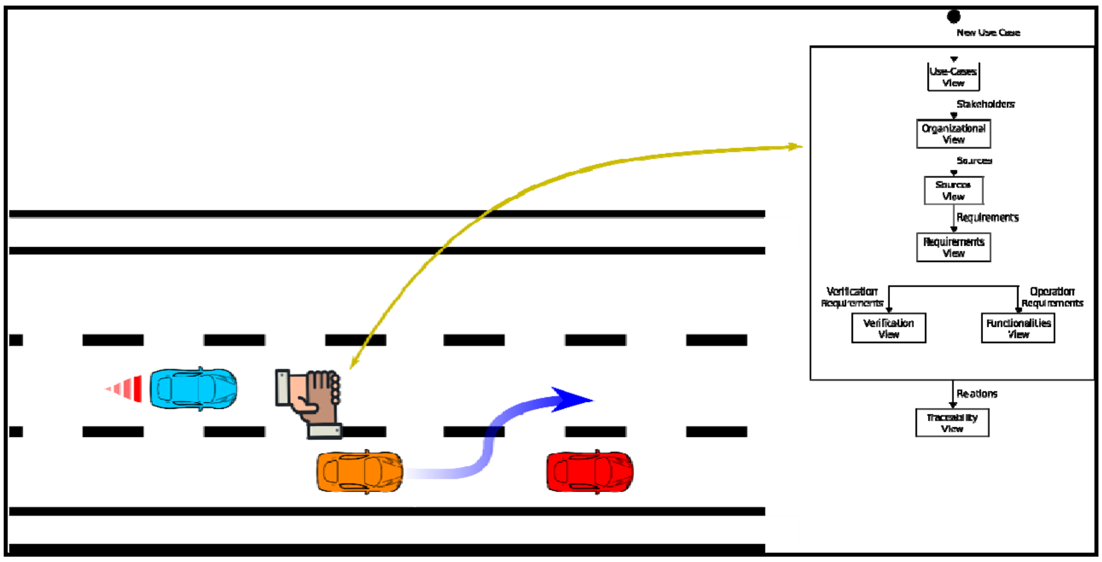
An overview on systems of systems control : general discussions and application to multiple autonomous vehicles

Mohamad Ali Assaad

Par **Mohamad Ali ASSAAD**

An overview on systems of systems control: general discussions and application to multiple autonomous vehicles

Thèse présentée
pour l'obtention du grade
de Docteur de l'UTC



Soutenue le 21 janvier 2019
Spécialité : Sciences et Technologies de l'Information et des Systèmes : Unité de recherche Heudyasic (UMR-7253)
D2466

An overview on Systems of Systems Control: General discussions and application to multiple autonomous vehicles

`Spécialité : Sciences et Technologies de l'Information et des Systèmes`

Mohamad Ali ASSAAD

PhD defense on January 21, 2019 in front of the jury composed of :

Université de technologie de Compiègne

Laboratoire Heudiasyc UMR CNRS 7253

Abstract

This thesis focuses on System of Systems (SoS) control, and how to build adaptable and reliable SoS. This work is part of the Labex MS2T laboratory of excellence on technological SoS development.

SoS are complex systems that consist of multiple independent systems that work together to achieve a common goal. SoS Engineering is an approach that focuses on how to build and design reliable SoS that can adapt to the dynamic environment in which they operate. Given the importance of controlling constituent systems (CS) in order to achieve SoS objectives, the first part of this thesis involved a literature study about the subject of SoS control. Some control methods exist for large-scale systems and multi-agent systems, namely, hierarchical, distributed, and decentralized control might be useful and are used to control SoS. These methods are not suitable for controlling SoS in its whole, because of the independence of their CS; whereas, multi-views frameworks are more suitable for this objective. A general framework approach is proposed to model and manage the interactions between CS in a SoS.

The second part of our work consisted of contributing to Intelligent Transportation Systems. For this purpose, we have proposed the Cooperative Maneuvers Manager for Autonomous Vehicles (CMMAV), a framework that guides the development of cooperative applications in autonomous vehicles. To validate the CMMAV, we have developed the Cooperative Lateral Maneuvers Manager (CLMM), an application on the autonomous vehicles that enables equipped vehicles to exchange requests in order to cooperate during overtaking maneuvers on highways. It was validated by formal scenarios, computer simulations, and tested on the autonomous vehicles of the Equipex Robotex in Heudiasyc laboratory.

Keywords: Architecture Frameworks, Autonomous Vehicles, Intelligent Transportation Systems, System of Systems, System of Systems Engineering, System of Systems Control.

Résumé

La thèse porte sur le contrôle des systèmes de systèmes (SdS) et , sur la manière de construire des SdS adaptables et fiables. Ce travail fait partie du laboratoire d'excellence Labex MS2T sur le développement des SdS technologiques.
Les SdS sont des systèmes complexes constitués de plusieurs systèmes indépendants qui fonctionnent ensemble pour atteindre un objectif commun. L'ingénierie des SdS est une approche qui se concentre sur la manière de construire et de concevoir des SdS fiables capables de s'adapter à l'environnement dynamique dans lequel ils évoluent. Compte tenu de l'importance du contrôle des systèmes constituants (SC) pour atteindre les objectifs du SdS , la première partie de cette thèse a consisté en une étude bibliographique sur le sujet du contrôle des SdS. Certaines méthodes de contrôle existent pour les systèmes à grande échelle et les systèmes multi-agents , à savoir , le contrôle hiérarchique , distribué et décentralisé peuvent être utiles et sont utilisés pour contrôler les SdS. Ces méthodes ne conviennent pas pour contrôler un SdS dans sa globalité et son évolution , en raison de l'indépendance de leur SC ; alors que les "frameworks" multi-vues conviennent mieux à cet objectif. Une approche de "framework" générale est proposée pour modéliser et gérer les interactions entre les SC dans un SdS.

La deuxième partie de notre travail a consisté à contribuer aux systèmes de transport intelligent. À cette fin , nous avons proposé le gestionnaire de manœuvres coopératives pour les véhicules autonomes (CMMAV) , un "framework" qui guide le développement des applications coopératives dans les véhicules autonomes. Pour valider le CMMAV , nous avons développé le gestionnaire de manœuvres latérales coopératives (CLMM) , une application sur les véhicules autonomes qui permet d'échanger des demandes afin de coopérer lors de manœuvres de dépassement sur autoroute. Cette application a été validée par des scénarios formels , des simulations informatiques , et testée sur les véhicules autonomes du projet Robotex au laboratoire Heudiasyc.

Mots clés: Architecture Frameworks , Contrôle des Systèmes de Systèmes , Ingénierie des Systèmes de Systèmes , Systèmes de Transport Intelligent , Système de Systèmes , Véhicules Autonomes.

Acknowledgment

First and foremost I want to thank my advisors Dr. Reine TALJ and Prof. Ali CHARARA for their patience, trust, and their kindness, and also for their time and guidance throughout the thesis. Secondly, I want to thank the jury members: Dr. Dominique LUZEAUX and Prof. Eric BONJOUR for their detailed and insightful reports, Prof. Bernard DUBUISSON, Prof. Robert PLANA, & Prof. Isabelle FANTONI for their interesting questions and comments. I want also to thank Heudiasyc laboratory as well as Université de technologie de Compiègne for giving me the opportunity and the means to perform this thesis with success. Most notably, I want to thank all their staff for their support in the various aspect of my thesis. I would like to thank my family for all their love and encouragement. For my parents who raised me with love and supported me in all my pursuits. I could not wish for a better family. Also special thanks go to my friends, whom trusted me and helped me go through rough times. I am blessed for having such people in my life. Lastly, I thank GOD for everything.
Thank You.

List of Publications

International Conferences:

1 M. Assaad, R. Talj, A. Charara, “Cooperative Lateral Maneuvers Manager for Multi-Autonomous Vehicles”, in IEEE Systems, Man, And Cybernetics (SMC), Miyazaki, Japan, Oct. 2018.

2 M. Assaad, R. Talj, A. Charara, “Case study on systems of systems: Cooperative Maneuvers Manager for Autonomous Vehicles”, in 13th IEEE International Conference on System of Systems Engineering (SoSE), Paris, June 2018.

3 M. Assaad, R. Talj, A. Charara, “A view on Systems of Systems (SoS)”, 20th World Congress of the International Federation of Automatic Control (IFAC WC 2017) - special session, Toulouse, France, Jul. 2016.

Journal Papers:

4 M. Assaad, R. Talj, A. Charara, “An overview of technological systems of systems: general discussions and a case study on intelligent transport”, in IEEE Systems Engineering Journal, second revision.

