with systems as complex as CPS. The concept of resilience was then introduced as a key solution to analyze the safety of such systems. Third, key factors for CPS safety were identified. These factors are essential and must be taken into account during the safety analysis of CPS.

The main goal of this thesis is to propose a methodology for the analysis of resilience of CPS and to investigate the impact of Artificial Intelligence (AI) on the safety of CPS. The next chapter will address the first point which is the proposition of a methodology for the analyis of resilience of CPS.

Chapter 2

A new methodology for analysis of resilience in ACPS

Contents

2.1 Introduction

In Chapter 1, I introduced the research problem of this thesis.

[DANI 1-07-20: the first chapter concerns basic definitions related to CPS and safety. You have to introduce the research problem explicitly (section). You can do it at the begin of Chapter 2 or at the end of Chapter 1]

[DANI 1-07-20: the following paragraph can be in the section research problem] I argued how traditional safety analyses and methods have limits when applied to ACPS, and I promoted the notion of resilience and identified the factors that are relevant to the resilience analysis of ACPS. My analysis shows that the environment is of a major importance. This importance is related, when we deal with ACPS, to the notion of uncertainty [32].

In this chapter, I promote a methodology for the analysis of resilience in ACPS. This chapter constitutes the main contribution to this thesis. First, I specify a use case based on a civilian application of autonomous drones to facilitate the discussion and the analysis (Section 2.2). Second, I introduce the notions of endogenous and exogenous resilience for ACPS, and I detail the proposed methods, techniques and tools for their analysis (Section 2.4). Finally, I address the tool development and I provide a feedback on the experience that I had in ordre to realize the system models and to develop the tools (Section 2.8).

2.2 Use case: Drone rescue system

Unmanned Aerial Vehicles are a very interesting class of CPS and in particular their civilian applications (Section 1.2.5.3). Drones are quite useful for healthcare organizations especially in emergency situations to avoid traffic jams, or when traditional transports are severely restricted, following a natural disaster for example. Common applications include live broadcasts of accidents to early grasp them, and deliver the required medical supplies for wounded people. UAVs are safe enough to transport disease test samples and kits in areas with high contagion. In emergency medicine, the studies have shown that drones are fast to deliver automated external defibrillator to rescue out-of-hospital heart attack victims using geographic information systems. Flying at speeds of up to 97 km/h, the drone can reach

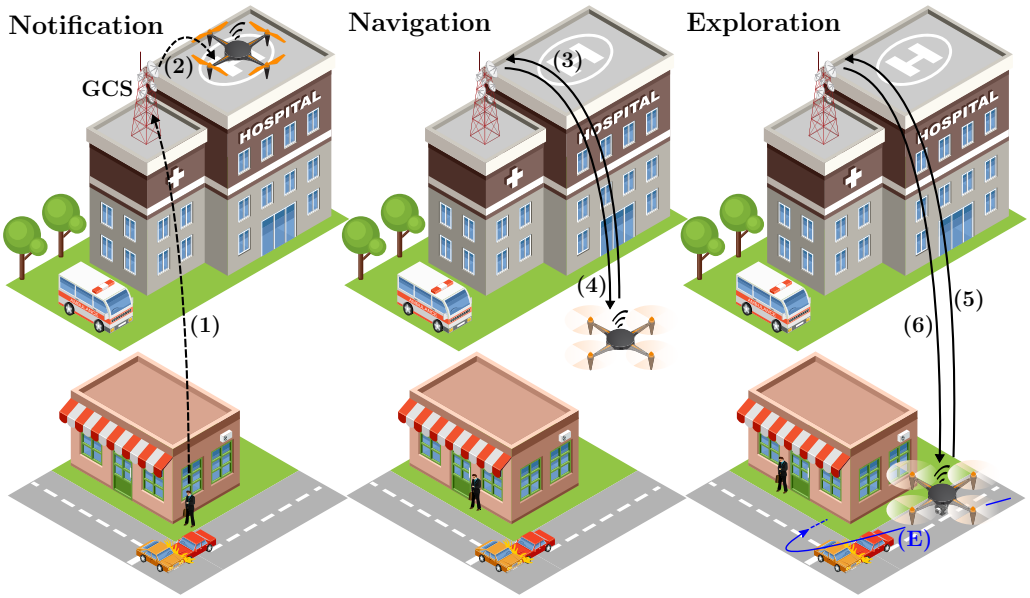


Figure 2.1: Drone/GCS crash exploration scenario. Notification phase: (1) a person on-site sends an alert to the GCS by smartphone; (2) GCS activates the drone; Navigation phase: (3) the drone continuously sends navigation data to GCS; (4) GCS controls the drone when necessary; Exploration phase: (E) while hovering, (5) the drone broadcasts the accident scene to GCS, which in turn (6) may order it to provide medical supply for victims if needed. Steps (3)/(4) and (5)/(6) are repeated with different frequencies (depicted by ⟶).

patients within a radius of 13 km^2 per mn versus 10 mn average for traditional services which increases the chance of survival to 80% [129]. Being connected to live-stream camera fixed on the drone, a Ground Control Station (GCS) instructs the first aid gestures for the patients, and provides the needed medical supply (transported as it happens by the drone).

Nevertheless, the usage of UAVs remains very challenging at the design and verification levels despite the fast progress of their related technologies. I specify the Drone/GCS crash exploration scenario (ref Figure 2.1), its *system requirements* to deal with the crash scenario, and I address safety issues for the software layer of a Drone/GCS-based urban rescue system.

2.2.1 Use Case Scenario

The use case scenario is described in Figure 2.1. By receiving a notification from on-site persons who notice the crash severity (step 1), the GCS activates and sends immediately a drone to the place of the accident (step 2). Once flying, the drone sends continuously its navigation data (including its position composed of the latitude, longitude and altitude), the distance to the nearest obstacle, and the battery level (step 3). The GCS sends back control commands to the drone (when necessary) which may be either i) a next valid position to be reached from a set of possible paths, or ii) an emergency retrograde action if the drone is unable to continue the mission, because of a weak battery charge for example (step 4).

When the destination is reached, the drone hovers around the accident location and broadcasts a live video stream of the situation (step 5), and continues to receive commands from the GCS to provide the appropriate medical supply, and assist the first aid gestures for the victims if required (step 6).

2.3 Related work

In the literature, some researchers aim to change the regular process of safety analysis, focusing mainly on negative causes and impacts of unwanted events, and take into account the success stories that deem insignificant [6, 81, 83]. With this aim, the authors promote a new safety analysis classification (Safety-I, Safety-II and Safety-III), and open up new perspectives on the consideration of resilience aspects. The reader is invited to refer to Section 1.3.4.3 for the definition of Safety I, Safety II and Safety III and their analysis.

Of the extensive literature around CPSs, I discuss some works related to the topics developed in this thesis. The European project CPSwarm aims to define approaches and tool chains to develop and test collaborative and reconfigurable autonomous CPSs (see `www.cpswarm.eu`). The project partners provide use cases about the usage of autonomous UAV systems in ambient environments, similar to our drone rescue system. Although, the case study was extracted from the CPSwarm workbench [130] and tailored with our interest to resilience.

Few academic works already exist around the study of CPS resilience because it

is a recent research topic. In [131], the proposed approach ensures resilience in CPS through self-healing structural adaptation. This involves adding and removing components, or even changing their interaction at run-time. The authors in [132] propose a hybrid theoretical framework for robust and resilient control design. The proposed framework is applied to the design problem of voltage regulators in synchronous machines and motors.

In [133], the authors distinguish between the notions of information and infrastructure dependability, and clearly illustrate the need to formally model and reason about the dependability aspects of CPS applications. In [134], the authors focus on the CPSs communication aspects and study the effects of intermittent data integrity guarantees on system performance under stealthy attacks. The authors in [135] also tackle the security issues in CPSs by identifying the problem of secure control, investigating the defenses that information security and control theory provide, and proposing a set of challenges that need to be addressed to improve the survivability of a CPS in case of cyber-attacks. In [136], the authors propose a software architecture of an agent-based production CPS and consider interoperability and data consistency aspects in their model. They also provide an implementation of such system to evaluate multi-agent approach with regards to conventional production processes.

Perhaps the closest work to the proposed methodology is [137]. The authors define a generic component-based CPS meta-model in UML, and show how it could be instantiated in concrete systems using a pattern-based methodology based on the so called Formal Concept Analysis (FCA) approach [138]. They also apply a knowledge-driven process to determine all kind of relations between the "cyber-" and "physical-" components of different subsystems and their underlying functionalities to deal with their related resiliency and redundancy properties. Our predictive analysis of resilience in CPSs is specific compared to their approach. By cons, our software design model is scalable in concrete implementations as emphasized in Subsection 2.5.1.

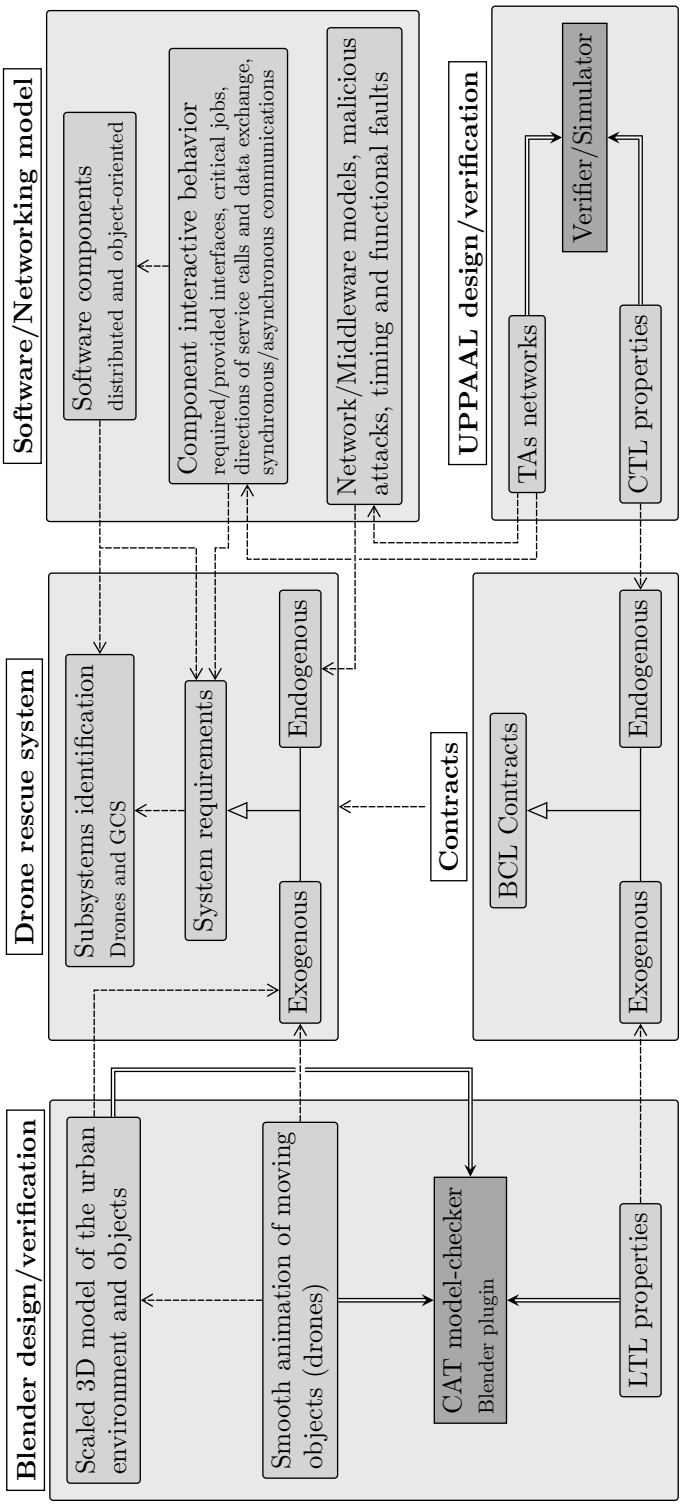


Figure 2.2: The diagram of our predictive methodology applied to analyze the endogenous and exogenous aspects of resilience in the considered drone rescue system; arrows significations: ⇢ uses or depends on, ⟹ tool input; ⟶ inherits from.

2.4 Proposed methodology

The graphic of Figure 2.2 represents the predictive methodology that I propose to address the endogenous and exogenous aspects of resilience in the drone rescue system. My *modus operandi* is firstly the identification of the subsystems involved, and the definition of their behavioral requirements.

Requirements are on the foundation of the design process, but are also used to define system *contracts* [139], required for the verification phase. As described in Section 1.6.1, a contract is semantically a Hoare triplet [140] that annotates a system *operation* by *assumptions* on the inputs and *guarantees* on the outputs. I use for that the Block Contract Language (BCL) [141] to extract contracts from requirements. BCL is pattern-based and semi-formal, easy to read, and convert to formal statements.

The software layer of our system is architected according to a Distributed Object-Oriented Component-Based Design (DOOCBD) approach extended from that of [12]. This new architecture embraces interface-driven composition, service-oriented interoperability, and direct data exchange between the behavioral jobs (tasks) of components. At run-time, these components are instantiated in distributed live objects that communicate (locally or remotely) while being deployed in connected software nodes embodied within the GCS and the drones.

In order to bring in design the real conditions of operation, I analyze the system behavior within abstracted models of wireless network and middleware: the scenarios of exchanging (Tx/Rx) data through the network physical layer should be emulated with respect to a bounded latency. In addition, I predict the system survivability when functional and timing faults or malicious attacks are detected.

The endogenous resilience involves the specification of all the critical scenarios and behavioral entities described above in TAs networks using Uppaal. Endogenous BCL contracts are then translated to safety and liveness CTL properties and analyzed by the simulator and model-checker of the toolbox. Further details are provided in Section 2.6.

Concerning exogenous resilience, the design step involves 3D modeling of the urban environment and smooth animations of moving objects. The benefit of 3D models is stating the obvious: they schematize concepts more distinctly than

any classic design language. Moreover, they allow an early feasibility of the system before realization. We use Blender with the built-in Contract Analysis Tool (CAT) to check if the exogenous BCL contracts, related to flight plannings and translated to LTL properties, are respected. If not, the flight path planning is corrected iteratively until the properties are met (see Section2.7).

2.5 Component-based architecture

The component architecture, depicted in Figure 2.3, presents the software structure of the drone rescue system, wherein nodes and partitions are not depicted. Given that it involves the use of several drones remotely guided by a GCS, its main components are highlighted: *Drone* and *GCS*.

As briefly stated in Section 2.2, *Drone* represents the main software unit of a connected drone. It is to perform the minimum of internal computations, and strictly follows the *GCS* commands, making it regularly aware of its state. All the heavy control operations, likely to consume an important amount of the drone energy (particularly the battery charge), are performed by *GCS* whose role is to fully control the drone mission remotely: paths computation, assistance decisions, retrograde actions, etc. It also controls the drone action on the crash site when filming the accident, and supplying the needed medical material.

Before starting the flight, the drone is activated by *GCS* by invoking the public (+) method *Activate*, provided remotely (@) from an instance of *Drone* running on the drone's embedded platform. Since it may simultaneously communicate with several drones, *GCS* maintains a map attribute *paths* (private –), associating each connected drone to a mission path, which is a cursor-moving array of records *Coordinates*, consisting of three fields: latitude, longitude and altitude. The flight path of each drone is initialized by *GCS*, and can be updated following unforeseen positions the drone may reach during its flight (*Control*). As soon as a position in the path is reached, the *Drone* instance updates (by the periodic job *Flight*) the next position by calling *Fly_To* from *GCS*.

In order to decide about the next position, *GCS* needs to be regularly informed of the drone's state (by invoking the method *Get_Data*). *Drone* periodically collects the following sensing data for *GCS*: 1) the current position (*Coordinates*)

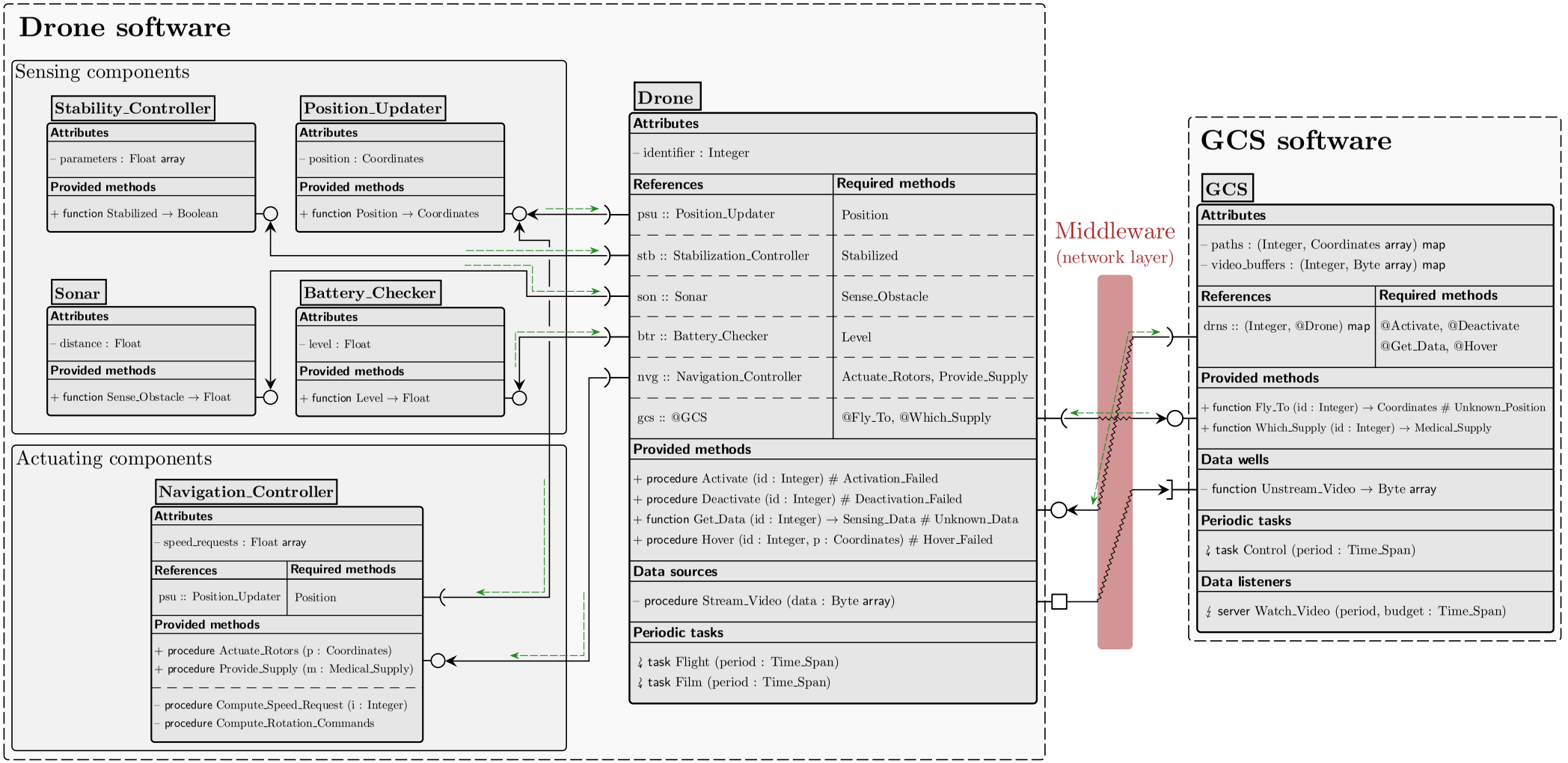


Figure 2.3: The software component architecture of the drone rescue system. Dashed arrows ⇢ represent the direction of data transitions when synchronous services are invoked: i) they are in the opposite direction of method call (continuous arrows ⟶) because transiting data are values returned or exceptions thrown to the caller components; ii) they are in the same direction of a method call arrow when data are directly passed as argument(s) to the called component.

from the component *Position_Updater* (that acquires the latitude and the longitude from a GPS unit, and the altitude from a barometric sensor); 2) the distance to the nearest obstacle from the component *Sonar* (using an ultrasonic sensor); 3) the battery level from the component *Battery_Checker*; 4) a stability boolean computed by *Stability_Controller* based on three-dimensional linear and angular acceleration parameters (resp. provided by an accelerometer and a gyroscope).

To reach the next position, *Drone* launches the speed control process by calling the method *Actuate_Rotors* (procedure) of the component *Navigation_Controller*. It computes the *Speed_Request* for the whole drone based on the individual *Current_Speed* of each rotor (provided by an instance of the component *Rotor* showed in Figure 2.4) and its current *Position*. The actual speed is computed for each rotor using the actual number of revolutions from the last speed computation (determined from the *pulses* generated by the motor *Encoder*). This sensor has a predefined resolution in PPR (Pulses Per motor Revolution), and is asynchronously read by the instance of *Rotor* using the data listener *Update_Pulses* (server) through the data well *Read_Pulses*.

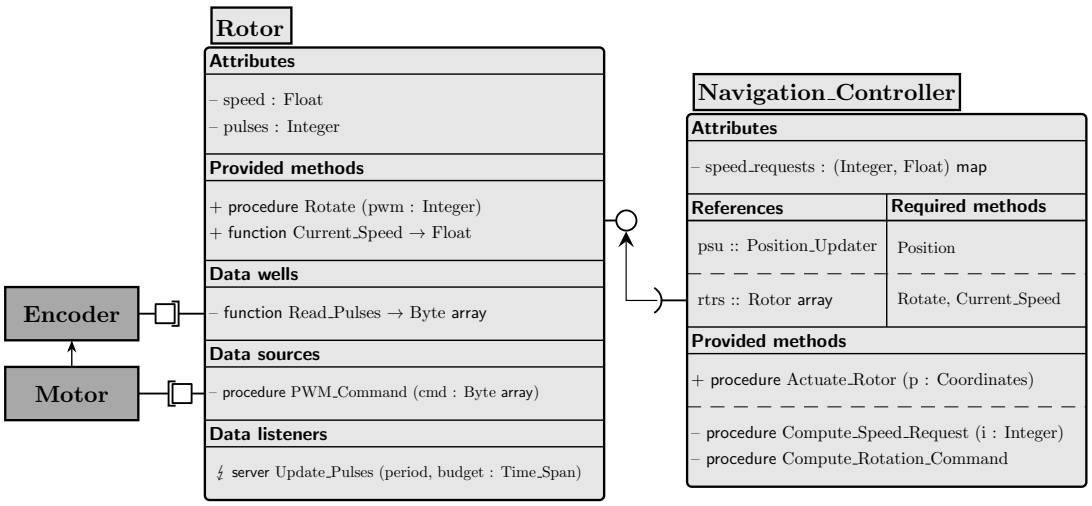


Figure 2.4: Concrete architecture and dependencies of *Navigation_Controller*

Once the speed request computed, *Navigation_Controller* actuates each rotor using a Pulse-Width Modulation (PWM) command (*Compute_Rotation_Command*)

passed to the method *Rotate* for each instance of *Rotor* interfaced with each motor (see Figure 2.4). The actual PWM value is proportionally calibrated in relation to the *error* between the speed request and the current speed. Each *Rotor* then applies the PWM command through the data source *PWM_Command*.

When the destination is reached, *GCS* signals the drone to hover around the accident (by invoking the method *Hover* provided by *Drone*), to broadcast a live stream video of the accident using the data source *Stream_Video*, and to provide the appropriate medical supply (*Which_Supply*), if needed. When the mission ends, the drone flies back to the starting position based on the same principle of the outward flight, and is deactivated (*Deactivate*) by *GCS* upon arrival.

2.5.1 Implementation feasibility

Object-oriented development has often been a hard sell in safety-critical systems industry [142]. The applied standards require extensive verification processes and real-time difficult to carry on by the dynamic aspects and flexibility of object-oriented paradigms (polymorphism, dynamic dispatch, late binding, overriding, etc). The distribution is also penalizing because of its semantics (message passing, remote dispatch and procedure call, etc). The Ada programming language is sufficiently expressive to implement our software model and can decidedly deal with these disadvantages. It is strongly typed and object-oriented with a powerful and explicit support for tasking, concurrency, multiprocessor architectures, compiler directives (*pragmas*), design by contracts [139] (the SPARK language [143]), etc. It allows developers to exploit the object-oriented assets while avoiding vulnerabilities and ensuring real-time [12, 142]. Besides, subsets of Ada are the target of many design and code generation toolboxes widely used in the industry (like SCADE Suite and Atelier B).

A speed control application for connected wheeled robot platoons, based on the DOOCBD approach, was discussed in [12]. The implementation is mainly in Ada, and based on annexes D and E of the Ada Reference Manual resp. of real-time and distributed systems. Annex E (abbreviated DSA) provides support for efficient distribution by making the middleware layer completely transparent and the development easier. The distribution in the application is managed by the

middleware PolyORB [144] (maintained by AdaCore).

The covered control scenarios and interoperability aspects in [12] are very close to those presented and discussed in this article. The Ada application is a proof of concept of our formal design, and we expect to contemplate prototyping real drones in the future. The development process will be much easier to approach. The predictive analysis presented in the next two sections has pre-chewed a lot of the work.

Concerning wireless connectivity in UAVs, an in-depth analysis of the implementation opportunities and challenges is provided in [145]. According to the authors, low-altitude short-range line-of-sight communication scheme seems to be the best suited to our case study, and may potentially lead to significant performance gains. This modality of wireless connectivity allows the dynamic adjustment of the UAV states to well suit the networking environment. For example, when a UAV experiences good channels with ground terminals, it can conserve energy to sustain good wireless connectivity in order to transmit more data to terminals. This is entirely what is needed in our medical rescue context. The standards IEEE 802.11p [146] and ITS-G5 [147] can be adopted to implement Drone-to-Drone or Drone-to-GCS communications since they suit such highly dynamic networking topology.

2.6 Analysis of endogenous resilience

In order to handle endogenous resilience in our context, we distinguish four different features which are: 1) the interactive behavior between drones and the GCS during their flights, 2) their behavioral actions (and reactions) while interacting with the GCS, 3) wireless network latency impact on timing predictability, and 4) survivability in the presence of malicious attacks and functional or timeout faults. Design and verification approaches dealing with the second feature, which cover particularly the individual internal behavior of component units, are well established both in academia and industry.

In this section, I provide a formal methodology to reason about (by abstract prediction) and verify together the first, the third, and the fourth features since they are challenging and lack of a deep investigation in the literature. The formal

design under UPPAAL reflects the relevant parts of the components behavior, which are related to their synchronous interoperability in the presence of middleware and network abstract models. Asynchronous data exchange is restricted to non-critical communications and hardware management and not considered in the proposed design.

Verification process

I opted for a verification-by-contract process making use of the text-based BCL language [141] to define the endogenous resilience contracts. A BCL contract $C \triangleq (\mathsf{A} : \bar{a} \mid \mathsf{G} : \bar{g})$ is an assume/guarantee statement where $\bar{a}$ is a vector of assumptions and $\bar{g}$ is a vector of the guarantees. Assumptions constrains whether a specification meets the guarantees. The BCL rationals should be as simple as possible to reason about requirements and their dependencies. They are very useful to decompose, make easier hard verification processes, and to exhaustively define the formal properties.

I start by a first contract on the worst-case call blocking time (WCBT) allowed for remote method invocations.

$$
\begin{array}{lcl}
\text{WCBT} & \triangleq & (\mathsf{A} : \text{awn} \mid \mathsf{G} : \text{mcr}) \\
\text{awn} & \triangleq & \mathsf{Always}\ [\ \text{wireless connection is reliable}\ ] \\
\text{mcr} & \triangleq & \mathsf{Everytime}\ [\ \text{a job calls a method remotely}\ ] \\
 & & \mathsf{Then}\ [\ \text{a response is received}\ ] \\
 & & \mathsf{Within}\ [\ x\ \text{tu (time unit)}\ ]
\end{array}
$$

Contract WCBT stipulates that the guarantee on the remote call responsiveness (mcr) is relative to the availability and reliability of the wireless network (assumption awn). Since assumption awn cannot be specified in UPPAAL, it is assumed to be always true. This contract can be specified for a periodic job by the following three CTL formulae (typewritten in the query language of UPPAAL):

$$
\begin{array}{lcl}
\texttt{WCBT1} & \triangleq & \texttt{A[] Time\_In imply h[job] <= MAX\_WAIT} \\
\texttt{WCBT2} & \triangleq & \texttt{A[] Time\_Out imply h[job] > MAX\_WAIT} \\
\texttt{WCBT3} & \triangleq & \texttt{Decision --> Delay}
\end{array}
$$

where `WCBT1` is a safety property stating that always in every trace of the job, being in the location Time_In means that a return value is received within

MAX_WAIT (h[job] is reset to 0 before the method call). Otherwise, the job response is timeout (h[job] > MAX_WAIT): Time_Out is reached (WCBT2), and timeout_count is decremented. The liveness property WCBT3 ensures that waiting is not infinitely blocking and the periodic activity always happen. - -> represents $\rightsquigarrow$.

WCET	$\triangleq$	(A : hao, wcbt | G : et)
hao	$\triangleq$	Always [hardware is reliable]
wcbt	$\triangleq$	Always [call blocking time satisfies WCBT]
et	$\triangleq$	Everytime [job periodic cycle is released]
		Then [job terminates its periodic activity]
		Within [p tu]

Contract WCET is about the worst-case execution time of periodic jobs. It stipulates that timing predictability (guarantee et) of periodic executions relies on the reliability of the embedded platform and components (battery, sensors and actuators, etc) of the drone (assumption hao) and the guarantee of Contract WCBT (assumption wcbt). Since assumption hao cannot be specified, it is assumed to be always true.

Endogenous	$\triangleq$	(A : hao, mcr | G : atr, cer, ttr)
atr	$\triangleq$	Everytime [intrusion is detected]
		Then [retrograded mode is activated]
		Immediately
cer	$\triangleq$	Everytime [critical error happens]
		Then [retrograded mode is activated]
		Immediately
ttr	$\triangleq$	Everytime [timeout happens several times]
		Then [retrograded mode is activated]
		Immediately

Contract Endogenous stipulates that whatever a malicious attack, a critical error, or recurrent timeouts happen, then the retrograde mode is activated (resp. defined by the guarantees atr, cer and ttr).

```
Malicious_Attack  ≜  E<> intrusion
Critical_Error    ≜  E<> critical_error
Max_Timeouts      ≜  E<> timeout_count == 0
Endogenous        ≜  (intrusion ||
                      critical_error ||
                      timeout_count == 0) -->
                      Retrograded_Mode
```

The first three properties should be checked to ensure that there is some traces where `intrusion`, `critical_error`, and `timeout_count == 0` hold eventually so as `Endogenous` is guaranteed to be checked on real traces. If only some of them hold, the liveness property `Endogenous` may hold with no trace to check for the others which is not representative.

Jobs *Flight* and *Control* resp. of the components *Drone* and *GCS* were checked to be deadlock-free (`A[] not deadlock`), and safe according to the verification process herein.

2.7 Analysis of exogenous resilience

Exogenous resilience is relative to the system's operations in its ambient environment. Concretely, it should be checked by simulation or by target tests on the signals acquired from sensors and/or applied on actuators. In order to analyze the exogenous resilience in offline, I provide a virtualization of the CPS in its environment to have a better understanding of safety-related issues, and to rigorously reason about them. In this section, I address the 3D modeling as a new CPS design perspective. A 3D approach could be used to represent the system into the real physical world.

The work done in this thesis comes as a continuity of [148]. My work completes it specifically for CPS visualization and for the evaluation of how such models can help engineers to rigorously reason about safety-related issues.

2.7.1 Why 3D modeling ?

A new viewpoint: Conventional 2D modeling methods are often hard to understand due to dependencies scattered across a large number of diagrams, state machines and behavioral descriptions. Moreover, designing complex systems requires a multi-disciplinary expertise. It takes time for engineers to effectively learn the modeling methodology. The 3D designs assigned with computational logic and verification techniques could be the way forward to system engineering. The work done in this thesis involves exploring and further elaborating this idea, with a rigorous evaluation of the advantages.

The benefits: The aim is not to replace conventional designing methods, but to acquire an added-value through which 3D design can be used in a variety of ways to benefit from a variety of advantages such as:

- A way to more effectively communicate with suppliers / customers / non-expert parties.
- Reinforce concepts and accelerate the design process by easily creating 3D models and animations for design reviews to be discussed in meetings.
- A better grasp and understanding over complicated concepts which is the case in modern CPS. As a study shows in medical field, students who were taught using 3D models answered quicker and had a better understanding than those who did not [149].
- 3D design provides a bridge between technical and non-technical people by creating a common framework environment.

2.7.2 Use cases

In order to validate the proposed approach for the analysis of exogenous resilience of CPS, I considered two different use cases which are the drone rescue system (previously described in Section 2.2) and the vehicular platooning system. Considering multiple use cases aims to reinforce the proposed methodology and validate the approach to analyze exogenous resilience of ACPS.

2.7.2.1 Drone rescue system

The drone rescue system has been previously described in Section 2.2. It is important to recall that within the drone rescue system, the GCS has the role of controlling drone missions remotely, and also performing all heavy computations likely to consume energy, vital for flights. Since it has a better awareness of the urban environment, the GCS performs the important task of *path planning*. By considering the current position of the drone, the destination and the urban cartography, the GCS computes by iterative correction the path that the drone needs to follow until the destination. Figure 2.5 illustrates the planning process on a sub-path between two positions. After determining the next position p_2 to be reached, the GCS acts lazily and considers the simplest path which is the segment p_1p_2 (Figure 2.5, left). Next, it discovers (by anticipated verification) that p_1p_2 intersects with a building (only permanent obstacles are considered). Finally, it recomputes the path by adding an intermediate position p_3 above the building (Figure 2.5, right).

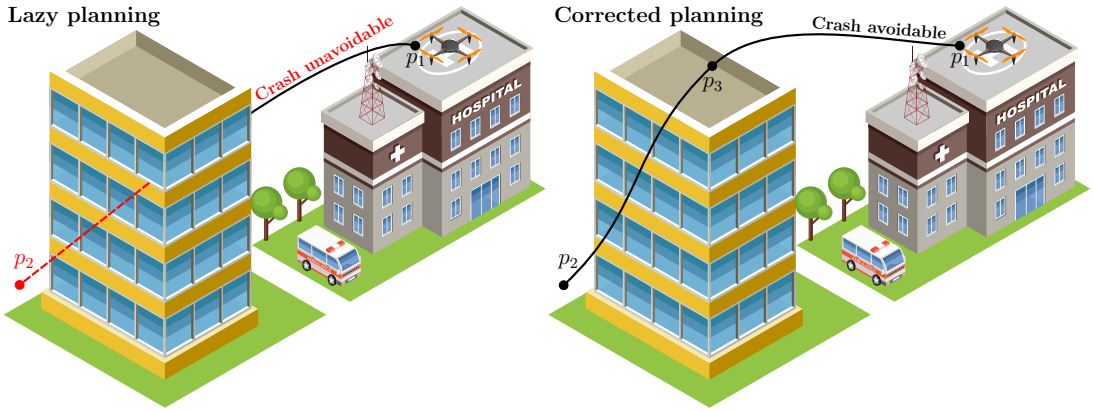


Figure 2.5: Path planning; *lazy planning* (left): the first planned path p_1p_2 of the drone between p_1 and the sub-target position p_2 is computed in a lazy way and leads to a crash with an obstacle building; *corrected planning* (right): GCS recompute a new path $p_1p_3p_2$ that passes over the building to avoid the crash.