Contents

List of Figures

List of Tables

Acronyms

ACRE	Approach to Context-based Requirements Engineering
AGN	Autonomous Ground Navigation
APTS	Advanced Public Transportation Systems
ARTS	Advanced Rural Transports Systems
ATIS	Advanced Travelers Information Systems
ATMS	Advanced Traffic Management Systems
ATS	Air Transportation System
AVCS	Advanced Vehicles Control Systems
CLMM	Cooperative Lateral Maneuvers Manager
CMMAV	Cooperative Maneuvers Manager for Autonomous Vehicles
CSP	Customer Security Program
CS	Constituent Systems
CVO	Commercial Vehicles Operation
DODAF	Department of Defense Architecture Framework
DoD	Department of Defense
ETSI	European Telecommunications Standards Institute
FAA	Federal Aviation Administration
GPS	Global Positioning System
GV	Ground Vehicles
ICC	Incentive Compatibility Constraint
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
INCOSE	International Council on Systems Engineering
ISO	International Organization for Standardization
ITS	Intelligent Transportation Systems
IT	Information Technology
LSS	Large-Scale Systems
MAS	Multi-Agent Systems
MBRE	Model-Based Requirements Engineering
MBSE	Model-based Systems Engineering
MODAF	Ministry of Defense Architecture Framework
MoE	Measures of Effectiveness
NAF	NATO Architecture Framework
NATCA	National Air Traffic Controllers Association
NATO	North Atlantic Treaty Organization
OMG	Object Management Group
PID	Proportional-Integral-Derivative
RFP	Request for Proposal
ROS	Robot Operating System

SE	Systems Engineering
SoI	System of Interest
SoSE	System of Systems Engineering
SoS	System of Systems
SV	Subject Vehicle
SWIFT	Society of Worldwide Interbank Financial Telecommunication
SysML	Systems Modeling Language
TOGAF	The Open Group Architecture Framework
TTC	Time-To-Collision
UAV	Unmanned Aerial Vehicles
UDP	User Datagram Protocol
UID	Unique Identifier
UML	Unified Modeling Language
V2V	Vehicle-to-Vehicle
V2X	Vehicle-to-X

Chapter 1

Introduction

Humans build systems to solve a problem or fulfill a need. Sometimes these problems or needs are not complicated in nature, or their solution is rather simple, so simple systems may do the job. But sometimes, they are complicated, so we need complex systems to get us through. The nature of a system, however, does not depend only on the problem that we want to solve, or the need we want to fulfill, but it also depends on our requirements from that system, and the environment of that system. Interacting with the environment does not only affect the system itself, but also may affect other systems in that environment as well. In an environment that contains large numbers of systems affecting and interacting with each other, a global system emerges that displays behaviors that could not be traced back to a single or set of parts (e.g. a global transportation system). For a long time, such systems were either unknown to exist, or when such system is identified, it is studied as a specific case. Recently, these systems are getting famous, and they are knows as Systems of Systems (SoS). SoS are dynamic systems in which parts are by themselves independent systems, and work together either because they share the same objectives or because working together gives them benefits they could not achieve alone.

This thesis is part of the Labex MS2T[1] "Control of Technological Systems-of-Systems", a multi-disciplinary scientific program that targets a scope of applications that may be very large. This program focuses on 4 main axes of research (Fig. 1.1): interaction and cooperation between systems, uncertainty management, optimized design of technological SoSs, and dynamics of SoS: emergence and agility. The main focus of this thesis is the subject treated by the first axis: "interaction and cooperation between systems". Most notably, in the context of SoS control, we explore the approaches used for control in SoS, and similar systems such as large-scale systems (LSS) and multi-agent systems (MAS), in order to answer to the following question: how does SoS authority control the constituent systems (CS) in order to achieve the global goals of the SoS?

The first part of the thesis consisted of a detailed study of SoS in the literature. Due to the diversity of domains of applications of SoS, there exist large number of definitions that define SoS in the context of a specific domain of applications, whereas other generic definitions that ignore the domain of application and focus on SoS as systems tend to give characteristics that describe SoS as well as its CS. This litera-

[1]www.labexms2t.fr

[2]www.labexms2t.fr/presentation/overview.html

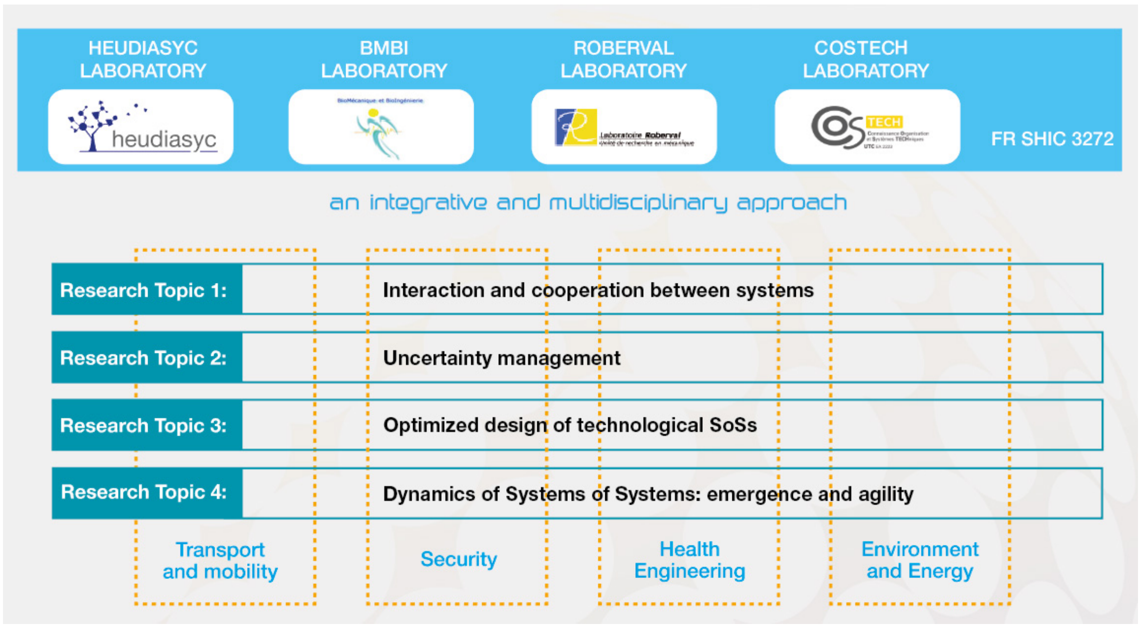


Figure 1.1: Labex MS2T Project Scheme: 4 Axes of research, with 4 laboratories involved [2].

ture study consisted also of studying the different types of SoS, which are regrouped based upon several attributes such as the management type, or the role of CS in the SoS. The most pertinent types are the directed, acknowledged, collaborative, and virtual SoS, where SoS are grouped based upon the management style adopted within. The dynamic and evolutionary nature of SoS led to the emergence of a new engineering discipline that builds upon traditions systems engineering, in order to address the unique challenges that face SoS, namely, SoS Engineering. There is an identified research gap that exists in SoS engineering in subjects such as SoS integration, emergent behaviors in SoS, and SoS control.
The first contribution of this thesis is in the subject of SoS control, and most notably, the control methods that are used to control similar systems such as LSS and MAS. When systems become large and complex, the conventional centralized control methods become insufficient, and non-centralized methods such as hierarchical and distributed control are used to overcome the shortcoming of centralized control. There are cases where these methods are used in the context of controlling SoS. However, the study of these methods showed that they are not suitable to be adapted in SoS because of their dynamic nature, and the independence of their CS. To capture the complexity of SoS, and the numerous interrelationships that exist inside SoS, and to put rules that guide CS so that their individual behavior results in the achievement of SoS objectives, multi-views framework are a suitable tool that is used to build SoS. Multi-view frameworks give us the ability to represent a complex SoS, and assign rules and guides for CS. They also support high-level analysis and validation which are important decision-making aiding tools during the design and development of SoS.
After the literature study, and to apply what we have learned, our second contribution, in the context of Intelligent Transportation Systems, is a framework that guides the development of cooperative applications in autonomous vehicles (such as cooperative overtaking, shared parking, etc.). The Cooperative Maneuvers Man-

ager for Autonomous Vehicles (CMMAV) uses SoS principles and is based on the COMPASS's SoS-ACRE framework (Holt, S. Perry, et al. 2015), but uses slightly different viewpoints and methodology.
The first application that uses the CMMAV is the Cooperative Lateral Maneuvers Manager (CLMM), which uses the recommendations of the CMMAV in order to allow autonomous vehicles to overtake on highways in a cooperative manner. Formal scenarios, computer simulations, and experimentation on the autonomous vehicles of the Heudiasyc laboratory, within the "the equipment of excellence" Equipex Robotex[3] project, were used to validate the CLMM, and the results showed that the CLMM respects the requirements of its CS and the different entities that are interested in it, and that the behavior of the SoS also matches the predefined desired behavior.

1.1 Thesis Organization

This thesis is organized as follows: Chapter 2 contains the state of the art of SoS in the literature, existing definitions of SoS, the different taxonomies used to describe different types of SoS, and some of the challenges that face SoS development and engineering. The topic of SoS engineering opens the door to questions about SoS control, and how SoS authority must handle its CS in order to achieve the desired operation within the SoS. This topic is the subject of Chapter 3, where we look at control paradigms and methods used in complex systems similar to SoS, and discuss how we can apply such methods (if they are applicable) to SoS control. Chapter 4 tackles the subject of modeling in SoS, most precisely, architecture frameworks that are used to model and analyze SoS during design and operation phases. Inspired by this approach, and by the ACRE-SoS framework, Chapter 5 describes our approach to cooperative autonomous driving: the CMMAV. This Chapter describes the different views contained in the CMMAV, as well as the use-case based methodology adopted in its development. Chapter 6 introduces the first application that was built using the CMMAV framework that caters for the cooperative overtaking maneuvers. The Cooperative Lateral Maneuvers Manager (CLMM) is the application developed for this purpose, and is the subject of this Chapter. Finally, to end it all, Chapter 7 concludes this work, and gives some future perspectives.

[3]httt://equipex-robotex.fr/

Chapter 2

Systems of Systems

2.1 Introduction

Humans build systems to solve a problem or fulfill a need. When we need drinking water, we build systems to extract water from mountains or distill seawater, and when we want to enjoy the beauty of the ocean, we build yachts and boats. Sometimes these problems or needs are not so complicated in nature or their solution is rather simple, so simple systems may do the job, for example a mercury thermometer to measure the temperature of a room, but sometimes, they are complicated, so we need complex systems to get us through, like for example an airport with all its buildings and staff to manage different air trips. The nature of a system however does not depend only on the problem that we want to solve, or the need we want to fulfill, as we can measure the temperature of a room with a simple mercury thermometer, and we can do it with a very advanced temperature sensor that is much more advanced and complex. What changes here is our requirements of the system. In the first case, we wanted only to measure the temperature of the room, while in the second case, we wanted a very precise measurement that could not be achieved by the simple measurement tool. Furthermore, another factor that affects the nature of a system is its environment, or the physical place in which a system operates. Since systems interact with the world, the environment in which they operate might constraint the system: a robot that is designed to move on a closed rail with no obstacles is different than a robot that is designed to move in a factory, where it should perceive its environment to avoid collision with objects, for example. Interacting with the environment does not only affect the system itself, but also may affect other systems as well. A vehicle moving on a road might affect its neighbor vehicles, when for instance it changes its speed, this behavior may lead to other vehicles changing their speeds or their lanes. This interaction lead to an implicit relation between different systems that emerged because they share the same environment. In an environment that contains large numbers of systems affecting and interacting with each other, a global system emerges that shows behaviors that could not be traced back to a single or set of parts (e.g. a global transportation system). For a long time, such systems were either unknown to exist, or when such system is identified, it is studied as a specific case, but recently, these systems are getting famous, and they are known as System of Systems (SoS). SoS are dynamic systems in which parts are by themselves independent systems, and work together either because they share the same objectives or because working together gives

them benefits they could not achieve alone. Studying SoS is very important because it gives us insights about how to build them, how to improve them, and thus use the advantages they provide.
The objectives of this chapter are the following:

1. Introduce SoS and their different types;
2. Provide examples that are helpful to fully understand SoS;
3. Introduce the set of challenges facing SoS;

2.2 Systems

The word "system" has multiple meanings depending on the context. The most pertinent definition for us here is "a regularly interacting or interdependent group of items forming a unified whole" [1]. This definition contains two important statements:

- Regularly interacting or interdependent group of items;
- Forming a unified whole.

The first statement tells us that there is a group of items that are working together with a constant or definite pattern, they might be interdependent (they are dependent upon one another), or just interacting (they influence one another). The second statement tells us that this group of items is not regarded as simply a group of items, but rather as a unified whole, which act together to achieve this group's or the whole's goals, even if this means some items might sacrifice for it. More precisely, in systems engineering (SE) field, the definition of "system" is more adapted to the cases treated by SE. Let us take a look on some of these definitions:

- A "system" is the combination of elements that function together to produce the capability required to meet a need. The elements include all hardware, software, equipment, facilities, personnel, processes, and procedures needed for this purpose (NASA 2016).
- Combination of interacting elements organized to achieve one or more stated purposes [2].
- Homogeneous entity that exhibits predefined behavior in the real world and is composed of heterogeneous parts that do not individually exhibit that behavior and an integrated configuration of components and/or subsystems (INCOSE 2015).

These definitions are more specific in defining a system from SE perspective, by providing boundaries to the more general definitions. An airplane with its different parts alongside the crew, achieving the purpose of transporting passengers from point A to point B is a system, and a water boiler system boiling water is a system as well. From the previous definitions we can extract two important points: A system always has an objective (or set of objectives), and its behavior to achieve its objectives is predefined.

[1] www.merriam-webster.com/dictionary/system

[2] ISO/IEC/IEEE 15288:2015 Systems and software engineering — System life cycle processes, 4.1.46.

System Objectives: Every system is built to solve a problem, or to achieve an objective: a car is built to transport its passengers from a source to a destination, and a weather satellite is built to make specific measurements of the atmosphere. The objectives of these systems are defined by their owners or operators, and all the parts must work together to achieve them. It should be noted that system objectives and functionalities are not same, as will be discussed later on.

System Behavior: To achieve its objectives, a system must be able to perform a series of tasks or processes, and it must perform them in a predefined way. This is important for many reasons: if the behavior is predefined, any problem is easier to be identified and solved, also a predefined behavior is easier to test prior to the system deployment to avoid any operational surprises. For example, to heat water to precisely $85.5°C$ using a water heater, if its behavior is not predefined, it might heat water sometimes to $70°C$, and sometimes to $110°C$. The behavior of a system is defined by its operators or owners.

Systems are not limited to machines though, societies are systems in which people work together to achieve the common goal of well being. A company with its employees, management, and assets is a system in which all parts work together to maintain and grow it is a system as well. In biology, every living being is a system that is composed of many organs and cells that work to preserve that being. Systems exist everywhere, and in every domain of science. Going back to human built systems, which are the focus of this thesis, an airplane and a water boiler are both systems based upon the previous definitions, they both have objectives, and their behavior is predefined. However, there is an important difference that exists between those two systems, namely their complexity. An airplane is a complex system, whereas a boiler is a simple system, but what is a complexity? And what is a complex system? The answers to these questions are the topic of the next section.

2.2.1 Objective vs. Purpose

In this thesis we treat the subject of SoS, wherein the subject of systems management is pertinent as we will see later on. For this reason, we are going to use the terms "objective" and "purpose" multiple times referring to the system objectives and purposes. These two terms differ slightly in meaning, that is why we are going now to highlight the difference between them.

2.2.1.1 Purpose

Purpose describes the reason why something exists. The purpose of a mobile phone is to give its owner the ability to communicate with other people. A system's purpose is the reason why this system was created, the overall vision or mission for this system. It is abstract and intangible.

2.2.1.2 Objective

The objective is more specific than purpose. It is a concrete action that will lead to achievement for purposes. A mobile phone has multiple objectives that lead to achieve its purpose, such as enabling internet connectivity, and making vocal calls, for example. While a system's purpose represents the why, its objective represents the how. The objective is specific, and tangible, and can be measured and validated.

2.2.2 Systems Functionalities

A functionality is an action performed by a system. System functionalities are any action that the system is capable of doing, whether it be gathering some type of information, or moving an object. Some functionalities in a system are mandatory to have, which represent the minimum set of actions a system must do in order to achieve its objectives, and consequently, fulfill its purpose. However, there are some functionalities that are added to systems to improve their operation. For example, a fingerprint scanner on a desktop computer can store passwords. This saves five minutes every time the user forgets a password and has to look it up or reset it, it also makes the computer more secure. A desktop computer can function very well even without having the functionality to store passwords, but having this functionality is beneficial since it improves the user experience.

> Functionalities are all the actions a system can do, and objectives are the actions that a system must do.

2.3 Complex Systems

Before discussing complex systems, let us discuss complexity itself. Complexity is a very complex term to define, since it is very context dependent. The origin of complexity comes from the Latin word *complexus*, and it translates literally to “woven together” or “entwined”. Complexity in systems is used to describe the state of a system that has many parts, which have intricate, or nonlinear relationships. To understand this, let us look at a simple system, in which the parts have linear relationships. Let us reconsider the boiler: it has a cold water feed, a tank that contains the water, a coil that transforms electricity into heat, and gauges the adjust the flow of water. The relationship between the let us say the coil and water and the effects they have on each other are known, the coil heats the water. If the water is not boiling as expected, we can trace that to the coil, if the coil is not damaged, then there is a problem in the electrical circuit, and so on. In such systems, namely simple systems, the cause and effect relations between parts are known and relatively simple, which is not the case in complex systems. In complex systems, a modern passenger cars for instance, which consists of large number of components working together to enable the car to go from a source to a destination. Different sensors acquire different inputs like speed and acceleration, actual weight and other measurements which are used to control the behavior of the car. If we consider the lateral acceleration measurement, it affects speed control, slip control, and steering control, which are affected themselves by a number of variables. Finding the cause of an issue in a car is not trivial, even though some causes are more likely the reason behind a specific issue (for example eroded fuel filter is likely the reason behind increased fuel consumption), often when an issue appears in a car, it needs to be checked up to identify the exact reason. In summary, complex systems are systems that consist of a large number of components which have nonlinear relationships. Large number of components and nonlinear relationships are not enough to identify a complex system alone, however, when they exist in a system, this system shows interesting behavior called “emergent behavior”, that might be a good indicator of the complexity of the system (Boccara 2010).