The next step is dedicated to the realization of the 3D model. The details of choosing the tool and realizing the 3D models of the use case are discussed in Section 2.8, devoted to the tool development together with my feedback on the implementation issues.

After constructing the 3D model (ref. to Section 2.8), I define Contract Exogenous as follows:

$$
\begin{array}{lcl}
\text{Exogenous} & \triangleq & (\mathsf{A} : \text{ok}, \text{oaw} \mid \mathsf{G} : \text{cf}, \text{da}) \\
\text{ok} & \triangleq & \mathsf{Always}\ [\ \text{obstacles are buildings}\ ] \\
\text{oaw} & \triangleq & \mathsf{Always}\ [\ \text{GCS is aware of obstacles}\ ] \\
\text{cf} & \triangleq & \mathsf{Always}\ [\ \text{collisions are avoided}\ ] \\
\text{da} & \triangleq & \mathsf{Always}\ [\ \text{drone altitude} < \text{MA meters}\ ]
\end{array}
$$

Contract Exogenous states that a flight path is collision-free (guarantee cf), and the altitude coordinates of the path positions are always bounded by a maximum altitude MA fixed in meters (guarantee da). I assume that obstacles are restricted to buildings, and GCS is aware of their positions and dimensions (assumptions ok and oaw). I adopt LTL to specify contracts and automatically verify them (see Section 2.8 for the development details).

The above contract can be simply translated to the following LTL property pattern:

$$\mathsf{Safe}(\bar{p}) \triangleq \Box \left[\bigwedge_{p_i \in \bar{p}, B \in \mathcal{B}} (p_i \notin B \wedge p_i.altitude < \text{MA}) \right]$$

where $\bar{p}$ is the path, and $\mathcal{B}$ is the set of building obstacles B_i defined in the 3D city model as parallelepiped objects. Video animations of lazy and corrected flight paths are available under the following links to show whether the LTL properties are met or not:

- lazy planning: `https://youtu.be/MdaZhvlz_l8`
- corrected planning: `https://youtu.be/5cW6PBzoIj8`

2.7.2.2 Vehicle platooning system

The work presented in this section has been done in collaboration between several authors. The work addresses safety and security properties from early development stages to Ada code and prototype [12]. My personal contribution has been the contract specification, the 3D model representation, tool development and exogenous resilience analysis.

The concept of autonomous vehicle platooning aims to increase roads capacities and traffic fluidity. Autonomous vehicles are organized in tightly controlled platoons that operate close together. A highway for example can accommodate more vehicles when organized as platoons compared to classic human driving [150].

Adaptive Cruise Control (ACC) systems are well-known in vehicle platooning systems and currently available in many of upscale vehicles. A vehicle with ACC is commonly equipped with front radars. When a preceding vehicle is detected by these radars, the ACC system adjusts the vehicle's velocity in order to maintain a fixed time-gap to the preceding vehicle. All this happens without the driver's intervention. The follow-up is the Cooperative Adaptive Cruise Control (CACC). This concept augments ACC with wireless communication capabilities and enables a closer inter-vehicular cooperation which improves the traffic flow. Wireless communication allows vehicles to extend view beyond the line of sight of the front radars and allows faster transmission of speed updates between vehicles. However, in both types of system, the driver is still partly responsible for the vehicle's operation [151].

By adopting an Autonomous Connected Vehicle Platooning (ACVP) concept, control becomes fully automated, driver-free and cooperative. The Automated Highway Systems (AHS) is a variant of ACVP systems and has been under research by the Program of Advanced Technology for Highway (PATH) for several years. It aims to make vehicles in highways guided autonomously to their destination under controlled and optimized traffic flow for maximum efficiency and safety.

Platooning control functions The main functions to control the behavior of vehicles in a platoon are mostly: longitudinal and lateral control, string stability, lane tracking and changing, maneuver coordination for platoon formation and split. These control functions are introduced in [12].

The longitudinal speed control [152] consists in adapting the vehicle's velocity compared to that of the preceding one using the throttle and brakes. Implementation of longitudinal control are also high dependent on the headway from the preceding vehicle. Front-radar and image-processing sensors are typically used to get measurements of these inputs. It should provide comfortable ride for passengers and be accurate so that safety can be guaranteed.

The lateral control [152] consists in keeping the vehicle in the middle of the road (or the lane) by tracking its median trajectory. Designing such functionality involves a trade-off between the ride quality and the system accuracy, just like for longitudinal control. The challenges handled in the design of lateral control systems include high-speed operation using a purely "look-down" sensor system without transitional lateral position measurements. It is also concerned with lane changing from the current lane to an adjacent one. This aspect of lateral control is considered to be the most challenging as it involves more vehicle dynamics, changes of the radar targets but also more coordination and communication between vehicles.

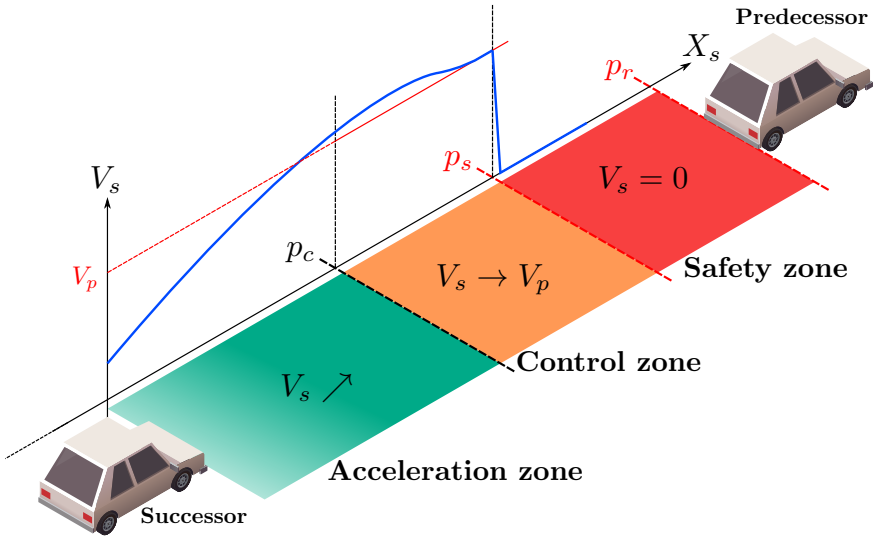


Figure 2.6: Longitudinal speed control. Figure extracted from [12]

The relative distance between a vehicle and its predecessor is subdivided into three zones as schematized in Figure 2.6:

- *Safety* zone (SZ): this is the area behind the predecessor vehicle between its rear position p_r and the limit the successor shall not cross, that is $p_s = p_r - \text{ds}$ with ds is a constant *safety distance*;
- *Control* zone (CZ): this is the area beyond SZ between p_s and the position from which the successor starts to stabilize gradually its regime V_s so that

the safety distance ds is maintained between them, that is $p_c = p_s - \text{os}$ with os is a relative distance called the *stabilization offset*;

- *Acceleration* zone (AZ): being in this zone, the successor is still far from the predecessor and has a leeway to accelerate briskly and reach CZ quickly.

When the successor's front position X_s exceeds p_c, it requests, at periodic instants, the predecessor's velocity V_p (constant in Fig. 2.6) which in turn responds by sending the information before the next request. This is critical: the exchange delay should be deterministic to guarantee a safe and stable behavior. The successor adapts accordingly its acceleration so that both vehicles roll at the same velocity. The stabilization offset os shall be large enough to prevent bodywork shake-up during speed control. Shake-up occurs when the successor enters SZ while it is reducing velocity to align progressively with that of its predecessor, braking is triggered prematurely.

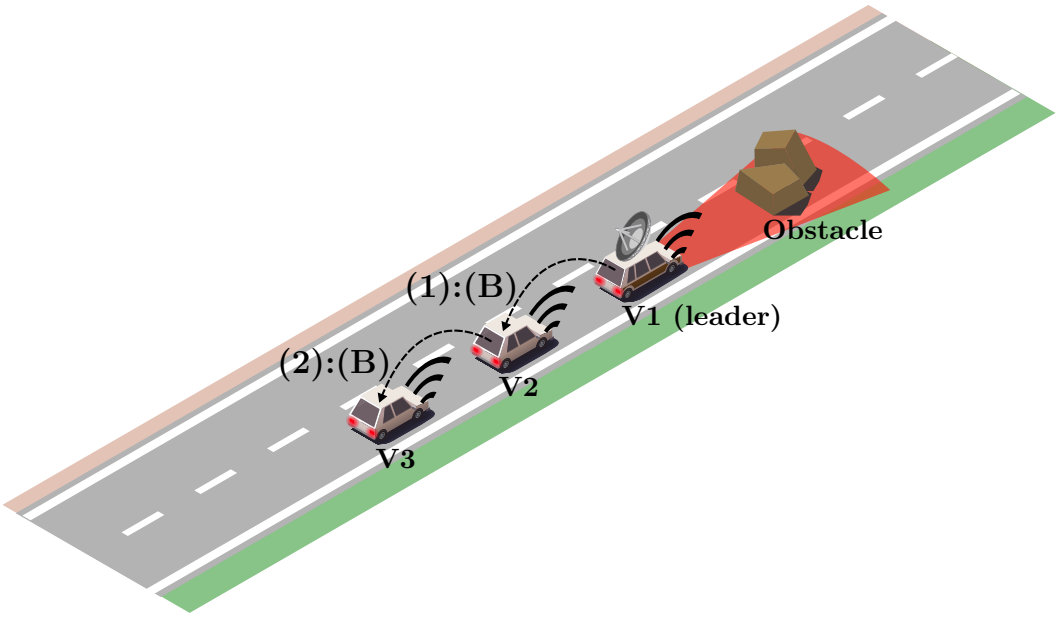


Figure 2.7: Platoon obstacle handling: the steps (1) and (2) represent the braking alarm (B) propagation to the followers by the leader's detection of the obstacle. Figure extracted from [12].

The second control scenario is depicted in Figure 2.7. When the leader detects an obstacle, it brakes immediately and alerts (by V2V) V2 to perform an

emergency brake and propagate alert to V3.

First, I represent this scenario as a 3D model. Modeling details are presented in Section 2.8.

After constructing the 3D model, I define Contract Exogenous as follows:

$$
\begin{array}{lcl}
\text{Exogenous} & \triangleq & (\mathsf{A} : \text{ffr}, \text{V2V} \mid \mathsf{G} : \text{ca}, \text{sd}) \\
\text{ffr} & \triangleq & \mathsf{Always}\ [\ \text{Functional front radar}\ ] \\
\text{V2V} & \triangleq & \mathsf{Always}\ [\ \text{V2V communications}\ ] \\
\text{ca} & \triangleq & \mathsf{Always}\ [\ \text{collisions are avoided}\ ] \\
\text{sd} & \triangleq & \mathsf{Always}\ [\ \text{distance between each vehicle} > \text{safety distance}\ ]
\end{array}
$$

Contract Exogenous states that the system needs to guarantee that collisions are avoided (guarantee ca) and also that *safety distance* is always respected (guarantee sd). I assume that the front radars are always functional in order to detect the obstacles and also that V2V communications are always functional so that the signal sent by the leader can be transmitted to the following vehicles. The Blender built-in tool CAT is used to specify and verify the LTL properties of the scenario. We define a model specific variable that we call *front_distance*. For each vehicle constituting the platoon, a variable corresponding to the distance between the vehicle and the nearest obstacle in front of it is assigned. Checking if the Contract Exogenous is valid is equivalent to comparing the *front_distance* to the *safety distance* for each vehicle and for each frame of the model execution. If, for each vehicle, the distance between it and the vehicle in front of it is greater than the safety distance, then it is possible to guarantee that there is no possible crash. Otherwise, this guarantee is not valid.

2.8 Tool implementation and feedback

This section is dedicated to my work on tool development and implementation for both endogenous and exogenous resilience. I will also provide a feedback concerning the work that I have done in order to highlight the main challenges that I encountered during the realization of this work.

2.8.1 Endogenous resilience: UPPAAL

In order to analyze the endogenous resilience, I adopted the tool UPPAAL to represent the system as a network of timed automata. I chose UPPAAL because it is an integrated tool environment for modeling, simulation and verification of real-time systems [128]. It is appropriate for our use case because the formal design under UPPAAL reflects the relevant parts of the components behavior, which are related to their synchronous interoperability.

The models has been introduced in [8], and are available for download at `https://github.com/mouelhis/uppaals`. I have participated to elaborate the UPPAAL models of the periodic jobs *Control* and *Flight* resp. of *GCS* and *Drone*.

The design that I propose is divided into two sub-models `Drone_Flight.xml` and `GCS_Control.xml` (respectively Figure 2.8 and Figure 2.9) and constrained by urgent and committed locations (when possible) in order to reduce the state space, prevent false counterexamples, and speedup model-checking.

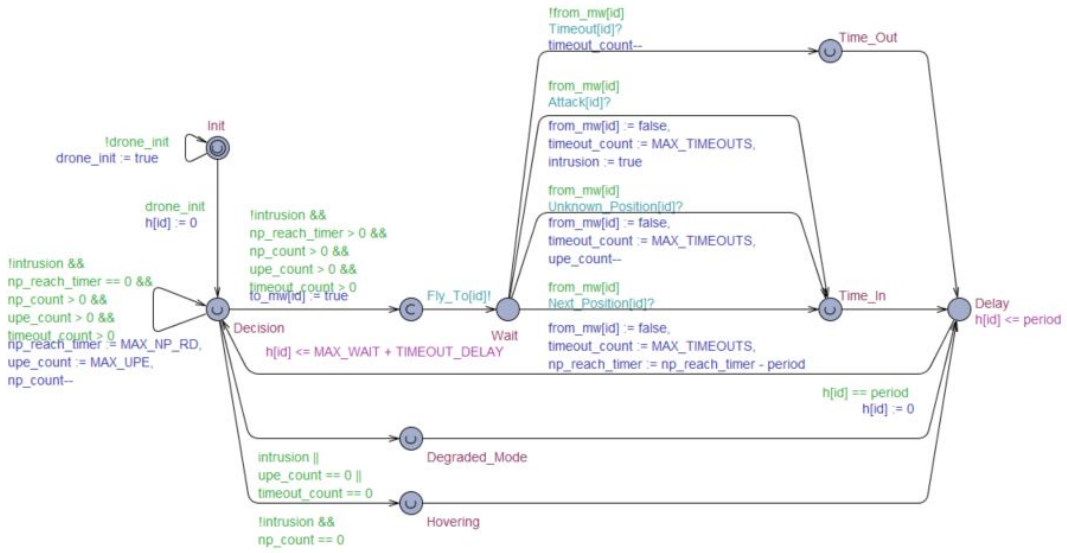


Figure 2.8: UPPAAL model of the drone's job: flight

I consider a model with two drones and one GCS. The GCS model is *multi-task* and *multi-call*: the job *Control* is instantiated twice one per connected drone; several method calls can be made simultaneously by drones in the node GCS. However,

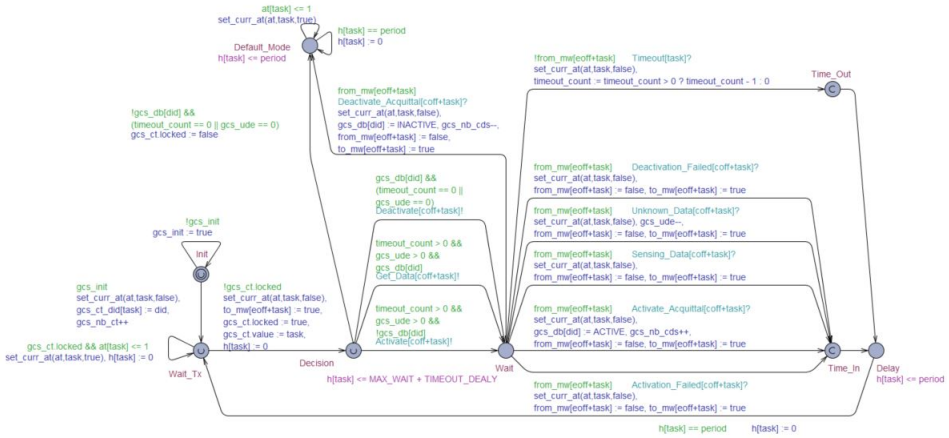


Figure 2.9: UPPAAL model of the GCS's job: control

the drone model is *mono-task* and *mono-call*: the job *Flight* is instantiated once in a drone node, and a method cannot be invoked simultaneously several times since only the GCS calls methods from drones, and calls are synchronous. The network communication band is composed of three half-duplex tracks: messages cannot transit simultaneously on a track regardless of their directions (*GCS* to *Drone* or *Drone* to *GCS*). The middleware Rx receives data from each subsystem in a circular FIFO buffer.