2.3.1 Emergent Behavior

Emergence occurs when a group of parts, namely the whole shows properties (or behaviors) that could not be explained beforehand with the simple knowledge of the properties (or behaviors) of the parts. Emergence exists in all disciplines that deal with systems: biology, cosmology, engineering, etc. Emergence occurs in systems that are composed two or more parts, that lead, when they interact, to one or more behaviors on the whole's level that does not reside in any individual part. From biology, birds are an example of a system that has an emergent behavior, namely flying: none of the parts or organs of a bird can fly by itself. However the whole, or the bird flies. We might try to trace the flying behavior to the wings, however, if take two wings and flapped them, it is very unlikely that we could fly. The flying property, or the emergent behavior in a bird is the result of all the parts working together. Continuing with our previous examples, the car and boiler: if we are to trace the "moving upwards" behavior of a passenger car to its parts, the closest behavior that exists to moving is in its wheels, however, alone, the wheels move downwards on an uphill road. On the other hand, all properties and behaviors that exist in the boiler could be traced to its parts: boiling is a property of water when heated, and heating is a property of the coil when an electrical current passes through. Emergence is one of the most important properties in complex systems, and often systems are built for their emergent behaviors: vehicles to move stuff, airplanes to fly, etc. These types of behaviors are desired known behaviors, we know that when we build a car it will move, and we build it to do so, however, emergent behaviors might sometimes be undesirable: a car slipping on a wet road, for example. It is known that a car might slip on a road if certain conditions are met, and so we equip cars with systems that have the objectives of making sure that the car does not slip. Emergent behaviors could be classified into 4 categories based on these two criteria (Fig. 2.1): desirability and predictability. Usually system

Emergence	Desirable/positive	**Undesirable/negative**
Expected emergent behavior	Reason for building the SoS (**SoS objective**)	**Mitigate** by appropriate design measures, such as threat/risk analysis and countermeasures
Unexpected emergent behavior	Sometimes (however, quite rarely) an SoS shows unexpected, **beneficial behaviour** 🤔	Unexpected & undesirable negative emergent behavior is one of the **critical risks** of most SoS

Figure 2.1: Emergent behavior classification (Furrer 2017).

builders focus on improving known desirable emergent behaviors, and limiting known undesired emergent behaviors from appearing in the system. Known behaviors are identified in systems either by experience (a certain behavior appeared before in a similar system), or by studying the system before its deployment, however, the problem is with unknown emergent behaviors. The problem with unknown behaviors is that they appear when a complex system is operating, if we are lucky, these behaviors might be desirable, like flying airplanes, or undesirable but not critical to the operation of the system (such as temporal overheating of a component in the system), but if unfortunately we are unlucky, such behaviors might lead to the destruction of the system (e.g. the destruction of a power plant).

2.4 System of Systems

So far we have discussed systems as separated entities that are built to solve a problem or to achieve an objective, and we showed that a system might be simple or complex, based on its parts and their inter-relationships, and their emergent behaviors. However, in reality, systems are rarely separated when they operate, and they often have several types of relations with the world through their environments. These relations with the world might take two forms: explicit relations and implicit relations.

Explicit Relations: This form of relations is known and accounted for before deploying the system, for example a car running on fuel will interact with gas stations in order to get the resources it needs to operate, or a power plant that uses fuel to produce energy which takes fuel as a resource and produces energy as a product.

Implicit Relations: This form of relations is either unknown or unaccounted for before deploying the system, for example a car slowing down will force the car behind to slow down as well to avoid collision.

Regardless of the environment in which a system operates in, explicit relations exist in all systems because all systems require some type of resources to operate, and/or produce a product. On the other hand, implicit relations are often observed in systems that share their environment with other unrelated systems. Implicit relations are not limited to forced relations, and they extend as well to conscious decisions made during the operation of the system.
To demonstrate the previous types of relations, consider the following scenario: a company LC_A wants to transport goods overseas at the time t_A, pays a transportation company TC to move its goods. It happens that around the same time, another company LC_B also needs to transport goods to the same destination as LC_A but at time t_B, and pays TC to transport them. Assume that the time difference between both orders is $\delta T = t_A - t_B = 1day$. To reduce the number of trips, TC contacts both companies in order to reach a possible agreement on a single date for the transportation, and informs them that they will pay less if they ship their goods at the same time. Obviously LC_A, LC_B, and TC are three systems that have each separate objectives. The relations that exist between LC_A and LC_B, on one hand, and TC, on the other hand, are explicit relations since both LC_A and LC_B know that they will be using another company to move their products when building the system, and in consequence achieve their objectives. However, the relation between LC_A and LC_B is implicit, since it was not thought beforehand, but rather it is the result of the 3 systems sharing the same operation environment, and it affects them since either LC_A, or LC_B or both might change their transportation date to benefit from

TC's offer. The 3 systems now share a single objective, which is transporting the goods, and according to the definition of systems and complex systems mentioned in Section 2.2 and in Section 2.3, during the period of shared operation, they could be seen as a single system. This system, which consists of multiple systems that operate together to achieve the now shared objective, is what we call a System of Systems (SoS).

2.4.1 SoS Historical Background

SoS is a concept that describes a specific type of systems that form from two or more systems, that are related to explicit or implicit relations, or both. If we trace SoS in the literature, it was first mentioned in the year 1956, when (Boulding 1956) used SoS to describe the objective of a general system's theory, that seeks to create a spectrum of theories that use the similarities between disciplines in system theory. This spectrum, which they called "System of Systems", perform the function of a "gestalt" in theoretical construction. A "gestalt" is a German word for form or shape[3], and is used to describe a whole which is greater than the sum of its parts. In contrast with emergence and complex systems, this definition describes a complex system more than a SoS as we will see later.

Later on, SoS was used by (Ackoff 1971), (Jacob 1974), and (Jackson and Keys 1984), to in the domains of organization's management, biology, and operational research respectively. Ackoff used the term SoS in an attempt to organize different types of systems into a system that highlight their differences and similarities. Jacob described SoS as "every object that biology studies" (Jacob 1974), and Jackson & Keys used "SoS Methodologies" that use the relations between problem-solving methodologies and systems-based problem-solving methodologies to solve operational research problems.

In the 4 previous papers, the term SoS was used to describe an object, an idea, or a set of methodologies, rather than describing what is a SoS. We should note, however, that since the first mention of SoS, it was always related to emergence, which is an integral part of SoS. The first attempt at describing SoS as a concept was by (Eisner, Marciniak, and McMillan 1991) where they described SoS as a group of multiple independent systems acquired independently. These systems, when operating together, form a multifunctional solution for a global "coherent mission". They further note that the optimization of individual systems in a SoS does not guarantee the optimization of the SoS. This first definition of SoS introduced the notions of independent systems, and the global mission. Another view on SoS comes from (Shenhar 1994), where they considered SoS as a "large collection or a network of systems functioning together to realize a common task". While the last 2 definitions describe a whole that is formed from multiple systems, the way these systems are put together to form the whole, or the SoS, differs between the two definitions: Eisner et al. talk about SoS as if they are built by an entity to solve a problem (systems are acquired), while Shehnar considers only the result.

Adaptation, or evolutionary development was introduced by (Holland 1995) in 1995, where they view SoS as constantly adaptive systems. This adaptation is the result of different local rules that reside in the parts, and their capacity of auto-organization. In the same year (Owens 1996) used this same concept to introduce SoS in the military domain.

[3]https://en. wikipedia.org /wiki/Gestalt

In 1996, SoS and military domain witnessed the second important association when (Manthorpe 1996) introduced the concepts of command, control, computation, communication, and information (C4I), and intelligence, surveillance, and reconnaissance (ISR) to SoS in military applications. The same year witnessed what would be considered later the most accepted definition of SoS in (Maier 1996). Unlike previous attempts at defining SoS using a textual description, Maier described SoS using a set of 5 characteristics that when respected in a system, this system could be considered a SoS. Moreover, the parts that form the SoS are now described as constituent systems (CS) rather than parts or subsystems. These characteristics are

- operational independence of CS,
- managerial independence of CS,
- geographical distribution of CS,
- emergent behavior in the SoS,
- evolutionary development of the SoS.

The detailed description of each characteristic will be given in the next section. The importance of this definition is that it did not impose any limits on the way SoS forms, and it removed a lot of ambiguities that existed in the previous definitions. For example, using Shenhar's definition, a power plant is considered as a SoS, as well as a network of sensors deployed in a large field is also a SoS, but if we apply Maier's characteristics, both the power plant and the network of sensors are not SoS (see Section 2.4.3). This is because Shenhar's definition does not impose any specific conditions on the CS, whereas Maier's definition does.

In 1997 (Kotov 1997) defined SoS in the context of information-intensive systems as distributed, large-scale systems, in which components are complex systems by themselves, where the main emphasize is on communication between CS. Kotov did not only define SoS, by he also proposed the first approach to model and analyze SoS that considers CS as nodes in a hierarchical structure using a C++ based library called CSL, and applied this tool to a case study of a global transportation company. The emphasize on communication in SoS is further supported by (Maier 1998), where he considered that communication and CS interfaces are the most important aspect in SoS.