2.8.1.1 Analysis

Using these models, I analyzed by prediction the safety behavioral requirements of the GCS and drones when remote communication timeouts, malicious attacks, and functional errors occur. The properties were defined according to the verification process detailed in Section 2.6. The model's properties took 45 mn to be verified. The model-checking was performed using the 64 bits version 4.1.19 of Uppaal running on a Core i7-4710MQ machine at 2.5 GHz.

2.8.1.2 Feedback

This section is dedicated to my feedback about using Timed Automata, patterns and UPPAAL tool in order to analyze endogenous resilience of ACPS. When I started this work, I was not familiar with timed automata neither with UPPAAL.

I had basic notions about transition systems and classic automata. It took me several weeks to get familiar with this new concept. Throughout my learning process, the following papers have been particularly helpful [87, 153]. I highly recommend them for all beginners who want to learn about timed automata.

The models that I have implemented are not exhaustive but sufficiently representative to verify endogenous contracts under the hypothesis of considering a maximum number of two drones. I considered UPPAAL in order to represent the considered system as a network of timed automata. It took me several weeks to get familiar with this tool and to get the models done. This learning phase has been done thanks to the help of Mr. Sebti Mouelhi, PhD, who is an expert in formal methods. My representation of the GCS/Drones system suffers from the combinatorial explosion problem when I tried to address several drones to be controlled by the GCS. I began to face to deadlock problem and the endogenous resilience properties could not be verified. I suggest that simplifying the drone and the middleware models by reducing the state number could resolve this problem and could make it possible to verify endogenous resilience properties over a fleet of drones.

2.8.2 Exogenous resilience: CAT

In order to analyze the exogenous resilience, I represent the overall system as a 3D model, specify the contract-based properties in LTL and analyze them by an ad-hoc plug-in (CAT).

2.8.2.1 Choosing the tool

As a tool for 3D modeling, we use **Blender**. Blender is a free and open source 3D modeling software [154]. I chose Blender as modeling tool, because it offers the flexibility and expression power in order to implement verification over 3D models. Blender's internal libraries are implemented in Python and C++. The built-in Python console and text editors makes it simple to test and customize scripts. And since it is an open source software, it is possible to change and adapt any aspect of it at will.

2.8.2.2 Definitions and key concepts

Frame is a visual drawing used as a time quanta for smooth transitioning while displaying a certain animation. Frames Per Second (FPS) is a variable that can be manipulated when rendering graphics to choose how smooth the animation is.

Scenario is an ordered sequence of *frames*. *Keys* represent the set of values of all parameters involved within the model at a specific *frame* such as: objects location, physical (or logical) constraints, computation assignment to an object, value of a signal (non exhaustive list). Since it is impossible to meticulously define the animations frame per frame, *interpolation* is applied to solve this issue. Blender allows the definition of pertinent positions or keys such as the input: At frame $F-1$, Object A is at position $(0,0,0)$ and at frame $F-60$ Object A is at position $(10,10,10)$. 3D modeling software have pre-built algorithm to interpolate the input and associate Object A's location to all the frames from 2 to 59. The created sequence forms a scenario. In Blender, it is possible for the user to choose the adequate interpolation method between three different interpolations which are: *constant*, *linear* and *Bézier*. The most used interpolation method is the Bézier one since it produces smooth and more realistic animations.

Object properties Every object in the 3D model has predefined properties. These properties can be extended by creating any model-specific variables that can be attached to real and physically visible objects or virtual and invisible objects. Common properties for an objects are:

- Location: (X,Y,Z) coordinates that can vary as specified by the *key frames*.
- Rotation: (X^O,Y^O,Z^O) coordinate angles which specify angular rotation.
- Parenting and constraints relations: a precise description of dependencies between objects and what rule govern the interactions such as deformation or collision.

2.8.2.3 Contract Analysis Tool (CAT)

I adopt and update previous work done in [148] in temporal verification of animated scenarios and, using Blender, develop Contract Analysis Tool (CAT) for verifying temporal constraints over 3D models. CAT requires a 3D model of the system. This model is created with functional objects, constraints, physical and logical dependability between objects or computational units defined. Once the 3D model defined, it is then possible to attach animated scenarios describing the system's behavior including but not limited to: physical objects movement, signals propagation, material deformation. As previously discussed in Section 2.4, the defined contracts that are related to the exogenous resilience are specified using Linear Temporal Logic (LTL) syntax. I intentionally adopt LTL because it is widely used in contract based tools [155, 156] and specifications. As showed in Figure 2.10, the user can define a list of propositions and then use them in a temporal constraint. Then, by clicking on the button "Verify Temporal Constraint", the tool verifies if the model satisfies or not the temporal constraint. If it does, the sentence "The model satisfies the constraint" (as in the example illustrated in Figure 2.10). Otherwise, the sentence "The model DOES NOT satisfy the constraint" is displayed.

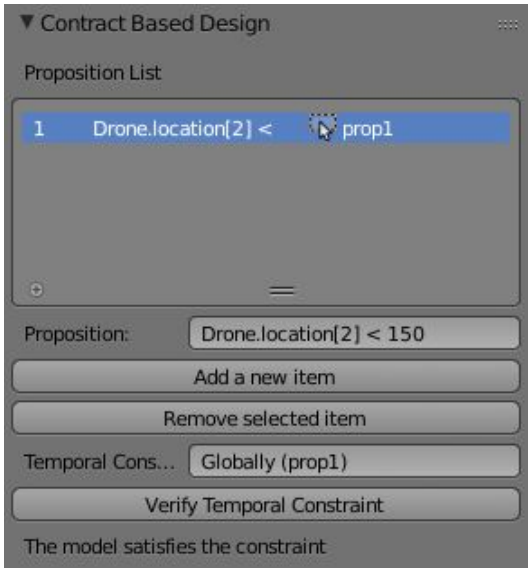


Figure 2.10: Contract Analysis Tool (CAT) interface in Blender

Verifying temporal constraints over 3D models requires assigning states to *frames*. For example: $G\Phi$ (Globally Φ) is true if Φ is true in every frame. As a result, the notion of time is abstracted to frames: it is possible to choose which time unit to assign to each frame by manipulating the FPS (Frames Per Second).

2.8.2.4 Modeling the use case

Use case: Drone rescue system

Using Blender, I specify a 3D model illustrating this use case. The model represents a city with different buildings and streets. (Remark: the 3D model is used as an interface with CAT and does not represent a real town.) Once the GCS receives the signal that there is an accident somewhere in the city, it sends the command to the drone to take off and to fly in order to explore the accident location (Figure 2.11). After take off, the drone begins the navigation phase which is the most important and the most critical. It is essential to take the shortest path in order to reach the destination as fast as possible, but it is also essential to avoid obstacles.

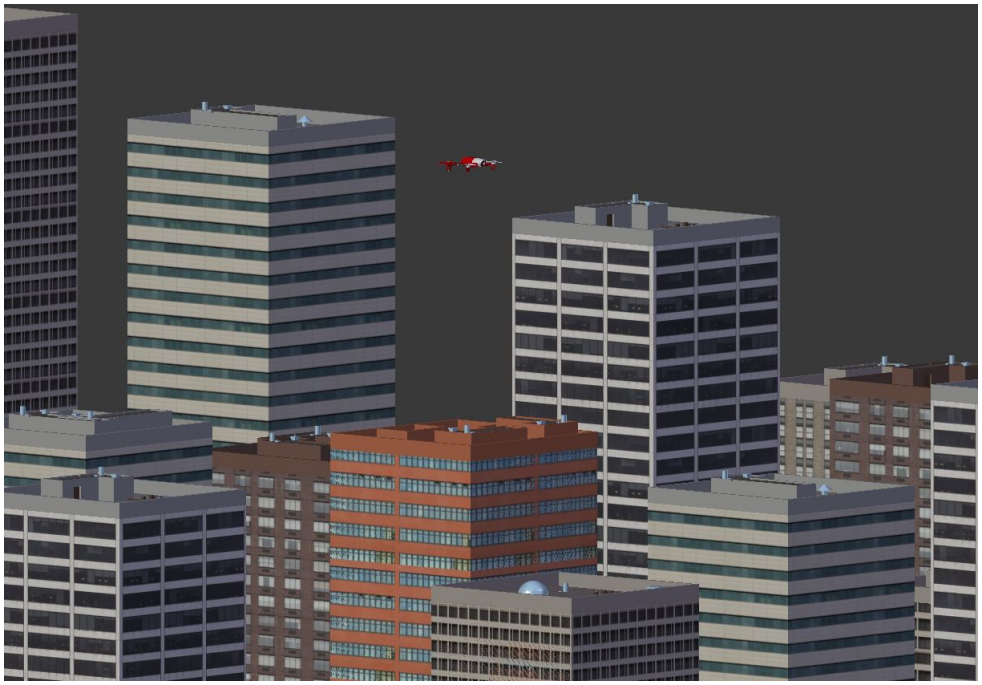


Figure 2.11: Blender 3D model of the drone rescue system

Use case: Vehicular platooning system

In order to analyze the exogenous resilience of the vehicular platonning system, I consider the scenario of three connected vehicles V1, V2 and V3, firstly developed in [12]. Vehicles V1 and V2 are already forming a platoon of which V1 is the leader. At the instant t_0 of Figure 2.12 (left), in order to tail-merge in the platoon, V3 sends its remote reference by wireless connection to a mobile *base station* (BS) installed on the leader when entering its *coverage area* (step 1). In turn, BS sends back the references of V1 and V2 to V3 (step 2). We talk here about Vehicle-to-Base (V2B) communication. Once connection is established and the references of V1 and V2 are acquired, V3 can consequently communicate directly with each of them. I point here that the coverage area of BS should be larger than that of vehicles to detect the approach of new merging vehicles as soon as possible. At $t_1 > t_0$ of Figure 2.12 (middle), V3 accelerates briskly to catch up with V2 (step 3). The front-radars of V3 and V2 are clearly used to compute the distances to their predecessors resp. V2 and V1. By approaching V2 at $t_2 > t_1$ of Figure 2.12 (right), V3 controls velocity so that collision with V2 is avoided by respecting a prefixed minimal inter safe distance. Besides, stability should be guaranteed for the platoon by preventing shake-up in case where a vehicle does not respect the safety distance to its predecessor and brakes prematurely (step 4). The speed control of a vehicle is defined based on the velocity of its predecessor communicated by Vehicle-To-Vehicle (V2V) under real-time determinism.

During the analysis of exogenous resilience for the vehicle platooning systems, I only focus on the longitudinal speed control. A special attention is given to Vehicle-To-Everything (V2X) communication technologies as they represent a major upgrade in improving passengers comfort, preventing dangers and they also promote a smooth transition to fully automated vehicles [12]. I consider two common control scenarios in ACVP systems based on V2X communications to illustrate how our proposed methodology is suitable for the analysis of exogenous resilience of ACPS. These scenarios are: 1) the tail merging of a new connected vehicle in a platoon already in circulation, and 2) the propagation of braking alarms to followers when the leader vehicle detects an obstacle.

Using Blender, I construct a 3D model illustrating a platoon of three vehicles going on a highway (Figure 2.13). The leader of the platoon detects a static

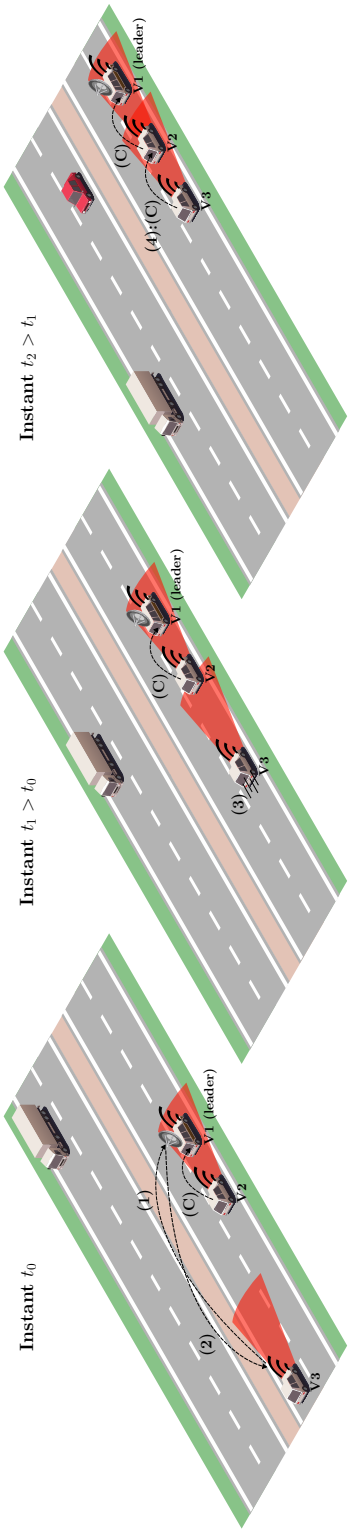


Figure 2.12: Figure extracted from [12]. Tail-merging in platoon already in circulation; step (1): V3 requests the references of vehicles covered by BS installed on the platoon's leader vehicle (V1); step (2): BS sends back to V3 the references of V1 and V2; step (3): V3 tail-merges in the platoon (already composed of V1 and V2) and accelerates to catch up with V2; step (4): V3 controls (C) its velocity based on that of V2 in order to keep a minimal safety distance and avoid collision between them. Figure extracted from [12]

obstacle on its lane (a stationary lorry in our scenario) and starts braking. A signal is sent to the following vehicles who perform at their turn an emergency brake.

Figure 2.13: Blender 3D model of the vehicular platooning system

2.9 Conclusion

In this chapter, I introduced a methodology for the safety analysis of ACPS and discussed it via the drone use case. I decompose the resilience of ACPS into two different categories: endogenous and exogenous. Both resilience properties are specified via contracts, which provide a uniform tool-independent formalism useful to reasoning about the system. Endogenous resilience contract-based properties are analyzed via timed automata and UPPAAL tool. Exogenous resilience contract-based properties are analyzed via 3D modeling and CAT (Contract Analysis Tool). The methodology is supported via two use cases: the drone rescue system and the vehicular platooning system. My obtained results consolidate the viability of the methodology and demonstrate the feasibility of verifying safety-related properties using 3D models and CAT. The next chapter is dedicated to the investigation of the use of Artificial Intelligence (AI) on safety of CPS and more specifically the use of genetic algorithms for drones navigation.

Chapter 3

Safety and autonomous navigation for ACPS

Contents

3.1 Introduction

In the previous chapters, I analyzed resilience of ACPS by considering their interactions with the ambient environment. In this chapter, I focus on how ACPS navigate autonomously in their environment by considering the use case of drones in urban environments. The overall aim is to address resilience for autonomous navigation. The work presented in this chapter constitutes a first step into analyzing autonomous path planning of drones in urban environment. A relevant perspective of this work would be to consider and study uncertainty degree in autonomous navigation algorithms and its impact on resilience.

Urban environments have a high density of physical obstacles (such as buildings) as well as no-fly zones (such as airports), making the navigation aspect particularly challenging.

I start be giving a brief overview on autonomous navigation and existing approaches for solving path planning problems. Then, I describe the specific problem and I define a cost function, which allows me to evaluate the expected results. Afterwards, I propose a non-deterministic approach, based on Genetic Algorithms, for solving the problem. To study the impact on the safety level of the system, I compare the achieved results to the ones obtained by applying the deterministic algorithm A*.

3.2 Autonomous navigation

Autonomous navigation is one of the most important requirements of an intelligent vehicle, drones included. Robot navigation is a designed process toward a target position while avoiding obstacles. As well described in [157], robot navigation can be broken down into four basic components which are: (i) *perception*, the robot uses its sensors to extract meaningful data from its environment; (ii) *localization*, the robot determines its location in the environment; (iii) *path planning*, the robot determines the path that will be followed in order to reach its goal; (iv) *motion control*, the robot regulates its motion in order to accomplish the desired trajectory.

Robot path planning problem usually consists of finding a path plan allowing the robot to travel between two locations labeled as the start and destination loca-

tions respectively, to carry out a specific mission. The path must be collision-free (or feasible) and satisfies use-case-dependent optimization criteria [158]. Straightforwardly, any navigation algorithm embraces at least three different robotic fields: navigation, safety and performance.

Robot path planning methods can be divided into classical and heuristic methods. Most important classical methods consist of cell decomposition method, potential field method, subgoal method and sampling-based methods [157]. Heuristic-based methods include neural networks, fuzzy logic, nature-inspired algorithms such as genetic algorithms (GA), Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). [157] is a survey that gives a well-established overview on the heuristic approaches involved in robot path planning.

Finding the shortest path in a graph by ensuring compliance with additive constraints has been recently proved as NP-hard problem [159]. The author addresses the problem of constrained navigation with mandatory waypoints for vehicle navigation. The proposed approach combines constraint solving techniques with an Ant Colony Optimization (ACO). The hybridization relies on a static probing technique which builds up a search strategy using a distance information between problem variables and a heuristic solution.

Broadly speaking, all the existing approaches used for solving the robot path planning problem can be classified into two different techniques [160,161]: (i) *global path planning* or off-line path planning and (ii) *local path planning* or on-line path planning. A global path planner usually generates a low-resolution high-level path based on a known environment map. The method is valuable of producing an optimized path. However, it is inadequate reacting to unknown or dynamic obstacles. On the other hand, local path planning algorithms do not need environment information in advance. It usually gives a high-resolution low-level path only over a fragment of global path based on data incoming from ob-board sensors. Local path planning works effectively in dynamic environments. However, it is inefficient when the target is away from the start position or if the environment is cluttered [157].