The use of SoS in military domain was considered again in the year 2000 by (Pei 2000) where he introduced integration systems in SoS. Integration systems are systems that are added to SoS for the purpose of coordinating between different CS in military SoS. The years that follows witnessed a huge increase in the use of SoS approach in several fields of applications, and on different levels as well. From generic architecture frameworks that are intended to serve as blue prints for SoS design and analysis [(Partners 2005), (Ballagny, Hameurlain, and Barbier 2009), (Officer 2010)], to SoS development processes and models [(Shams et al. 2008), (Kewley Jr and Tolk 2009), (Acheson, C. Dagli, and Kilicay-Ergin 2013)]. SoS specific applications in domains such as defense applications [(D. L. Farroha and B. S. Farroha 2011), (Sanduka and Obermaisser 2014)], transportation [(DeLaurentis 2005), (Farcas et al. 2010)], and cyber security (D. L. Farroha and B. S. Farroha 2011) to cite a few. In 2005, the first international conference dedicated to SoSE was launched:

IEEE Systems of Systems Engineering Conference[4], which is a yearly conference that has the objectives of gathering the SoS community to improve the state of SoS development. Several European studies addressed SoS as well: T-Area-SoS[5] which is a collaboration between organizations from the United States and Europe for the purpose of creating a SoS roadmap for SoS development, Road2SoS[6] which also focuses on creating a roadmap for SoS development in the energy, manufacturing, crisis management, and traffic control sectors, and Compass[7] which is a European study that focuses on model-based approaches applied to SoS development.
With all the interest surrounding SoS, we wonder what is a SoS? And why are they so important to study? The next section will answer these questions.

2.4.2 SoS Definition

In the literature, we observe two different types of definitions for SoS. The first type is domain-based, this means that definitions of this type try to define the use of SoS in a specific domain:

Definition 2.4.1. *Enterprise systems of systems engineering is focused on coupling traditional systems engineering activities with enterprise activities of strategic planning and investment analysis. (Carlock and Fenton 2001)*

Definition 2.4.2. *SoS integration is a method to pursue development, integration, interoperability, and optimization of systems to enhance performance in future battlefield scenarios. (Pei 2000)*

Definition 2.4.3. *SoS is every object that biology studies. (Jacob 1974)*

The second type of definitions are those that try to define SoS regardless of the domain. They try to give indications that can be used to identify a system as a SoS:

Definition 2.4.4. *A set of several independently acquired systems, each under a nominal systems engineering process; these systems are interdependent and form in their combined operation a multi-functional solution to an overall coherent mission. (Eisner, Marciniak, and McMillan 1991)*

Definition 2.4.5. *A large widespread collection or network of systems functioning together to achieve a common purpose. (Shenhar 1994)*

Definition 2.4.6. *SoS are large-scale integrated systems which are heterogeneous and independently operable on their own, but are networked together for a common goal. (Mo Jamshidi 2008)*

Definition 2.4.7. *SoS brings together a set of systems for a task that none of the systems can accomplish on its own. Each constituent system keeps its own management, goals, and resources while coordinating within the SoS and adapting to meet SoS goals.* [8]

[4]http://sosengineering.org
[5]www.tareasos.eu
[6]http://www.road2sos-project.eu/
[7]http://www.compass-research.eu/
[8]ISO/IEC/IEEE 15288:2015(E), Annex G.

2.4.3 SoS Adopted Definition

For the first part of this thesis, where we discuss SoS, control and modeling, we are interested in them as systems in general, and we do not have any preference to any application domain. Thus, the second group of definition which defines SoS regardless of its domain is more pertinent for us. The different definitions contained in this group share different characteristics that describe either the SoS as a system or its CS: definition 2.4.4 talks about independent CS, and that they work for an overall coherent mission, a point shared by all other definitions. Definition 2.4.5 mentions the widespread of CS, while definition 2.4.6 focuses on their heterogeneity and independence. Using these similarities in definitions, and studying existing SoS, Maier (Maier 1998) provides a definition based on characteristics that should exist in a system, in order to be classified as a SoS:

Definition 2.4.8. *SoS is an assemblage of components which individually may be regarded as systems, and which possesses two additional properties: operational and managerial independence.*

Three other characteristics are often used alongside the previous 2 when describing SoS (Maier 1996): emergent behavior, geographic distribution, and evolutionary development. In fact these 3 are the natural consequences of independent systems working together as we will show later on. But before that, let us define operationally independent, and managerially independent systems.

2.4.4 Independent Systems

The notion of independence in SoS is important and often used to describe CS that constitutes the SoS. In (DiMario, J. T. Boardman, and B. J. Sauser 2009) authors use autonomy as a characteristic to describe CS in SoS as a substitute for independence. Independence can be used to describe multiple aspects of a system: a system might be resource independent, human independent, and so on. Autonomy, in (DiMario, J. T. Boardman, and B. J. Sauser 2009), is described as “the ability to exhibit autonomic properties” such as self-configuring, self-optimizing, self-healing, and self-protecting. All of the previous properties exist in any operationally and managerially independent system, and therefore we will discuss only these 2 characteristics in this section.

2.4.4.1 Operational Independence

A system that has its operational independence is a system that can operate by itself, without the need for other systems. For example, consider a drone controlled by a human operator: while the drone itself is a system, it is not an operational independent system because to operate it needs the human operator. However, if we consider the whole system, drone + operator, this system might be operationally independent.

> An operationally independent system is a system that is capable of achieving its objectives by itself.

In the previous example, if the drone’s objective is to go from point A to point B, both the human operator and the drone must operate together to achieve this objective, since the drone is not capable of navigating itself. On the other hand, an autonomous vehicle is an operationally independent system because once an objective is assigned, it is capable of achieving it.

2.4.4.2 Managerial Independence

Systems management is an umbrella term that includes monitoring, design, resource allocation, etc. of a particular system. Managerial independence of a system means that all decisions related to any aspect of a system within a defined scope are made within the system. The management of a system might change during its life-cycle: during design cycle, the management is responsible for design decisions such as system architecture, technology used and its intended behavior. During the operation cycle, the management tasks consist of resource management, setting the system's objectives, choosing the right operational mode, etc. Reconsider the drone from the previous section, if it has the capability to set objectives without the operator's interference, then we could consider it managerially independent, whereas if the operator is the one that sets its objectives, then the drone by itself is not managerially independent.

2.4.5 Other Characteristics

Autonomy, belonging, diversity, connectivity, and emergence are used in the literature as defining characteristics of SoS [(J. Boardman and B. Sauser 2006), (John Boardman and Brian Sauser 2008)]. Autonomy is the ability of a CS to make independent choices, while belonging means that CS choose to belong to the SoS, and that they could leave it whenever they desire, and diversity represent the heterogeneity of CS in SoS. These 3 characteristics could be traced directly to the managerial independence of the CS, because managerial independence means the ability to make independent choices, one of which is the belonging to the SoS. Having multiple managerially independent systems working together implies the diversity in a SoS, because each system might have different technologies, or different objectives. As for connectivity, when multiple independent systems work together, they must communicate in order to cooperate, this means that connectivity is implicitly included in the previous definition. Finally, emergence as we have seen in the Section 2.3.1, is the natural result of independent complex systems interacting. That is why, in what follows, to justify the classification of a system as a SoS we will only prove the operational and managerial independence of its CS. This does not mean, however, that the other characteristics could be neglected, since they are key points to be thought of when designing a SoS: what are the emergent behaviors that might raise? How to ensure the connectivity of heterogeneous CS? etc.

2.4.6 Cyber-Physical Systems and SoS

Cyber-physical systems (CPS), are systems which "consist of a computing device interacting with the physical world in a feedback loop" (Alur 2015). This means that a CPS is any system in which a software controls a real world physical entity. In this modern era, almost all systems are considered as CPS: power plant, electric vehicles, smart lights, etc. When a system of interest consists of multiple CPS, it is referred to as cyber-physical SoS (CPSoS) (Elshenawy, Abdulhai, and El-Darieby 2018). It is important to keep this special category in mind when working with SoS, because even if the SoS of interest is not CPSoS, it would most likely have multiple CPS as CS. Recognizing this beforehand, we could anticipate a lot of challenges that are inherent in CPS, such as system security, and we could take advantage of such systems.

2.4.7 Discussion

SoS were, are, and will always exist, because they are the natural consequence of independent systems interacting because they share the same environment and/or objectives. This led as we have seen to the numerous and diverse definitions that try to define SoS. Understanding SoS is important because it enables us to achieve 2 objectives: Learning how to build a robust and reliable SoS, and improving existing SoS. One challenge that we face in this regard is the fact that operational and managerial independence are qualitative criteria, which means that there are no values to measure the degree of independence a system has in contrast to some quantitative criterion. However, let us put independence on a scale ranging from completely independent to completely dependent, and assuming that complete independence means that all decisions are made within the system, and complete dependence means that no decisions are made within the system, and they all come from outside the system. To make things clearer, let us build an operational and managerial independence scales and place some systems on them. In Fig. 2.2 we show 3 system examples placed on the managerial independence scale: a power plant, drone, and human-driven vehicle without the driver. A human-driven vehicle without its driver might be considered a managerially dependent system because all decisions regarding objectives, operations and so on are made by the driver, who is an outsider to the system. In contrast, in a power plant where all decisions are made by its management, which resides inside the system, we could say that this is a managerially independent system. In between the two edges of the scale, systems such as the drone that receives its objectives let us say from someone outside, but also has the ability to take decisions on resource management for example is a typical example of a system that is neither managerially dependent nor independent, but rather in between. In reality, before joining the SoS, CS are operationally and managerially

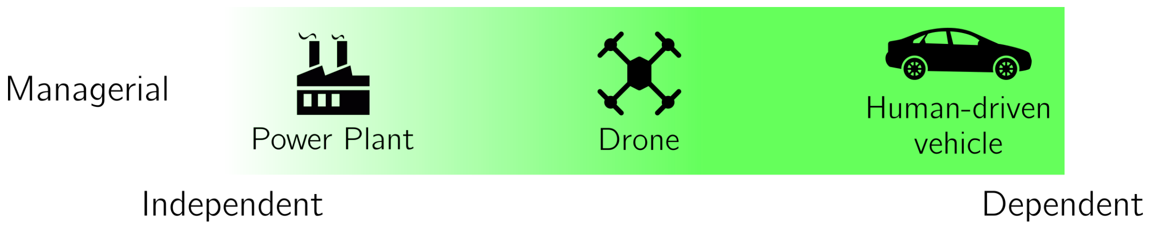


Figure 2.2: Managerial Independence: From completely independent to dependent systems.

independent: all decisions are made within the system, and it can operate without the need of any other external system. However after joining the SoS, CS usually compromise some of their independence in order to benefit from their belonging to the SoS.