Recently, several **deep learning** approaches have been applied to local path planning [162]. Deep Reinforcement Learning (DRL) have achieved remarkable success in many challenging tasks [163, 164]. Different from previous supervised

learning methods, DRL based approaches learn from a large number of trials and corresponding rewards instead of labeled data. In order to learn a sophisticated control policy with reinforcement learning, robots need to interact with the environment for a long period to accumulate knowledge about the consequences of different actions [162]. Collecting such interaction data in real world is expensive, time consuming, and sometimes infeasible due to safety issues [165].

Obviously, the combination of both global and local path planning techniques [160, 161] is advised to enhance their advantages and eliminate some of their weaknesses, e.g. [166–168]. For example, in [169], the authors introduces EDNA (Exploratory Digraph Navigation Using A*) as an autonomous navigation system for robots in a partially unknown environment.

Finally, advances in autonomous navigation algorithms are often studied separately by other fields related to robotics, such as sensor data processing and mechatronics, resulting in a lack of cross-layer integration and synchronization that affect safety due to incompatible timing constraints and error handling of different components in the autonomous robot. To this end, in [170], the authors introduce LEN Safety (where LEN stands for Lifelong Exploratory Navigation), an integrated contract-based robot navigation stack from sensors and actuators to artificial intelligence functionality. LEN Safety is resource- and safety-aware and it allows continuous operation of a mobile robot within a complex and uncontrolled environment.

3.3 Problem description

In natural language, the problem can be described as follows:

*Given a map with a set of **physical known obstacles**, a set of **regulatory-related obstacles (RRO)** and the drone's battery level, the drone must take off at **Start** and reach the **Destination** while avoiding all the physical obstacles. The drone's path should contain as few as possible RRO and its length must be less than the drone coverage range.*

As previously stated in section 3.2, there are two main approaches for solving robot path planning problems which are: offline and online path planning. In the

case of existing drones where the battery lifetime is very limited, the most appropriate approach is the offline path planning. This leads to deport all computation-consuming operations to the Ground Control Station (GCS). The GCS is in charge of computing the path planning process and sends the coordinates to the drone to follow. In our problem resolution, we consider a similar approach. We suppose that we have an offline map representing all obstacles and we execute the path planning process on the GCS before sending the path to the drone.

Path planning problem can often be associated with searching for the shortest path. In the case of UAV path planning, the optimal path is more complex and has to take into consideration multiple constraints such as drone battery, collision avoidance, etc. In order to consider these constraints, a cost function is used and the path planning algorithm is transformed to a search for a path that will minimize the cost function. Minimizing the path cost corresponds to maximizing the desired characteristics fulfillment [171].

3.4 Method

3.4.1 Cost function

In order to define the cost function, I capitalize on the work done in [171], where the authors establish a comparison of parallel genetic algorithm and particle swarm optimization for real-time UAV path planning.

The cost function, which I propose, is defined as follows:

$$F_{cost} = C_{battery} + C_{RRO} + C_{length} \tag{3.1}$$

where $C_{battery}$ penalizes paths with length superior to the drone battery level, C_{RRO} penalizes paths that crosses regulatory-related obstacles (RRO) which are defined by regulatory authorities, and C_{length} penalizes longer paths. C_{length} and C_{RRO} are optimization criteria used to improve the quality of the paths . C_{length} is defined to be in range of [0,1]. C_{RRO} is defined to be in range of $[0, N]$ where $N \in \mathbb{N}$ is the number of the regulatory-related obstacles existing in our map. On the other hand, $C_{battery}$ is a feasibility criterion that must be satisfied for the final path to

be valid. Feasibility criteria are defined to be equal to 0, when they are respected, or in the range $[P, P+1]$ when they are not, where $P \in \mathbb{R}_+^*$. In our case, we define P to be equal to N is order to ensure that unfeasible paths that have a length exceeding the drone coverage distance always have a cost greater than any feasible ones. It is important to note that in our implementation, we seek to minimize the number of regulatory-related obstacles but we sanction paths that have a length that exceeds the autonomy of the drone.

In this cost function, the term associated with the length of the path is defined as follows:

$$C_{length} = 1 - \frac{L_{P_S P_D}}{L_{path}} \tag{3.2}$$

therefore

$$C_{length} \in [0, 1] \tag{3.3}$$

where $L_{P_S P_D}$ is the length of the straight segment connecting the start point and the destination point, L_{path} is the length of the actual path.

The term related to regulatory-related obstacles is defined as follows: let N_{obs} be the number of RRO crossed by the given path and N be the number of total RRO existing in our map, then:

$$C_{RRO} = N_{obs} \tag{3.4}$$

therefore

$$C_{RRO} \in [0, N] \tag{3.5}$$

The term associated with the drone battery level is defined as follows:

$$C_{battery} = \begin{cases} 0 & \text{, if } L_{path} < L_{battery} \\ (N+2) + 1 - (\frac{L_{P_S P_D}}{L_{path}}) & \text{, if } L_{path} > L_{battery} \end{cases} \tag{3.6}$$

therefore

$$C_{battery} \in \begin{cases} 0 & \text{, if } L_{path} < L_{battery} \\ [N+2, N+3] & \text{, if } L_{path} > L_{battery} \end{cases} \tag{3.7}$$

where $L_{P_S P_D}$ is the length of the straight segment connecting the start point and the destination point, L_{path} is the length of the actual path and $L_{battery}$ is the

coverage distance depending on the battery level of the drone.

During the execution of the path planning algorithm, my goal is to find a solution which minimizes the cost function and, therefore, find the shortest path that is within the range coverage of the drone and crosses the minimum number of RRO. By considering a single cost function, there is a risk of not finding an acceptable solution. In my case,

$$F_{cost} \in \begin{cases} [0, N+1] & \text{, if } L_{path} < L_{battery} \\ [N+2, 2N+4] & \text{, if } L_{path} > L_{battery} \end{cases} \tag{3.8}$$

Consequently, I consider that all solutions with a cost function superior to $N+1$ correspond to unfeasible paths. The cost function that I defined is tailored to my use case and it can be easily modified and applied to other different scenarios as described in [171].

3.4.2 Environment representation

The first step of path planning is to discretize the world space into a representation that will be meaningful to the path planning algorithm. This representation is closely related to a search algorithm and some algorithms will only perform well when they are coupled with a specific environment representation [171].

In my implementation (see Figures 3.1 and 3.2), I capitalize on the work done in [171] by using an approximate cell decomposition of the terrain. We start by representing the environment as a 2D matrix. For each cell of the matrix, I assign a number that corresponds to the elevation of the terrain (or the buildings) in that location as shown in Figure 3.1.

In the considered use case, I assume that **the drone navigation will be executed at a constant altitude**. In other words, the drone will take off vertically at the start position, and once the operation's altitude reached, the drone navigates horizontally until attaining its destination position. Afterwards, the drone lands vertically. This assumption allows representing the environment as a two dimensional (2D) grid where all the cells with an elevation higher to a certain altitude are considered as physical obstacles as shown in Figure 3.2.

The European regulation of unmanned aircraft operations established by the

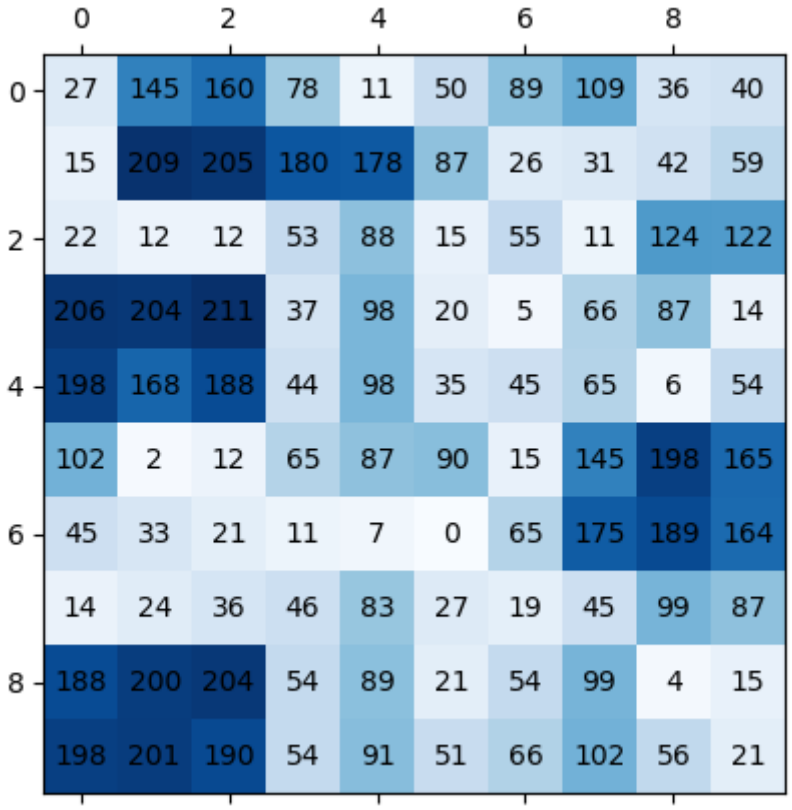


Figure 3.1: 2D matrix representation of the environment. The number in each cell represents the terrain elevation at that location

European Aviation Safety Agency (EASA) [172] divides drone operations into three categories as follows: open (low risk), specific (medium risk) and certified (high risk). For the open category, the aircraft is not allowed to operate at a height exceeding 150 meters above the ground [172]. I arbitrarily consider a minimal distance that the drone must respect with regards to physical obstacles such as building roofs. I define this distance to be equal to 15 meters. Consequently, as shown in Figures 3.1 and 3.2, each cell with a certain elevation higher than 135 meters is considered as a physical obstacle that the drone must avoid during its flight.

In addition to the physical obstacles taken into account, I also consider regulatory-related obstacles which are related to the drone operations regulation as shown in Figures 3.1 and 3.2. I refer to this type of obstacle by RRO. RRO include airports,

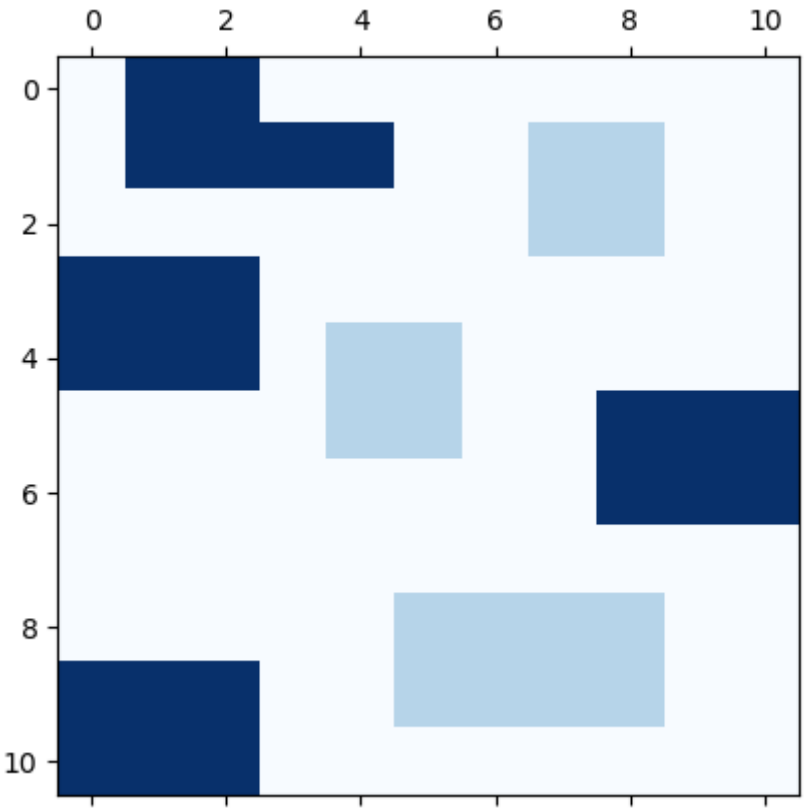


Figure 3.2: Obstacle representation. Physical obstacles are represented in dark blue and regulatory-related obstacles in light blue

stadiums, embassies, factories, *etc.* The drone is not allowed to fly over such areas unless the regulation authority allows him to.

During the path planning process, both physical and regulatory-related obstacles must be taken into account. The ideal path would be the shortest path to not include any obstacle. Nevertheless, there are multiple constraints involved in the path planning process. The first constraint is the drone battery autonomy lifetime. The battery embedded on the drone is relatively small in order to not impact the drone's weight. For instance, with small quadricopters, the autonomy is estimated between 20 and 30 minutes. This constraint impact directly the path planning process, as the obstacle-free path can be very long and the drone autonomy may not be sufficient to follow this particular path. The second constraint arises within environments with a high density of obstacles. For certain environments (mostly

urban), the combination of physical and regulatory obstacles is so important that the path planning algorithms are not able to generate any obstacle-free path. It is this particular problem that we address by generating a path that respects the drone autonomy, does not include any physical obstacle and contains as few as possible RRO.

3.4.3 Background

Of the extensive literature on existing autonomous navigation algorithms, I here discuss the definitions of A* and Genetic Algorithms which will be used in the proposed methodology.

3.4.3.1 A* algorithm

A* algorithm is one of the best known path planning algorithms, which can be applied on metric or topological configuration space [172]. This algorithm was initially designed for the graph transversal problems. Later, it was commonly used for path finding applications and had proved itself to be very effective for solving path finding problems in static environments. A* uses combination of heuristic and searching based on the shortest path. It is defined as best-first algorithm, because each cell in the configuration space is evaluated by the value: $f(v) = h(v) + g(v)$ where $h(v)$ is heuristic distance (Manhattan, Euclidean or Chebyshev) of the cell to the destination location and $g(v)$ is the length of the path from the start position to the destination position through the selected sequence of cells. Obviously, this sequence ends in the actually evaluated cell. Each adjacent cell of actually reached cell is evaluated by the value $f(v)$. The cell with the lowest value of $f(v)$ is chosen as the next one in the sequence [172]. Advantage of this algorithm is that the distances used as a criterion can be adopted, modified or another distance can be added. This feature gives a wide range of modifications of this basic principle.

3.4.3.2 Genetic algorithms

In order to resolve problems that are NP-hard (or NP-complete), a heuristic optimization approach is recommended [158]. One of these approaches is the use of

genetic algorithms. A Genetic Algorithm (GA) [173] is an evolutionary problem solving method, where the solution to a given problem evolves after a number of iterations. Based on the Darwin's theory of evolution, the GA simulates the evolution of a population of solutions to optimize a problem. Similarly to living organisms adapting to their environment over the generations, the solutions in the GA adapt to a fitness function over an iterative process using biology-like operators such as crossovers of chromosomes and mutations of genes. Genetic algorithms have been widely used to cope with the complexity of the path planning problems. As well summarized in [157], various studies have been executed based on GA in robot path planning domain. In [174], the authors address this issue by applying a knowledge based genetic algorithm (problem-specific genetic algorithm) instead of the standard GA. The algorithm is designed with both domain knowledge and small-scale local search. The proposed method is suitable in both static and dynamic environments. This algorithm is extended in [175] for multiple mobile robots in dynamic environments.

3.4.4 Problem resolution

As I aim to solve the problem described in section 3.3, I propose the approach illustrated in Figure 3.3.

In urban areas, the density of phyiscal obstacles and RRO is so important that it is very difficult to find a path that avoids all obstacles and that is feasible by the drone's battery. My goal is to test if it is possible to cross some RRO in order to accomplish the mission. If there is not a solution that is 100% safe, it may be acceptable to make a compromise by crossing some RRO. To this prupose, I apply the following reasoning: first, I apply A* algorithm to test if there is an available path without any RRO and that has a length inferior to the drone coverage distance. If not, I use genetic algorithms in order to generate a path that includes the minimum possible number of RRO and that takes into consideration the drone's autonomy. The method will be called throughout this section as Hybrid Genetic Algorithm (HGA) by combining multi-population genetic algorithm and A* which to evaluate each individual. The pseudocode of the proposed HGA is described in the following algorithm.

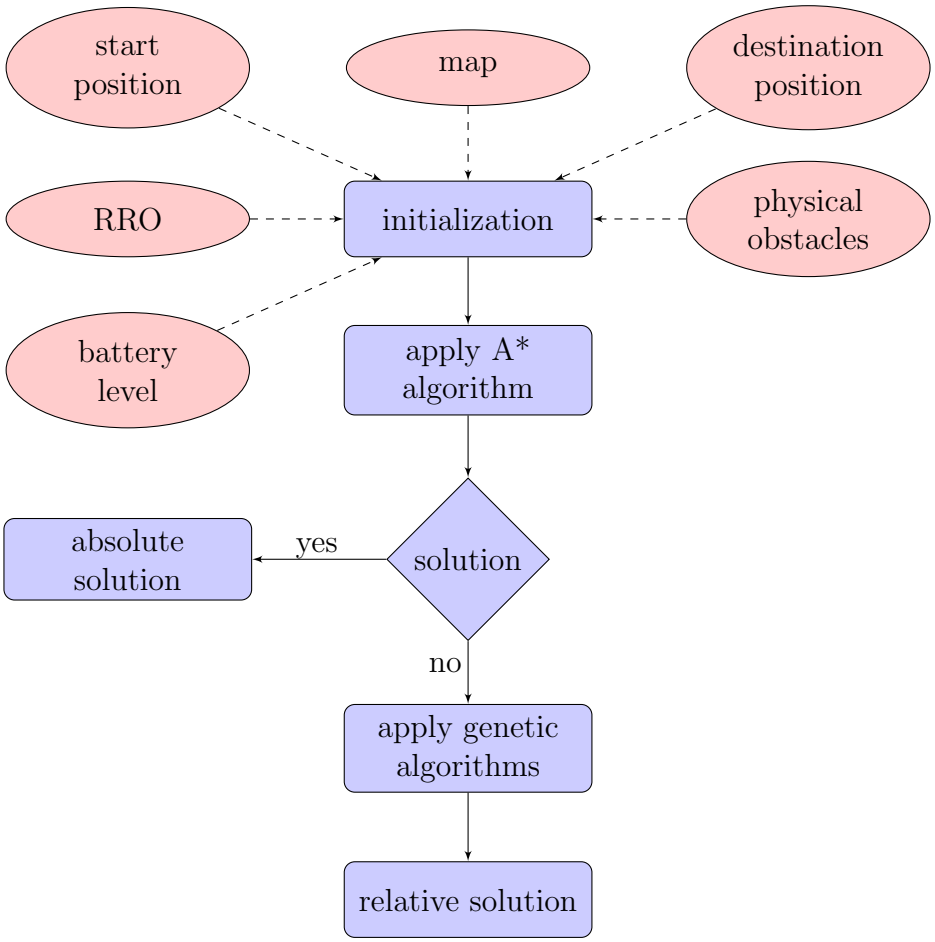


Figure 3.3: Flowchart of our approach

The procedure `createInitialPopulation` generates the first population. It is parameterized by the number of individuals per population and by the 2D map. Each individual is respectively a 2D map created as follows: first, obstacle-free zones and physical obstacles are kept same to the ones existing in the map. Second, RRO are randomly deleted for each individual by transforming zones that contains RRO to obstacle-free zones.

In subsection 3.4.3.2, we have highlighted the use of genetic algorithms to address path planning problems. To the best of our knowledge, all the algorithms proposed consider populations of paths and throughout the algorithm, the goal is to find the best path. In the proposed method, I consider **populations of maps** since my goal is to minimize the number of RRO while taking into consideration

Algorithm 1: Hybrid Genetic Algorithm

```
1  begin
2      numberOfGenerations = 100
3      numberOfIndividualsPerGeneration = 8
4      numberOfParents = 4
5      numberOfCrossovers = 10
6      mutationRate = 0.4
7      initPop ← createInitialPopulation
8      for i = 1 to numberOfGenerations do
9          if (i = 1) then
10             pop[i] ← initPop
11         fitness ← computeFitnessPop (pop[i])
12         parents ← selectMatingPool (pop[i])
13         add (parents, pop[i+1])
14         for j = 1 to numberOfCrossovers do
15             children ← crossover (parents)
16         for k = 1 to numberOfChildren do
17             λ ← a random value ∈ [0.0, 1.0]
18             if (λ ≤ mutationRate) then
19                 mutation (children[k])
20         add (children, pop[i+1])
21     bestIndividual ← selectBestInd (pop)
22     bestPath ← aStar (bestIndividual)
23     return bestPath
```

the autonomy of the drone. This approach is innovative and allows the reasoning about the safety of the aircraft.

Once the first population created, it is evaluated by calling the procedure `computeFitnessPop` which computes the fitness of each individual using the cost function defined in section 3.4.1. After assigning a cost to each individual, the function `selectMatingPool` is called to select the parents which are the individuals with the lowest cost function. The number of parents is entered as a parameter and the selected parents are added to the next population. The selected parents are also used to generate the children by applying the crossover and mutation operators. The number of generated children is defined as the number of individuals

per population minus the number of parents. This choice is made in order to make sure that all populations have the same size.

The procedure `crossover` mixes the genes of the parents two by two in order to generate children. This operation is not done randomly, however it is done according to the cost function of each parent.

I differentiate between three different cases:

- If the cost functions of both parents are greater than $N + 1$, i.e. if no parent have a path that is feasible by the drone's battery. In this case, the crossover is done randomly. Each child's chromosome (each map cell) is selected with a probability equal to 50% to be taken from one parent or another.

- If the cost function of one parent is lower than $N + 1$ while the cost function of the other parent is greater than $N + 1$, i.e. if one parent has a feasible path while the other has not. In this case, all cells covered by the path are transmitted to the child. The rest of the cells is filled with a probability equal to 50% to be taken from one parent or another.

- If the cost functions of both parents are less than $N + 1$, i.e. if both parents have feasible paths. The parent with the lowest cost function is selected. The cells covering the best path of the selected parent are transmitted to the child.The rest of the cells is filled with a a probability equal to 50% to be taken from one parent or another.

After applying the crossover operator on all parents two by two, it is possible to have more children than needed since I try all combinations. In this case, I select the best ones, i.e. the ones with the lowest cost function. This selection ensures having the same number of individuals for all populations.

The procedure `mutation` is responsible for changing one gene of each child. I predefine a *mutationRate* which is a value between 0 and 1. For each child, I generate a random value λ between 0 and 1. If λ is lower than *mutationRate*, I apply the mutation operator by changing a random cell that was corresponding to a RRO to an obstacle-free cell. Afterwards, the children are added to the next population.

This process is repeated until reaching the predefined $numberOfPop$ corresponding to the number of populations. Afterwards, the best individual is selected by using the function `selectBestInd`.

3.5 Results

This section reports and discusses computational results obtained by applying the method described in the previous section. Before displaying the results of our implementation, I start by defining a baseline with which we will compare the obtained results.

3.5.1 Baseline

As a baseline, I chose **A* with a modified heuristic**. Traditional A* algorithm has proven itself to be very efficient to resolve path planning problems in static environments with a low density of obstacles. In the use case, there are constraints such as high obstacle density (both physical and regulatory-related) and also the drone coverage range. Under such constraints, traditional A* algorithm is not vert suitable. It either doesn't find any obstacle that avoid all obstacles, or it returns a path that is very long and that exceeds the drone coverage distance. In order to overcome these challenges, I modify the heuristic of A* algorithm. For physical obstacles, I don't change anything compared to traditional A*. On the other hand, I allow the searching algorithm to consider cells that represents RRO but with a certain cost so that it is possible to explore zones representing RRO.

3.5.2 Computational results

For computational results, it is important to distinguish different cases. I make the choice of associating each case with a different representative map. Afterwards, I run simulations on each map with both the baseline and the defined hybrid genetic algorithm. The comparison between the two approaches is made afterwards. With regard to the size of the map, I arbitrarily choose the size 10*10 because my goal is to show how efficient is the proposed approach to find solutions compared to

the baseline.

Concerning the computation of the paths length, I consider a simple method which is the following: the distance between two adjacent cells (both horizontally and vertically) is considered to be equal to 1, and the distance between two diagonally aligned cells is equal to $\sqrt{2}$.

For all configurations, I define the cell (0,0) tho be the start and the cell (9,9) to be the destination. The obstacles change for each configuration (both PO and RRO). The goal is to find the best path from the start to the destination while avoiding all the PO. The best path should contain as few as possible RRO and its length should be less than the drone coverage range. It is evident that the best path is not always the shortest.

For the HGA, we choose the following parameters as showed in 1:

- number of generations = 100
- number of individuals per generation = 8
- number of parents = 4
- number of children = number of individuals per generation - number of parents = 4
- mutation rate = 0.4

3.5.2.1 First configuration

The first configuration that I consider is when there is an optimal solution which is feasible with the drone's autonomy. Basically, the optimal solution is the shortest path that does not include any obstacle. This configuration is very basic and all existing path planning algorithms should be able to provide a solution for it. As illustrated in Figure 3.4, the solution provided by the baseline is identical to the one provided by our hybrid genetic algorithm.

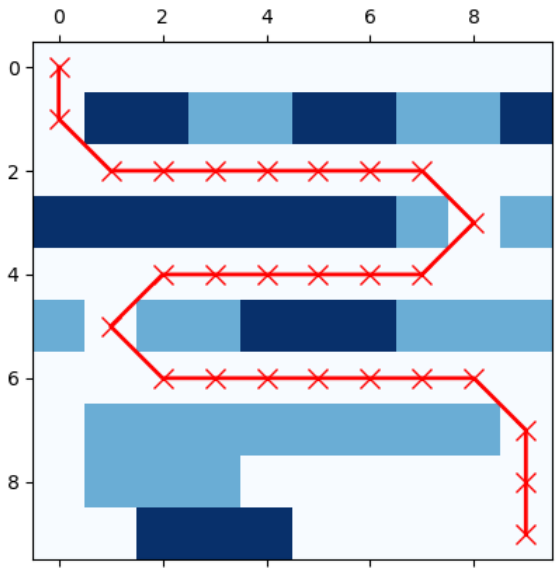


Figure 3.4: Optimal solution provided by both A* and HGA for first configuration

In this case, it is assumed that the drone has an autonomy that allows it to cover a distance equal to 30 units. The optimal path showed in Figure 3.4 has a length that is equal to 28 which is inferior to the drone's covering range.

3.5.2.2 Second configuration

The second configuration that I consider is when the optimal solution is not feasible with the drone's autonomy. I consider the same obstacles as in the first configuration but with a lower drone autonomy. In this case, I consider that the drone has an autonomy that allows it to cover a distance equal to 20. The solution found in the previous configuration becomes not feasible since its length is superior to 20. When running A* on this configuration, the algorithm is blocked and does not provide any solution. However, the HGA provides a solution as showed in Figure 3.5. The provided path has a length equal to 16.24 units.

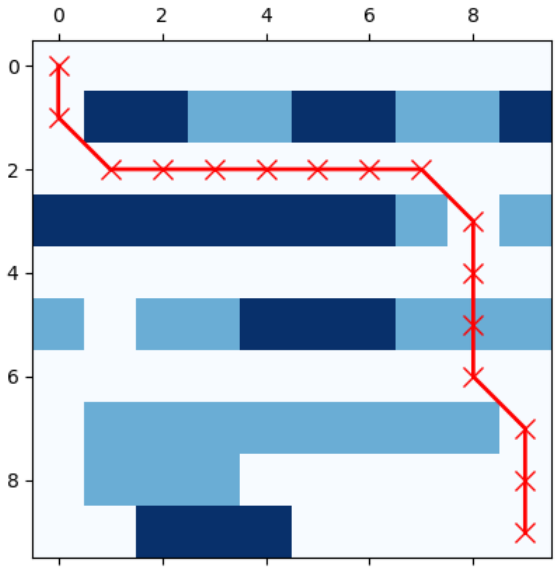


Figure 3.5: Solution provided by HGA for second configuration

3.5.2.3 Third configuration

The third configuration is when there isn't any obstacle-free path that connects the start and the destination. This case is very broad and contains a variety of sub-cases. We consider a configuration so that it is impossible to connect the start and the destination without encountering any obstacle as showed in Figure 3.6. When running A* on this configuration, the algorithm is blocked and does not provide any solution. However, the HGA provides a solution that crosses the minimal number of RRO in order to reach the destination which is 3 in this configuration.

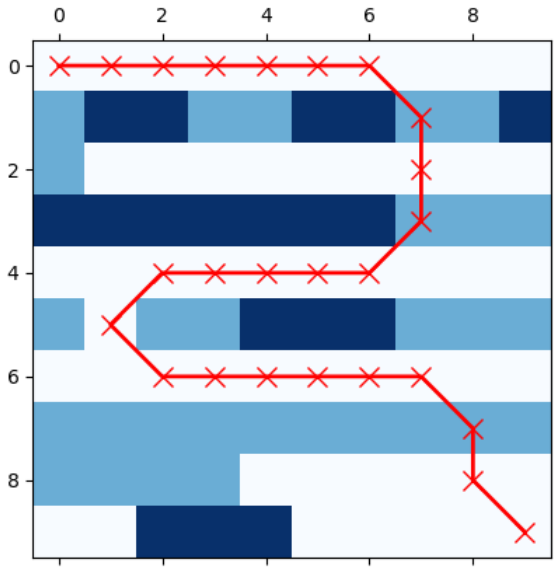


Figure 3.6: Solution provided by HGA for third configuration

3.6 Implementation and Feedback

This section is dedicated to the implementation and my personal feedback about the work presented in this chapter involving autonomous navigation for ACPS. I also provide several perspectives that can be considered for future works using the method that I propose.

3.6.1 Implementation

During the implementation phase, I used Python tool for coding the hybrid genetic algorithm. The implementation is totally configurable. The parameters number of generations, number of individuals per generation, number of parents, number of children and mutation rate can be adjusted for each execution.

The results presented earlier were displayed on maps with a size equal to 10*10. It is important to mention that I tried to run the algorithm on maps with a bigger size (100*100 and 500*500) and I have noticed that the computation time is not impacted. It remains inferior to 1 second with the hardware that I used which is an i7 Intel Core processor with a 2.8 GHz frequency.

3.6.2 Feedback

The work presented in this chapter is based on proposing a method for autonomous navigation of ACPS in an environment with a high density of obstacles. The considered use case involves the civilian use of drones for emergency situations in urban environment. The contribution presented in this chapter is based on strong hypotheses which are: defining a constant drone altitude during navigation phase, knowing the environment map before the take-off (offline path planning) and also not considering a dynamic environment in which new obstacles can be added during the flight. These hypotheses were followed in order to reduce the complexity of the problem.

A first perspective to the existing work is considering a 3D map in which the drones can change its altitude during the navigation phase. This would add complexity to the path planning process and it would be interesting to test if the proposed hybrid genetic algorithm is suitable in this case. A second perspective to the existing work is to consider path planning in dynamic environment. As the drone is navigating to its destination, its sensors collect data that can be processed and considered during the path planning process such as dynamic obstacles. Considering this data and processing would be an interesting work of research to test the limit of the proposed method in dynamic environment. Another perspective is real-time simulation using Robot Operating System (ROS) for example. This simulation would test the proposed method under real-time constraints and verify if it is suitable to be used for online real-time path planning.

3.7 Conclusion

In this final chapter, I have addressed the problem of safety for autonomous navigation. I considered the use case of drone navigation in urban environment. I specified a cost function and a particular environment representation. I proposed a hybrid genetic algorithm in order to solve this problem. The obtained results have been illustrated by considering different map configurations. I also gave a personal feedback about the work presented in this chapter as well as several perspectives.

General conclusion

We are currently facing a growth in systems complexity, with increasingly advanced technologies. CPSs are subject to this technological evolution. The cyber part of CPS perform complex tasks to control sophisticated physical processes in environments, that are becoming more and more ambient, open and hazardous. In addition, CPS are required to be resilient to internal errors and disturbances, and able to recover with the minimum costs. Modeling such systems is difficult especially in critical contexts with regards to their hardware, software and networking architectures, and event unpredictability of their environments.

This PhD work is placed in this context. The main interest is the safety analysis of CPS and how to include it from the early stages of system design. In the first instance, I focused on CPS features in order to understand if existing safety analysis methodologies are adequate to ensure safe applications of CPS which are in most cases critical. I came to the conclusion that existing safety approaches used mainly for the development of embedded systems or software applications are not adequate for safety-critical CPS. The main reason is that high-integrity critical embedded systems can be decomposed to reduce their complexity and then analyzed. This decomposability is not easily applicable to CPS, which require an overall consideration of the system during safety analysis process. A capital example is the evolution in the automotive domain: from vehicles to autonomous vehicles.

To help the reasoning on methods and tools for safety of ACPS, I specified a use case based on autonomous drones. A first outcome of my analysis is the deter-

mination of safety features that needs to be taken into account in order ensure safe applications. I came to the conclusion that there are three key factors which are: the system, the human factor and the environment. There is a strong correlation between these factors and combining them is key to ensure safety. Moreover, I argue that the notion of resilience is more appropriate when dealing with ACPS. Therefore, I study a new methodology to analyze resilience in ACPS.

The first main contribution of this PhD is articulated around the proposition of a predictive methodology to analyze resilience of CPS. I considered the same use case which is the drone systems for civilian applications and more specifically a drone emergency rescue system. The proposed methodology exploits a contract-based approach [1], which allows defining resilience-related properties and reasoning (Safety III [6]) about both system hazards (Safety I) and wished behaviour (Safety II). More in detail, the proposed methodology is based on a distributed object-oriented component-based software architecture. The structure of an object-oriented component, from my viewpoint, is new compared to the CCM specifications [176] and other definitions [177], in which periodic jobs, data listeners, and references to component instances are implicit features. I dealt with endogenous and exogenous aspects of resilience of the case study at both design and verification levels. Endogenous resilience is reflected in the system capability of processing internal functional and timing faults, and resisting to cyber-attacks. Exogenous resilience relies on the system's ability to safely operate in its ambient environment.

To this effect, I have defined a formal methodology to predict the system's behavior by abstraction, and to verify its resilience properties. I used Uppaal networks of timed automata to model the distributed interoperability between subsystems, and to analyze its endogenous resilience under an abstract networking model. For the analysis exogenous resilience, I choose to model the system's behavior in its environment using 3D models, then I use a tool that I have developed during the PhD called Contract Analysis Tool (CAT) in order to verify contracts and safety properties over 3D models.