To summarize, Fig. 2.3 represents the boundaries of what we consider an independent system. An independent system is a system that consists of its operational part, i.e. the part that does what the system is built to do, and a managerial part, which is responsible for deciding what the system has to do. Since independent systems operate in an environment shared between different independent systems, there are several constraints that apply to the managerial level such as regulations, or the requirements of other stakeholders, while the operational part uses resources to achieve its functionalities, while at the same time accounting for the external factors that might affect it.

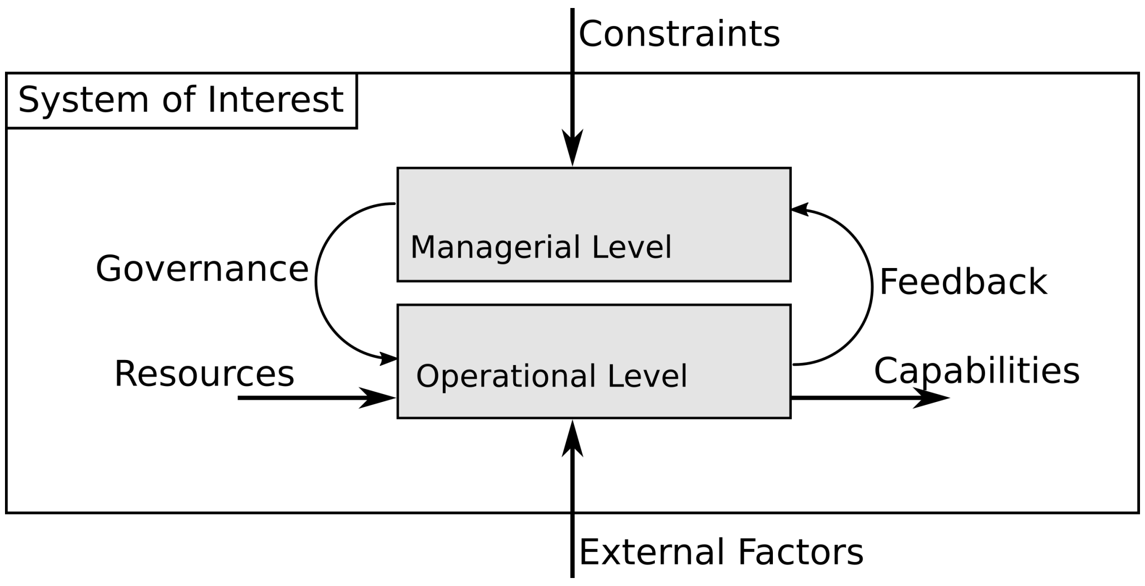


Figure 2.3: An independent system consists of its operational part and its managerial part together in the system.

2.5 SoS Taxonomy

Regardless of the their diversity either in their domain of applications, or the type and nature of the systems involved (CS), SoS face globally the same challenges, such as the possibility of undesired behaviors due to emergence, their fragility towards the surrounding environment, the dynamic nature of SoS due to CS joining/leaving the SoS virtually whenever they desire, etc. However, if we look at SoS management on a global level, i.e. who sets its objectives, how it is governed, who is responsible for managing resources, etc. we could find a clear distinction to be made between SoS. This distinction in the way SoS is managed led to multiple taxonomies used to classify different SoS: directed, acknowledged, collaborative, and virtual [(Maier 1998), (Dahmann and K. J. Baldwin 2008)]. Later on, three attributes were identified by (Collins, Doskey, and Moreland 2016) to further explain the differences between these 4 types. So let us first define each type, and then introduce the 3 attributes that we could use to better classify SoS. Finally, we will introduce some of the newer proposed taxonomies that use different attributes than SoS management to classify them.

2.5.1 SoS Management taxonomies

When comparing the management style between different SoS, authors came up with 4 different taxonomies in order to group different SoS with similar management style. Directed, acknowledged, collaborative, and virtual are the 4 groups that are used to distinguish between "single complex systems"-like SoS, namely directed SoS, and "chaotic complex systems"-like SoS. At first, 3 out of the 4 types were proposed by (Maier 1998):

2.5.1.1 Directed SoS

It is SoS which is built for specific predefined purposes, and is centrally managed throughout its operation to fulfill these purposes. The normal operation mode for CS in such SoS is subordinated to the SoS's central management, however, CS retain the ability to operate independently within the SoS. Organizations such as research laboratories, and communities such as AirBnb and Waze[9] are examples of directed SoS because the management is what defines all the rules that must be respected by their CS.

2.5.1.2 Collaborative SoS

It is SoS which consists of a group of CS that work together voluntarily, mainly because they share the same objectives, or because they share the same purposes. In this type of SoS, the rules that guide the evolution of the SoS, as well as the overall objectives are defined on the CS level collectively. An example of a collaborative SoS is the internet, where there are several independent organizations (internet suppliers, technology developers, etc.) that work together to make sure that the internet is operating and serving its users.

2.5.1.3 Virtual SoS

Emerge when different CS are related via implicit relations (Section 2.4) due to them sharing the same operation environment. These relations may be known or unknown, however, they lead to the emergence of SoS. The objectives of such SoS are relatively unknown, as well as the knowledge of all the CS that are part of this SoS. An example of a virtual SoS would be the global stock market, where the majority of factors that affects it are unknown. This leads to some unpredictability in the market, and sometimes to undesired behaviors such as market recessions in late 2000s and early 2010s.
Later on, the United States Department of Defense (U.S. DoD), a major contributor to the field of SoS Engineering, proposed the acknowledged taxonomy, in which SoS shares attributes from both directed and collaborative SoS. Acknowledged SoS were found to be more suitable to be applied in the DoD's applications.

2.5.1.4 Acknowledged SoS

These systems have a central management and resources for the SoS, but CS retain their full independence, and they are sharing the responsibility of the changes in the SoS alongside the SoS management. These SoS, however, has recognized objectives and management similar to directed SoS. Very large organizations such as governments and multinational companies are acknowledged systems, since the overall management is responsible only for defining the SoS objectives, and CS are responsible for their own management and rules.

2.5.1.5 Attributes to Distinguish Between Different Management Taxonomies

The 4 taxonomies used in the management taxonomies group are separated based on the following three attributes (Fig. 2.4): SoS Objectives, Governance, and the inter-relationship between CS (Collins, Doskey, and Moreland 2016). On one side, when CS collaborate for their interest, the overall objectives emerge alongside the

[9]https://www.waze.com

collaboration, they are rarely known beforehand. In this case, the resulting SoS is virtual, and it is a result of the interaction between its CS. On the other side, a SoS that is built from several independent systems, to serve a specific client, with well known objectives and management, that governs the interactions of its CS, is a directed system. Note that at any time SoS can change from one group to the other once a change happens to the management style adopted within.

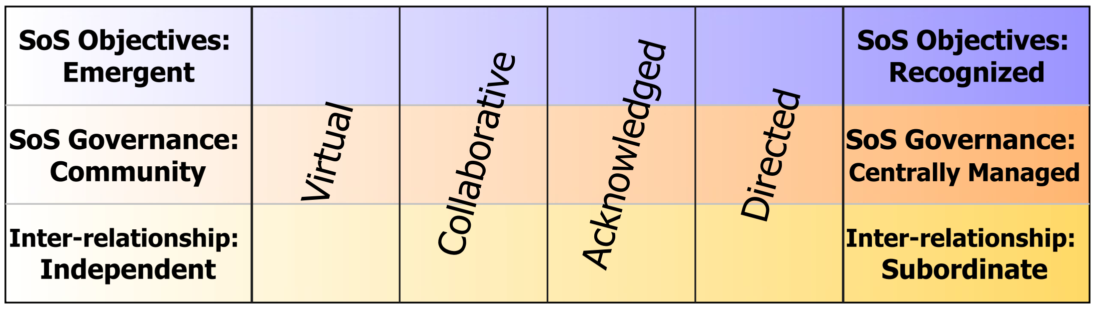

SoS Objectives: Emergent
SoS Governance: Community
Inter-relationship: Independent
Virtual
Collaborative
Acknowledged
Directed
SoS Objectives: Recognized
SoS Governance: Centrally Managed
Inter-relationship: Subordinate

Figure 2.4: Objectives, governance, and inter-relationship between CS are the 3 attributes that could be used to separate management taxonomies(Collins, Doskey, and Moreland 2016).

2.5.2 Other Taxonomies

Other than management taxonomies which describe the management style used in a specific SoS, there are 2 other groups of taxonomies that may be used to describe a SoS (Garnier 2018): anticipation taxonomies, and contracting taxonomies. The first group describes the degree of anticipation of engineering activities in the SoS, and the second describes the degree of integration of CS in the SoS.