The second main contribution of this PhD is an investigation over the impact of genetic algorithms on safety of the autonomous navigation system of a drone. More specifically, I tried to tackle the problem of drone autonomous navigation

in urban environment where there is a high density of both physical obstacles (such as buildings) and regulatory-related no-fly areas (such as airports). To this purpose, I proposed a solution based on a hybrid genetic algorithm that takes into consideration the battery level of the drone. I compared the results of my approach with a baseline, given by a deterministic algorithm. The measures are produced by executing the two algorithms on the same data. Despite the strong and restrictive hypothesis, I can state that the use of genetic algorithms during path planning process is suitable to improve the safety level of the navigation system.

Perspectives

There are many perspectives to improve and continue the work that has been carried out during this thesis and presented in this manuscript. I see four main directions for future works. The first one is to develop the networking aspect. As mentioned in Section 1.2.4.3, networking is an important feature of CPS. The methodology that I have presented for the analysis of resilience in CPS is based on a simplistic representation of the network. Considering a more realistic network model would be convenient to analyze the resilience of the system and its capacity to withstand different aspects of networking errors such as interference and chancing network topology. This direction is confirmed by the increasing trend of several (industrial and research) approaches devote to trustworthiness guarantee. They are based on the analysis of the impact of security failures on safety properties of a system.

The second direction for future work would be to consider online safety analysis for CPS. Obviously, the work carried out in this thesis considers an offline behaviour of the system. My analysis is carried out over a pre-defined system behaviour and cannot answer the following question: what if the system does not behave at intended? An interesting perspective would be to implement online safety analysis (using contracts for example) to validate safety properties on a dynamic CPS.

The third direction for future work would be to extend the work carried out in this thesis to a fleet of drones. It would be interesting to consider a swarm of drones communicating with each other and with the ground control station.

The fourth direction for future work would be to validate the obtained results via a demonstrator and a prototype using Robot Operating System (ROS) for example.

Bibliography

[1] A. Sangiovanni-Vincentelli, W. Damm, and R. Passerone, "Taming dr. frankenstein: Contract-based design for cyber-physical systems," *European journal of control*, vol. 18, no. 3, pp. 217–238, 2012.

[2] Sandro D'Elia, "CPS in EU Programmes," 2017, european Commission DG CONNECT.

[3] R. L. Swiggett, "Printed circuit armature," Jan. 31 1961, uS Patent 2,970,238.

[4] IISO 8402:1994 Management de la qualité et assurance de la qualité — Vocabulaire, 1994.

[5] D. Cancila, E. Laarouchi, and A. Bagnato, "Dependability of the transport of the future," *ADA USER*, vol. 38, no. 4, p. 236, 2017.

[6] E. Hollnagel, D. D. Woods, and N. Leveson, *Resilience engineering: Concepts and precepts*. Ashgate Publishing, Ltd., 2006.

[7] E. Laarouchi, D. Cancila, and H. Chaouchi, "Safety and degraded mode in civilian applications of unmanned aerial systems," in *2017 IEEE/AIAA 36th Digital Avionics Systems Conference (DASC)*. IEEE, 2017, pp. 1–7.

[8] S. Mouelhi, M.-E. Laarouchi, D. Cancila, and H. Chaouchi, "Predictive formal analysis of resilience in cyber-physical systems," *IEEE Access*, vol. 7, pp. 33 741–33 758, 2019.

[9] l. s. H. C. Emine Laarouchi, Daniela cancila, "Genetic algorithms based acps safety," in *Proceedings of the Cyber-Physical Systems (ICPPS)*, 2020.

[10] A. Avizienis, J.-C. Laprie, B. Randell *et al.*, *Fundamental concepts of dependability*. University of Newcastle upon Tyne, Computing Science, 2001.

[11] T. Frühwirth, L. Krammer, and W. Kastner, "Dependability demands and state of the art in the internet of things," in *2015 IEEE 20th Conference on Emerging Technologies & Factory Automation (ETFA)*. IEEE, 2015, pp. 1–4.

[12] S. Mouelhi, D. Cancila, and A. Ramdane-Cherif, "Distributed object-oriented design of autonomous control systems for connected vehicle platoons," in *Proc. of 22nd Int. Conf. on Engineering of Complex Computer Systems (ICECCS)*. IEEE, 2017, pp. 40–49.

[13] "CPSWarm European Project," https://www.cpswarm.eu/, online; accessed 03 April 2019.

[14] R. Passerone, D. Cancila, M. Albano, S. Mouelhi, S. Plosz, E. Jantunen, A. Ryabokon, E. Laarouchi, C. Hegedűs, and P. Varga, "A methodology for the design of safety-compliant and secure communication of autonomous vehicles," *Ieee Access*, vol. 7, pp. 125 022–125 037, 2019.

[15] D. H. Kim, *Introduction to systems thinking*. Pegasus Communications Waltham, MA, 1999, vol. 16.

[16] María Victoria Cengarle, Saddek Bensalem, John McDermid, Roberto Passerone, Alberto Sangiovanni-Vincentelli, Martin Törngren, "Characteristics, capabilities, potential applications of Cyber-Physical Systems: a preliminary analysis. Deliverable, CyPhERS : Cyber-Physical European Roadmap & Strategy," http://www.cyphers.eu/sites/default/files/D2.1.pdf, 2013, online; accessed: 05 April 2019.

[17] E. A. Lee and S. A. Seshia, *Introduction to embedded systems: A cyber-physical systems approach*. Mit Press, 2016.

[18] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad hoc networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[19] C. Greer, M. Burns, D. Wollman, and E. Griffor, "Cyber-physical systems and internet of things," *NIST Special Publication*, vol. 202, no. 2019, p. 52, 1900.

[20] C. E. Tuncali, H. Ito, J. Kapinski, and J. V. Deshmukh, "Reasoning about safety of learning-enabled components in autonomous cyber-physical systems," in *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 2018, pp. 1–6.

[21] CyPhERS FP7 Project. Cyber-Physical European Roadmap and Strategy, http://cyphers.eu/, online; accessed: 05 April 2019.

[22] "Platform4CPS European Project," https://www.platforms4cps.eu/, online; accessed 30 April 2019.

[23] H. Thompson, M. Reimann, D. Ramos-Hernandez, S. Bageritz, A. Brunet, C. Robinson, B. Sautter, J. Linzbach, H. Pfeifer, V. Aravantinos *et al.*, *Platforms4CPS, Key Outcomes and Recommendations*. Steinbeis-Edition, 2018.

[24] H.-M. Huang, K. Pavek, B. Novak, J. Albus, and E. Messin, "A framework for autonomy levels for unmanned systems (alfus)," *Proceedings of the AUVSI's Unmanned Systems North America*, pp. 849–863, 2005.

[25] J. M. Beer, A. D. Fisk, and W. A. Rogers, "Toward a framework for levels of robot autonomy in human-robot interaction," *Journal of human-robot interaction*, vol. 3, no. 2, pp. 74–99, 2014.

[26] ISO 31000:2018, Risk Management, 2018.

[27] C. Tannert, H.-D. Elvers, and B. Jandrig, "The ethics of uncertainty: In the light of possible dangers, research becomes a moral duty," *EMBO reports*, vol. 8, no. 10, pp. 892–896, 2007.

[28] M. H. Mishel, "Uncertainty in illness," *Image: The Journal of Nursing Scholarship*, vol. 20, no. 4, pp. 225–232, 1988.

[29] R. Rajkumar, I. Lee, L. Sha, and J. Stankovic, "Cyber-physical systems: the next computing revolution," in *Design Automation Conference*. IEEE, 2010, pp. 731–736.

[30] M. Conti, S. K. Das, C. Bisdikian, M. Kumar, L. M. Ni, A. Passarella, G. Roussos, G. Tröster, G. Tsudik, and F. Zambonelli, "Looking ahead in pervasive computing: Challenges and opportunities in the era of cyber–physical convergence," *Pervasive and Mobile Computing*, vol. 8, no. 1, pp. 2–21, 2012.

[31] D. Garlan, "Software engineering in an uncertain world," in *Proceedings of the FSE/SDP workshop on Future of software engineering research*. ACM, 2010, pp. 125–128.

[32] S. Ali, H. Lu, S. Wang, T. Yue, and M. Zhang, "Uncertainty-wise testing of cyber-physical systems," in *Advances in Computers*. Elsevier, 2017, vol. 107, pp. 23–94.

[33] M. Zhang, B. Selic, S. Ali, T. Yue, O. Okariz, and R. Norgren, "Understanding uncertainty in cyber-physical systems: a conceptual model," in *European conference on modelling foundations and applications*. Springer, 2016, pp. 247–264.

[34] M. Anand, E. Cronin, M. Sherr, M. Blaze, Z. Ives, and I. Lee, "Security challenges in next generation cyber physical systems," *Beyond SCADA: Networked Embedded Control for Cyber Physical Systems*, vol. 41, 2006.

[35] A. A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *2008 The 28th International Conference on Distributed Computing Systems Workshops*. IEEE, 2008, pp. 495–500.

[36] A. Cardenas, S. Amin, B. Sinopoli, A. Giani, A. Perrig, S. Sastry *et al.*, "Challenges for securing cyber physical systems," in *Workshop on future directions in cyber-physical systems security*, vol. 5, no. 1, 2009.

[37] P. Barsocchi, S. Chessa, I. Martinovic, and G. Oligeri, "A cyber-physical approach to secret key generation in smart environments," *Journal of Ambient Intelligence and Humanized Computing*, vol. 4, no. 1, pp. 1–16, 2013.

[38] M. Törngren and P. Grogan, "How to deal with the complexity of future cyber-physical systems?" *Designs*, vol. 2, no. 4, p. 40, 2018.

[39] D. E. Whitney, "Why mechanical design cannot be like vlsi design," *Research in Engineering Design*, vol. 8, no. 3, pp. 125–138, Sep 1996, online; accessed 19 April 2019. [Online]. Available: https://doi.org/10.1007/BF01608348

[40] A. Wavering, "Foundation for innovation: Strategic r&d opportunities for 21th century cyber-physical systems," *National Instution for Standardisation and Technology, Report January*, 2013.

[41] B. Saddek, C. María Victoria, P. Roberto, S.-V. Alberto, and T. Martin, CPS Technologies. Deliverable D4.2 of the CyPhERS FP7 project, September 2014.

[42] H. Kagermann, W. Wahlster, and J. Helbig, "Securing the future of german manufacturing industry: Recommendations for implementing the strategic initiative industrie 4.0," *Final report of the Industrie*, vol. 4, no. 0, 2013.

[43] H. Koziolek, R. Weiss, Z. Durdik, J. Stammel, and K. Krogmann, "Towards software sustainability guidelines for long-living industrial systems," *Software Engineering 2011–Workshopband*, 2011.

[44] H. S. Kang, J. Y. Lee, S. Choi, H. Kim, J. H. Park, J. Y. Son, B. H. Kim, and S. Do Noh, "Smart manufacturing: Past research, present findings, and future directions," *International Journal of Precision Engineering and Manufacturing-Green Technology*, vol. 3, no. 1, pp. 111–128, 2016.

[45] E. F. of the Future Research Association *et al.*, "Factories of the future: Multi-annual roadmap for the contractual ppp under horizon 2020," *Publications office of the European Union: Brussels, Belgium*, 2013.

[46] J. Manyika, M. Chui, J. Bughin, R. Dobbs, P. Bisson, and A. Marrs, *Disruptive technologies: Advances that will transform life, business, and the global economy*. McKinsey Global Institute San Francisco, CA, 2013, vol. 180.

[47] European Road Transport Research Advisory Council (ERTRAC). Automated Driving Roadmap, 2015.

[48] D. Watzenig and M. Horn, *Automated driving: safer and more efficient future driving*. Springer, 2016.

[49] "Deserve : Development Platform for Safe and Efficient Drive. European Project," http://www.deserve-project.eu/, online; accessed 30 April 2019.

[50] F. Pasqualetti, F. Dörfler, and F. Bullo, "Attack detection and identification in cyber-physical systems," *IEEE transactions on automatic control*, vol. 58, no. 11, pp. 2715–2729, 2013.

[51] S. Gisdakis, M. Laganà, T. Giannetsos, and P. Papadimitratos, "Serosa: Service oriented security architecture for vehicular communications," in *2013 IEEE Vehicular Networking Conference*. IEEE, 2013, pp. 111–118.

[52] K. Walia, "Utility drone market size & share report till 2025," 2019.

[53] D. Joshi, "Commercial unmanned aerial vehicle (uav) market analysis industry trends, companies and what you should know," *Business Insider*, 2017.

[54] A. Levin, "Google Spinoff's Drone Delivery Business First to Get FAA Approval," https://www.bloomberg.com/news/articles/2019-04-23/alphabet-s-drone-delivery-business-cleared-for-takeoff-by-faa, 2019, online; accessed 30 April 2019.

[55] Damien Licata Caruso, "Notre-Dame : comment des drones ont aidé les pompiers contre l'incendie," http://www.leparisien.fr, 2019, online; accessed: 30 April 2019.

[56] N. J. Bahr, *System safety engineering and risk assessment: a practical approach*. CRC press, 2018.

[57] A. K. Jardine and A. H. Tsang, *Maintenance, replacement, and reliability: theory and applications*. CRC press, 2005.

[58] H. Watson *et al.*, "Launch control safety study," *Bell labs*, 1961.

[59] O. Gaudoin, J. Ledoux *et al.*, *Modélisation aléatoire en fiabilité des logiciels*. Hermès Science, 2007.

[60] BBC News, "Unmanned Aircraft System (UAS) Traffic Management (UTM)," https://utm.arc.nasa.gov/index.shtml, online; accessed: 10 May 2019.

[61] CENELEC 50126. Railway applications - The Specification and Demonstration of Reliability, Availability, Maintainability and Safety (RAMS) - Part 2: Systems approach to safety. CENELEC standard, 2012.

[62] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE transactions on dependable and secure computing*, vol. 1, no. 1, pp. 11–33, 2004.

[63] J. H. Ziegeldorf, O. G. Morchon, and K. Wehrle, "Privacy in the internet of things: threats and challenges," *Security and Communication Networks*, vol. 7, no. 12, pp. 2728–2742, 2014.

[64] J. Guiochet, M. Machin, and H. Waeselynck, "Safety-critical advanced robots: A survey," *Robotics and Autonomous Systems*, vol. 94, pp. 43–52, 2017.

[65] Walid Taha, "Lecture Notes on Cyber-Physical Systems," http://bit.ly/LNCPS-2018, 2018, online; accessed: 24 April 2019.

[66] I. E. Commission *et al.*, "Functional safety of electrical/electronic/programmable electronic safety related systems," *IEC 61508*, 2000.

[67] IEC 60880. Nuclear power plants - Instrumentation and control systems important to safety - Software aspects for computer-based systems performing category A functions. IEC standard.

[68] IEC 61513. Nuclear power plants - Instrumentation and control systems important to safety - General requirements for systems. IEC standard.

[69] IEC 62138. Nuclear power plants - Instrumentation and control systems important for safety - Software aspects for computer-based systems performing category B or C functions. IEC standard.

[70] CENELEC 50128. Railway applications - Communications, signaling and processing systems - Software for railway control and protection systems. CENELEC standard.

[71] CENELEC 50129. Railway applications - Communications, signaling and processing systems - Safety related electronic systems. CENELEC standard.

[72] ISO 26262. Road vehicles - Functional safety. ISO standard.

[73] R. Chapman, "Correctness by construction: a manifesto for high integrity software," in *Proceedings of the 10th Australian workshop on Safety critical systems and software-Volume 55*. Australian Computer Society, Inc., 2006, pp. 43–46.

[74] J. Sifakis, "Embedded systems-challenges and work directions," *Lecture notes in computer science*, vol. 3544, p. 184, 2005.

[75] SAE. Architecture Analysis and Design Language (AADL). http://www.aadl.info/aadl/currentsite/, online; accessed: 24 April 2019.

[76] The GSN Working Group Online. Goal Structuring Notation (GSN). https://es-static.fbk.eu/tools/ocra/index.php?n=Main.HomePage, online; accessed: 24 April 2019.

[77] C. S. Holling, "Resilience and stability of ecological systems," *Annual review of ecology and systematics*, vol. 4, no. 1, pp. 1–23, 1973.

[78] P. L. Engle, S. Castle, and P. Menon, "Child development: Vulnerability and resilience," *Social science & medicine*, vol. 43, no. 5, pp. 621–635, 1996.

[79] G. Hamel and L. Valikangas, "The quest for resilience," *Revista Icade. Revista de las Facultades de Derecho y Ciencias Económicas y Empresariales*, no. 62, pp. 355–358, 2004.

[80] J.-C. Laprie, "From dependability to resilience," in *38th IEEE/IFIP Int. Conf. On dependable systems and networks*, 2008, pp. G8–G9.

[81] E. Hollnagel, "Rag-the resilience analysis grid," *Resilience engineering in practice: a guidebook. Ashgate Publishing Limited, Farnham, Surrey*, pp. 275–296, 2011.

[82] V. P. GALOTTI, A. K. RAO, and D. E. MAURINO, "Icao initiative prompts global approach to sms implementation," *ICAO Journal*, vol. 61, no. 6, 2006.

[83] E. Hollnagel, *Safety-I and safety-II: the past and future of safety management*. CRC Press, 2018.

[84] L. Sha, S. Gopalakrishnan, X. Liu, and Q. Wang, "Cyber-physical systems: A new frontier," in *2008 IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (sutc 2008)*. IEEE, 2008, pp. 1–9.

[85] P. Bagade, A. Banerjee, and S. K. Gupta, "Validation, verification, and formal methods for cyber-physical systems," in *Cyber-Physical Systems*. Elsevier, 2017, pp. 175–191.