2.5.2.1 Anticipation Taxonomies

The different taxonomies in this group describe the degree of anticipation of engineering activities such as refinement and maintenance in the SoS during its life cycle. 4 terms are used in this group: permanent, episodic, prepared, and phased SoS.

Permanent SoS, such as smart cities, are SoS that exist and are not expected to end, and they evolve during operations. The engineering effort in such SoS is more focused towards integrating new systems and technologies in a preexisting SoS.

Episodic SoS such as crisis management SoS, is SoS that gets deployed on demand, where the CS are prepared for typical scenarios, in such SoS, engineering activities such as monitoring are needed only when the SoS is deployed, and feedback and refinement when the SoS finishes its task.

The third type is **prepared SoS**, which is similar to the permanent SoS in that they both do not end, however, in this case, a prepared SoS does not exist before the engineering activities begin. After deployment, these SoS evolve and is maintained during operations. Air traffic management is an example of a prepared SoS.

The final type in the anticipation taxonomies group is the **phased SoS**. In contrast to episodic SoS, phased SoS are operational and do not get deployed on demand, however, there are several operational configurations that exist in the SoS, which is instantiated when required. A search and rescue SoS is a phased SoS where the involved CS choose the operational configuration based on the mission they want to perform.

2.5.2.2 Contracting Taxonomies

Contracting taxonomies are separated based on the way CS perform operations within the SoS. Similar to the 2 previous groups, there are 4 elements in the contracting taxonomies group: capability-based, function-based, service-based, and resource-based SoS.
Capability-based SoS function through capabilities provided by the CS. In large organization for example, companies contract other companies to provide capabilities instead of developing their own capabilities (hiring capabilities for example). This means that a particular CS will provide particular capability in the SoS as long as this capability is required. Whereas **function-based SoS** are SoS in which CS provides functions to the SoS/other CS. Just as with application-based configured systems (mainly heavy IT systems), different CS are contracted for specific functions they provide to the SoS, while at the same may be providing functions to other SoS. The third taxonomy in this category is the **service-based SoS**, where just as the previous 2 taxonomies, CS are provided as services to the SoS/other CS.
It might seem that the 3 taxonomies are similar since the terms capability, function, and service are similar and refer to something provided from a provider entity to a receiver entity. However, these 3 terms are not synonyms, and there is a slight difference between them: a capability is the ability to do something, whereas a function is what something does or is used for, and finally a service is doing the work for somebody.
The fourth and last taxonomy in this group is **resource-based SoS**, in which the interaction between CS is done by producing and consuming resources. These resources could be tangible (e.g. smart electric grids), or intangible (e.g. data). In this case, the most of the effort is focused towards enabling the exchange mechanisms between SoS/CS.

2.5.3 Discussion

The different taxonomies used to describe SoS are grouped into separate groups based on the aspects they describe: Management taxonomies describe the management style used to manage a particular SoS (Section 2.5.1), whereas participation taxonomies describe how CS participate in the SoS's operations. Regardless of what group of taxonomies we are interested in, it is important to identify the taxonomy that describes a SoS of interest accurately, because every taxonomy offers different approaches that must be followed in this SoS. To explain this we will describe the case of disqualifying SoS in the management taxonomies group. Consider the case of the internet. The internet is a collaborative SoS, in which the Internet Engineering Task Force[10] (IETF) is responsible for placing standards, but it has no power over applying these standards. It is up to the players (service providers, equipment manufacturers, etc.) to accept or reject a specific standard. A misclassification of the internet as a directed SoS for instance, gives the IETF an illusionary level of control that is much higher than the real control it exercises over the SoS, and in consequence its failure in attaining its objectives. The same goes to the other groups of taxonomies that may be used to describe a SoS.

[10]https://www.ietf.org

2.6 SoS Examples

Adopting an SoS approach to a certain situation is beneficial, and it could be seen from the diversity of application domains in which SoS approaches are used. SoS approaches are being used in healthcare [(Whittington and Dogan 2016), (Grigoroudis, Kouikoglou, and Phillis 2012)], in environmental applications (Butterfield, Pearlman, and Vickroy 2008), in infrastructure management (Otto et al. 2016), emergency and security management [(Liu 2011), (Mauss et al. 2015), (Daniel et al. 2009)], and power management and the use of smart grids [(Marvasti et al. 2014), (Arnautovic, Svetinovic, and Diabat 2012)]. Below we detail 2 examples of existing and underdevelopment SoS, from 2 domains: transportation, and economics.

2.6.1 U.S. Next Generation Air Transportation System

The Next Generation (NextGen) (Fig. 2.5) Air Transportation System (ATS)[11] is a project led by the U.S. Federal Aviation Administration's (FAA) that seeks to modernize the U.S. ATS, and it has the following 6 objectives (Planning and Office 2007): (1) Retain US leadership in global aviation, (2) expand capacity, (3) ensure safety, (4) protect the environment, (5) ensure US national defense, and (6) secure the nation. There are multiple reasons that motivated this project: congestion problems in the most frequented airports that are gonna worsen with increased demand, security issues that arose after 2001 events and led to tighten security procedures on travelers, which also contributed to the congestion and delays problems. Furthermore, environmental concerns from CO2 emissions, urban pollution and noise resulting from increasing the capacity of airports; and lastly, the need to preserve the leading position of the US ATS in the world, were all reasons behind the need for a new ATS that could achieve the previously stated objectives. Achieving those objectives is a very complex and time consuming task for a lot of reasons, including:

- ATS is governed by several national and international organizations (i.e. airport authorities and International Civil Aviation Organization) that have each different goals and objectives.
- Technologies intended to be used in NextGen are new technologies that need to be integrated into the existing systems (the shift from ground controllers to ADS-B[12] technology for example).
- The different constraints that must be taken into account such as environmental constraints and security constraints.

So how should the FAA approach the situation in order to reach a reliable and sustainable ATS? The first step would be to understand the type of system and environment in which they plan to integrate NextGen. The ATS is composed of airports, national organizations such as the National Air Traffic Controllers Association (NATCA) and different airlines to cite a few. All of these organizations, unions, airports and so on are very independent systems, which have their own management, resources and operations. They are operationally and managerially independent, geographically distributed, and in constant evolution. However, even with this independence, they choose to work together because they recognize that by

[11]for a more detailed case study of NextGen see (Gorod, White, et al. 2014)

[12]Automatic Dependent Surveillance - Broadcast.

cooperating, they could achieve more (this is the case for example of airline alliances such as skyteam[13] and star alliance[14]). So according to Maier's characteristics and most of the definitions, US ATS is a SoS. Furthermore, since every player in the system has its own funding, decision-making process, and goals, and since they share some of the objectives and goals which were not imposed by any single authority, this SoS is a collaborative SoS.
Let us suppose now that the FAA, in the face of this SoS, disregarded its type, and intended to treat this system as a directed SoS, in which they ignored all the players and proceeded to impose regulations and rules, without consulting those players. What would happen is that players will choose to ignore all regulations that do not suit them, and they might as well oppose to those regulations (by union strikes for example), and in the end, the NextGen project would not see the light. On the other hand, if the FAA recognizes that the ATS is a collaborative SoS, and that it has limited power over the players, it would approach the situation in another way (and that is in fact what has happened), and it would devise solutions and come up with regulations that help achieve the NextGen project, while at the same time satisfying all interested parties.