[86] T. A. Henzinger, "The theory of hybrid automata," in *Verification of Digital and Hybrid Systems*. Springer, 2000, pp. 265–292.

[87] R. Alur and D. L. Dill, "A theory of timed automata," *Theoretical computer science*, vol. 126, no. 2, pp. 183–235, 1994.

[88] F. Alkhateeb, E. Al Maghayreh, and I. A. Doush, *Multi-agent systems-Modeling, control, programming, simulations and applications*. InTech, 2011.

[89] V. Gunes, S. Peter, T. Givargis, and F. Vahid, "A survey on concepts, applications, and challenges in cyber-physical systems." *KSII Transactions on Internet & Information Systems*, vol. 8, no. 12, 2014.

[90] Aliyuda, "Towards the design of cyber-physical system via multiagent system technology," *International Journal of Scientific & Engineering Research*, vol. 7, no. 10, pp. 155–161, 2016.

[91] J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald, "Formal methods: Practice and experience," *ACM computing surveys (CSUR)*, vol. 41, no. 4, p. 19, 2009.

[92] E. M. Clarke and P. Zuliani, "Statistical model checking for cyber-physical systems," in *International Symposium on Automated Technology for Verification and Analysis*. Springer, 2011, pp. 1–12.

[93] E. M. Clarke, W. Klieber, M. Nováček, and P. Zuliani, "Model checking and the state explosion problem," in *LASER Summer School on Software Engineering*. Springer, 2011, pp. 1–30.

[94] A. Platzer and J.-D. Quesel, "Keymaera: A hybrid theorem prover for hybrid systems (system description)," in *International Joint Conference on Automated Reasoning*. Springer, 2008, pp. 171–178.

[95] B. Li, Z. Sun, K. Mechitov, G. Hackmann, C. Lu, S. J. Dyke, G. Agha, and B. F. Spencer, "Realistic case studies of wireless structural control," in *2013 ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2013, pp. 179–188.

[96] A. Cervin, D. Henriksson, B. Lincoln, J. Eker, and K.-E. Arzen, "How does control timing affect performance? analysis and simulation of timing using jitterbug and truetime," *IEEE control systems magazine*, vol. 23, no. 3, pp. 16–30, 2003.

[97] P. Finnegan, *World Civil Unmanned Aerial Systems Market Profile & Forecast*. Fairfax, VA, USA: Teal Group Corporation, 2019.

[98] R. L. Finn and D. Wright, "Privacy, data protection and ethics for civil drone practice: A survey of industry, regulators and civil society organisations," *Computer Law & Security Review*, vol. 32, no. 4, pp. 577–586, 2016.

[99] C. Bolkcom and B. Nuñez-Neto, "Homeland security: Unmanned aerial vehicles and border surveillance." LIBRARY OF CONGRESS WASHINGTON DC CONGRESSIONAL RESEARCH SERVICE, 2008.

[100] R. Clarke, "Understanding the drone epidemic," *Computer Law & Security Review*, vol. 30, no. 3, pp. 230–246, 2014.

[101] F. De Florio, *Airworthiness: An introduction to aircraft certification and operations*. Butterworth-Heinemann, 2016.

[102] M. Oncu and S. Yildiz, "An analysis of human causal factors in unmanned aerial vehicle (uav) accidents," NAVAL POSTGRADUATE SCHOOL MONTEREY CA GRADUATE SCHOOL OF BUSINESS AND PUBLIC ..., Tech. Rep., 2014.

[103] "Regulatory framework for drone operations in Europe," https://www.easa.europa.eu/easa-and-you/civil-drones-rpas, online; accessed 08 May 2019.

[104] M. W. Mueller and R. D'Andrea, "Stability and control of a quadrocopter despite the complete loss of one, two, or three propellers," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 45–52.

[105] F. Kendoul, "Towards a unified framework for uas autonomy and technology readiness assessment (atra)," in *Autonomous Control Systems and Vehicles*. Springer, 2013, pp. 55–71.

[106] T. B. Sheridan, *Telerobotics, automation, and human supervisory control*. MIT press, 1992.

[107] R. W. Proud, J. J. Hart, and R. B. Mrozinski, "Methods for determining the level of autonomy to design into a human spaceflight vehicle: a function specific approach," NATIONAL AERONAUTICS AND SPACE ADMINISTRATION HOUSTON TX LYNDON B JOHNSON ..., Tech. Rep., 2003.

[108] B. Clough, "Metrics, schmetrics! how do you determine a uav's autonomy anyway," in *Proceedings of the 2002 PerMIS Workshop*, 2002.

[109] D. Gettinger and A. H. Michel, "Drone sightings and close encounters: An analysis," *Center for the Study of the Drone, Bard College*, 2015.

[110] Y. Song, B. Horton, and J. Bayandor, "Investigation of uas ingestion into high-bypass engines, part 1: Bird vs. drone," in *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2017, p. 0186.

[111] K. Schroeder, Y. Song, B. Horton, and J. Bayandor, "Investigation of uas ingestion into high-bypass engines, part 2: Parametric drone study," in *58th AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, 2017, p. 0187.

[112] "DJI Introduces New Geofencing System for its Drones," https://www.dji.com/newsroom/news/dji-fly-safe-system, 2015, online; accessed 10 May 2019.

[113] Z. Liu, R. Sengupta, and A. Foina, "Quadrotors in smart cities avoiding helicopters," in *Proc. of American Control Conference, Boston, USA*, 2016.

[114] "Unmanned Aircraft System (UAS) Traffic Management (UTM)," https://utm.arc.nasa.gov/index.shtml, online; accessed 14 May 2019.

[115] T. Martin, B. Saddek, C. María Victoria, C. De-Jiu, M. John, P. Roberto, S.-V. Alberto, and R. Thomas, CPS: State of the Art. Deliverable D5.1 of the CyPhERS FP7 project, March 2014.

[116] D. Cancila, R. Passerone, T. Vardanega, and M. Panunzio, "Toward correctness in the specification and handling of non-functional attributes of high-integrity real-time embedded systems," *IEEE Transactions on Industrial Informatics*, vol. 6, no. 2, pp. 181–194, 2010.

[117] L. De Alfaro and T. A. Henzinger, "Interface automata," in *ACM SIGSOFT Software Engineering Notes*, vol. 26, no. 5. ACM, 2001, pp. 109–120.

[118] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.

[119] P. H. Feiler, D. P. Gluch, and J. J. Hudak, "The architecture analysis & design language (aadl): An introduction," Carnegie-Mellon Univ Pittsburgh PA Software Engineering Inst, Tech. Rep., 2006.

[120] M. Tiller, *Introduction to physical modeling with Modelica*. Springer Science & Business Media, 2012, vol. 615.

[121] S. T. Karris, *Introduction to Simulink with engineering applications*. Orchard Publications, 2006.

[122] P. A. Abdulla, J. Deneux, G. Stålmarck, H. Ågren, and O. Åkerlund, "Designing safe, reliable systems using scade," in *International Symposium On Leveraging Applications of Formal Methods, Verification and Validation*. Springer, 2004, pp. 115–129.

[123] S. Bernardi, J. Merseguer, and D. C. Petriu, "A dependability profile within marte," *Software & Systems Modeling*, vol. 10, no. 3, pp. 313–336, 2011.

[124] P. Nuzzo, J. B. Finn, A. Iannopollo, and A. L. Sangiovanni-Vincentelli, "Contract-based design of control protocols for safety-critical cyber-physical systems," in *2014 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2014, pp. 1–4.

[125] A. Benveniste, B. Caillaud, A. Ferrari, L. Mangeruca, R. Passerone, and C. Sofronis, "Multiple viewpoint contract-based specification and design," in *International Symposium on Formal Methods for Components and Objects*. Springer, 2007, pp. 200–225.

[126] E. M. Clarke, "The birth of model checking," in *25 Years of Model Checking*. Springer, 2008, pp. 1–26.

[127] P. Nuzzo, M. Lora, Y. A. Feldman, and A. L. Sangiovanni-Vincentelli, "Chase: Contract-based requirement engineering for cyber-physical system design," in *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2018, pp. 839–844.

[128] G. Behrmann, A. David, and K. G. Larsen, "A tutorial on UPPAAL," in *Proc. of 4th Int. Schl. on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM-RT*, ser. LNCS, no. 3185. Springer, Berlin, Heidelberg, 2004, pp. 200–236.

[129] A. Claesson, D. Fredman, L. Svensson, M. Ringh, J. Hollenberg, P. Nordberg, M. Rosenqvist, T. Djarv, S. Österberg, J. Lennartsson *et al.*, "Unmanned Aerial Vehicles (drones) in out-of-hospital-cardiac-arrest," *Scandinavian journal of trauma, resuscitation and emergency medicine*, vol. 24, no. 1, p. 124, 2016.

[130] A. Bagnato, R. K. Bíró, D. Bonino, C. Pastrone, W. Elmenreich, R. Reiners, M. Schranz, and E. Arnautovic, "Designing swarms of cyber-physical systems: the h2020 cpswarm project," in *Proc. of the Computing Frontiers Conf.* ACM, 2017, pp. 305–312.

[131] D. Ratasich, O. Höftberger, H. Isakovic, M. Shafique, and R. Grosu, "A self-healing framework for building resilient cyber-physical systems," in *Proc. of 20th IEEE Symp. on Real-Time Distributed Computing (ISORC)*. IEEE, 2017, pp. 133–140.

[132] Q. Zhu and T. Başar, "Robust and resilient control design for cyber-physical systems with an application to power systems," in *Proc. of 50th IEEE Conf. on Decision and Control and European Control Conf. (CDC-ECC)*. IEEE, 2011, pp. 4066–4071.

[133] G. Denker, N. Dutt, S. Mehrotra, M.-O. Stehr, C. Talcott, and N. Venkatasubramanian, "Resilient dependable cyber-physical systems: a middleware perspective," *Journal of Internet Services and Applications*, vol. 3, no. 1, pp. 41–49, 2012.

[134] I. Jovanov and M. Pajic, "Relaxing integrity requirements for attack-resilient cyber-physical systems," *arXiv preprint arXiv:1707.02950*, 2017.

[135] A. Cardenas, S. Amin, and S. Sastry, "Secure control: Towards survivable cyber-physical systems," in *Proc. of 28th Int. Conf. on Distributed Computing Systems (workshops) ICDCS*. IEEE, 2008, pp. 495–500.

[136] B. Vogel-Heuser, C. Diedrich, D. Pantförder, and P. Göhner, "Coupling heterogeneous production systems by a multi-agent based cyber-physical production system," in *Proc. of 12th IEEE Int. Conf. Industrial Informatics (INDIN'14)*. IEEE, 2014, pp. 713–719.

[137] M. Lezoche and H. Panetto, "Cyber-physical systems, a new formal paradigm to model redundancy and resiliency," *Enterprise Information Systems*, vol. 0, no. 0, pp. 1–22, 2018.

[138] M. Lezoche, E. Yahia, A. Aubry, H. Panetto, and M. Zdravković, "Conceptualising and structuring semantics in cooperative enterprise information systems models," *Computers in Industry*, vol. 63, no. 8, pp. 775–787, 2012.

[139] B. Meyer, "Applying Design by Contract," *Computer*, vol. 25, no. 10, pp. 40–51, 1992.

[140] C. A. R. Hoare, "An axiomatic basis for computer programming," *Communications of the ACM*, vol. 12, no. 10, pp. 576–580, 1969.

[141] O. Ferrante, R. Passerone, A. Ferrari, L. Mangeruca, and C. Sofronis, "BCL: A compositional contract language for embedded systems," in *Proc. of the IEEE Int. Conf. on Emerging Technology and Factory Automation (ETFA)*. IEEE, 2014, pp. 1–6.

[142] AdaCore, "High-integrity object-oriented programming in Ada," 2016, v1.4.

[143] ——, "SPARK 2014 Reference manual," 2014.

[144] AdaCore, "PolyORB User's guide," 2003.

[145] Y. Zeng, R. Zhang, and T. J. Lim, "Wireless communications with Unmanned Aerial Vehicles: opportunities and challenges," *IEEE Communications Magazine*, vol. 54, no. 5, pp. 36–42, 2016.

[146] IEEE Standards Association, "802.11p: Amendment 6 to the IEEE 802.11 for Wireless Access in Vehicular Environments (WAVE)," 2010, Standard.

[147] European Telecommunications Standards Institute, "EN 302 663: Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band," 2012, Standard.

[148] D. Cancila, H. Zaatiti, and R. Passerone, "Cyber-physical system and contract-based design: A three dimensional view," in *Proceedings of the WESE'15: Workshop on Embedded and Cyber-Physical Systems Education*. ACM, 2015, p. 4.

[149] Z. Li, Z. Li, R. Xu, M. Li, J. Li, Y. Liu, D. Sui, W. Zhang, and Z. Chen, "Three-dimensional printing models improve understanding of spinal fracture—a randomized controlled study in china," *Scientific reports*, vol. 5, p. 11570, 2015.

[150] L. Hobert, "A study on platoon formations and reliable communication in vehicle platoons," Master's thesis, University of Twente, 2012.

[151] G. Marsden, M. McDonald, and M. Brackstone, "Towards an understanding of adaptive cruise control," *Transportation Research Part C: Emerging Technologies*, vol. 9, no. 1, pp. 33–51, 2001.

[152] R. Rajamani, H.-S. Tan, B. K. Law, and W.-B. Zhang, "Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons," *IEEE Transactions on Control Systems Technology*, vol. 8, no. 4, pp. 695–708, 2000.

[153] J. Bengtsson and W. Yi, "Timed automata: Semantics, algorithms and tools," in *Advanced Course on Petri Nets*. Springer, 2003, pp. 87–124.

[154] R. Hess, *The essential Blender: guide to 3D creation with the open source suite Blender*. No Starch Press, 2007.

[155] A. Cimatti, M. Dorigatti, and S. Tonetta, "Ocra: A tool for checking the refinement of temporal contracts," in *2013 28th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2013, pp. 702–705.

[156] P. Nuzzo, A. L. Sangiovanni-Vincentelli, D. Bresolin, L. Geretti, and T. Villa, "A platform-based design methodology with contracts and related tools for the design of cyber-physical systems," *Proceedings of the IEEE*, vol. 103, no. 11, pp. 2104–2132, 2015.

[157] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[158] F. J. G. Carpio, "Two-staged local trajectory planning based on optimal pre-planned curves interpolation for human-like driving in urban areas," Ph.D. dissertation, 2018.

[159] F. Lucas, C. Guettier, P. Siarry, A.-M. Milcent, and A. De La Fortelle, "Constrained navigation with mandatory waypoints in uncertain environment," 2010.

[160] Y.-Q. Qin, D.-B. Sun, N. Li, and Y.-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in *Proceedings of 2004 international conference on machine learning and cybernetics (IEEE Cat. No. 04EX826)*, vol. 4. IEEE, 2004, pp. 2473–2478.

[161] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *International journal of physical sciences*, vol. 7, no. 9, pp. 1314–1320, 2012.

[162] G. Chen, L. Pan, Y. Chen, P. Xu, Z. Wang, P. Wu, J. Ji, and X. Chen, "Robot navigation with map-based deep reinforcement learning," *arXiv preprint arXiv:2002.04349*, 2020.

[163] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel *et al.*, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018.

[164] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale

data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[165] L. Xie, S. Wang, S. Rosa, A. Markham, and N. Trigoni, "Learning with training wheels: speeding up training with a simple controller for deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6276–6283.

[166] H. Zhang, J. Butzke, and M. Likhachev, "Combining global and local planning with guarantees on completeness," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 4500–4506.

[167] L. C. Wang, L. S. Yong, and M. H. Ang, "Hybrid of global path planning and local navigation implemented on a mobile robot in indoor environment," in *Proceedings of the IEEE Internatinal Symposium on Intelligent Control*. IEEE, 2002, pp. 821–826.

[168] Z. Bi, Y. Yimin, and Y. Wei, "Hierarchical path planning approach for mobile robot navigation under the dynamic environment," in *2008 6th IEEE International Conference on Industrial Informatics*. IEEE, 2008, pp. 372–376.

[169] F. M. de Chamisso, L. Soulier, and M. Aupetit, "Exploratory digraph navigation using A*," in *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[170] F. M. de Chamisso, D. Cancila, L. Soulier, R. Passerone, and M. Aupetit, "Lifelong Exploratory Navigation: an Architecture for Safer Mobile Robots," *IEEE Design and Test (accepted under publication)*, 2019.

[171] V. Roberge, M. Tarbouchi, and G. Labonté, "Comparison of parallel genetic algorithm and particle swarm optimization for real-time uav path planning," *IEEE Transactions on Industrial Informatics*, vol. 9, no. 1, pp. 132–141, 2013.

[172] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[173] M. Srinivas and L. M. Patnaik, "Genetic algorithms: A survey," *computer*, vol. 27, no. 6, pp. 17–26, 1994.

[174] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 5. IEEE, 2004, pp. 4350–4355.

[175] S. X. Yang, Y. Hu, and M. Q.-h. Meng, "A knowledge based ga for path planning of multiple mobile robots in dynamic environments," in *2006 IEEE Conference on Robotics, Automation and Mechatronics*. IEEE, 2006, pp. 1–6.

[176] International Organization for Standardization, "ISO/IEC 19500: Information technology – Object Management Group – Common Object Request Broker Architecture (CORBA)," 2003, Standard (rev. 2012).

[177] C. Szyperski, *Component Software: Beyond Object-Oriented Programming*, 2nd ed. Boston, MA, USA: Addison-Wesley Professional, 2002.