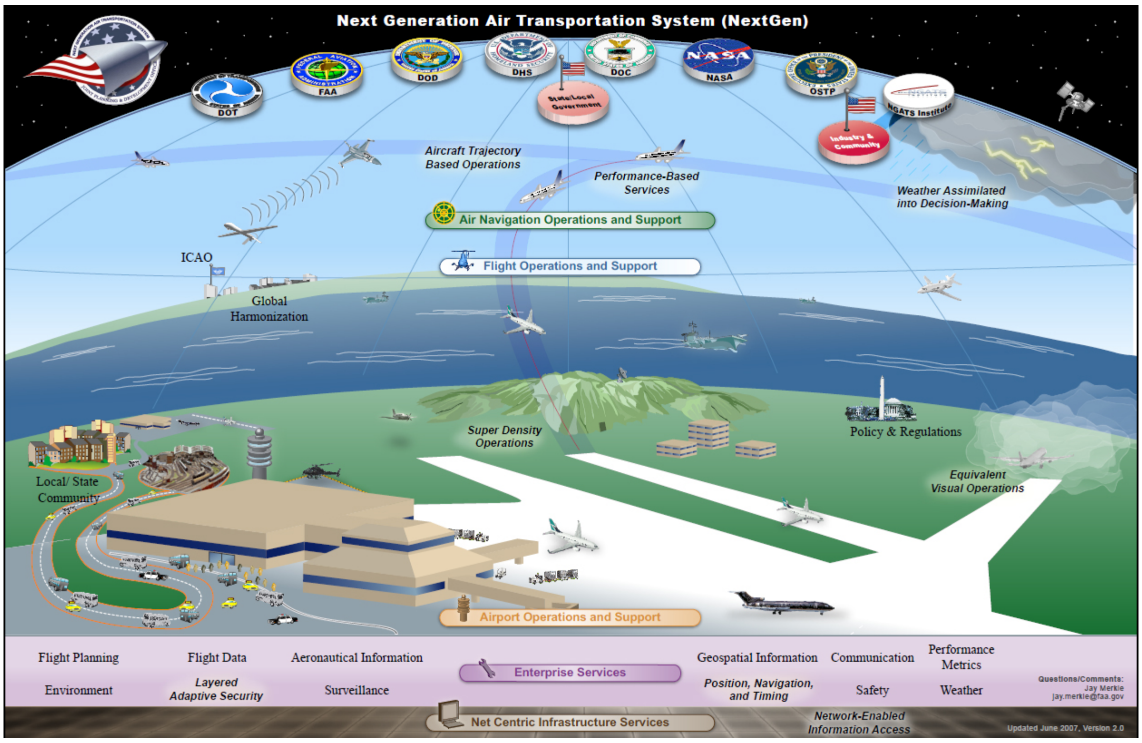

Next Generation Air Transportation System (NextGen)
DOT
FAA
DOD
DHS
DOC
NASA
OSTP
State/Local Government
Industry & Community
NGATS Institute
Aircraft Trajectory Based Operations
Performance-Based Services
Weather Assimilated into Decision-Making
Air Navigation Operations and Support
Flight Operations and Support
ICAO
Global Harmonization
Super Density Operations
Policy & Regulations
Local/ State Community
Equivalent Visual Operations
Airport Operations and Support
Flight Planning
Flight Data
Aeronautical Information
Enterprise Services
Geospatial Information
Communication
Performance Metrics
Environment
Layered Adaptive Security
Surveillance
Position, Navigation, and Timing
Safety
Weather
Questions/Comments: Jay Merkle jay.merkle@faa.gov
Net Centric Infrastructure Services
Network-Enabled Information Access
Updated June 2007, Version 2.0

Figure 2.5: U.S. NextGen ATS as SoS: Multiple levels of operations, and a lot of stakeholders and regulations (credit: NASA)

2.6.2 SWIFT and Secure Financial Messaging

The majority of financial exchanges happens via banks both nationally and worldwide. A typical money transfers via banks (also known as wire transfer) process happens as follows (Fig. 2.6):

[13]www.skyteam.com/en
[14]www.staralliance.com/en/home

1. An initiator (sender) initiates a request at their bank (Bank 1) providing the receiver's bank account details, and the amount of the transfer;
2. Bank 1 asks an intermediary bank (Bank X) to debit their account and credit the receiver's bank (Bank 2) account;
3. Bank 2 receives funds from Bank X and credit it to the receiver's account;
4. Finally, the receiver gets the transferred money.

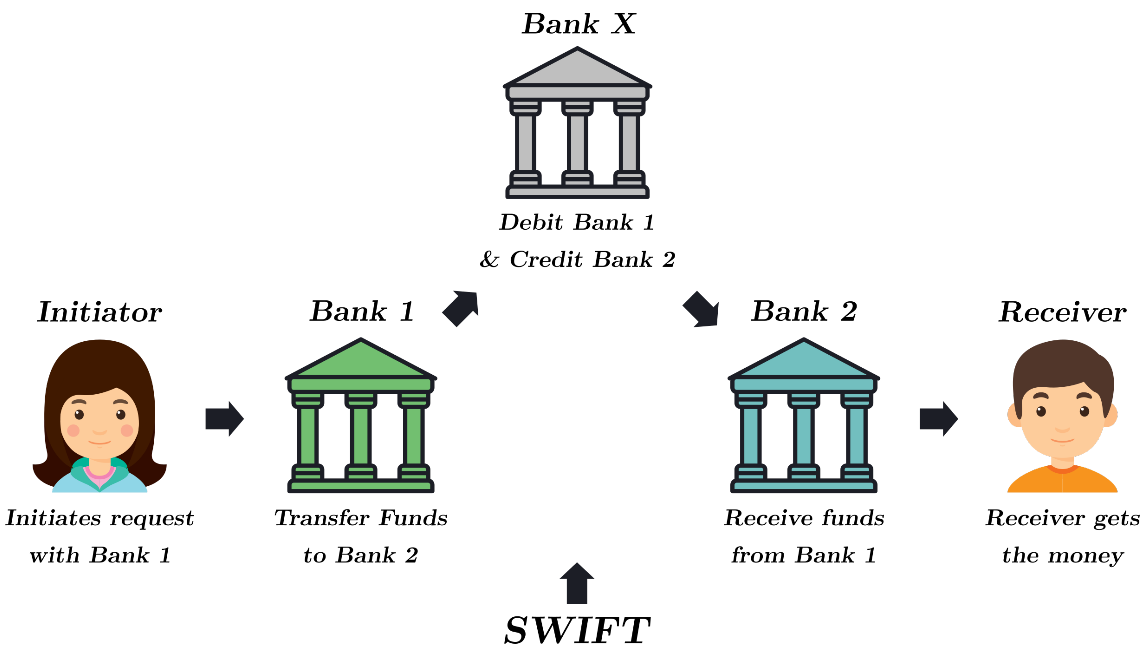


Figure 2.6: International Money Transfer: banks transfer funds between sender and receiver.

Notice that this exchange involved no actual money transfer between Bank 1, Bank X, and Bank 2, but rather the process involved digital messages exchanged online between them. In this example we are going to focus on this part of the process: financial communication between banks. In order to reach a successful exchange, the involved banks must use a common language and secure communication means. And this is where SWIFT comes to the picture. The Society of Worldwide Interbank Financial Telecommunication or SWIFT is a cooperative society owned and controlled by its shareholders , which operates a global network to facilitate the transfer of financial messaging[15]. SWIFT also provides standards together with ISO for message types used in financial exchanges (ISO 15022), for assigning bank identifiers (ISO 9362), and on a separate effort, the Costumer Security Program (CSP) which defines the security mechanisms that must be adopted in banks to be allowed to use SWIFT network. What is important from a SoS perspective about SWIFT is analyzing the SoS that emerged from this society. SWIFT was founded in Brussels in 1973, by 239 banks from 15 countries in the purpose of solving the communication problem mentioned previously. This is the first evidence of collaborative SoS that has formed to improve the state of financial messaging between its different CS.

[15]https://www.swift.com

Each CS (member bank) is an independent system, that serves its clients. Working in an isolated environment, a bank would not need to implement systems and technologies that allow them to communicate with other banks. However, banks do not work in an isolated environment for a number of reasons, one of which, the implicit relation that exists between different bank clients, where a client of Bank 1 wants to transfer funds to clients of Bank 2.
Furthermore, over the years, SWIFT has also become an important player in security measures used in member banks, because security is an important requirement of financial messaging. This new objective to this SoS led to the emergence of a new SoS, which involves companies that provide security solutions. These companies develop security solutions using SWIFT standards, and are allowed to generate certifications on behalf of SWIFT to certify that a certain bank respects SWIFT recommendations, and thus, it is allowed to use its network.

2.6.3 Discussion

It is very important to identify and study existing SoS because it gives us insights on how to better design, build, and maintain future SoS. The most important decisions regarding SoS under development are made during design time. During that time, the majority of emergent behaviors that might show in the SoS during operations are unknown, and there are several different design choices that are available to choose from (management style, technologies to be used, the nature of the relation between the SoS and its CS, etc.). Often simulations are the go-to solution used to make decisions during design time, but the accuracy of simulations depends on the accuracy of the model used to simulate the SoS, which is often incomplete. But having a live example to learn from and extract guidelines and lessons is invaluable, which is why studying existing SoS is important. However, one should be careful when doing this as not to miss classify their SoS and study the wrong SoS, because as we have seen misclassification is a serious issue in SoS.

2.7 SoS Engineering (SoSE) vs. Systems Engineering (SE)

So far we have seen that SoS are systems, but they possess some characteristics that separate them from other types of systems. The differences between SoS and other types of systems are so radical that the traditional systems engineering (SE) discipline is not enough to engineer SoS. The differences between traditional SE and SoSE are shown in Table 2.7 (Gorod, Brian Sauser, and John Boardman 2008). In a system, the parts are chosen as to achieve one objective (or set of objectives). These objectives are specific to this system, and all parts of the system will work to achieve these objectives. Whereas in SoS, CS have already their proper objectives. They belong to the SoS because of the common goal, but whenever the conditions of belonging to the SoS conflict with their interests, they might leave the SoS, thus, when we build a SoS, one important aspect is to assure that CS have reasons to work in this SoS, i.e. they share the same objectives. In traditional SE, this is not the case, because the CS are only parts, their only purpose is to work inside the system. The evolutionary development characteristic of SoS is also a source of major difference. Traditional systems have a specific life cycle that is well defined during the developments of the system, with clear information about the boundaries

	SE	SoSE
Focus	Single Complex System	Multiple Integrated Complex Systems
Objective	Optimization	Satisfaction, Sustainment
Boundaries	Static	Dynamic
Problem	Defined	Emergent
Structure	Hierarchical	Network
Goals	Unitary	Pluralistic
Approach	Process	Methodology
Timeframe	System Life Cycle	Continuous
Centricity	Platform	Network
Tools	Many	Few
Management Framework	Established	?

Figure 2.7: Systems Engineering vs. SoS Engineering (Gorod, Brian Sauser, and John Boardman 2008)

in which this system can operate. The response of the system to any changes in the environment is simulated and known. In SoS, the system faces two types of changes, exterior changes from the environment in which the SoS works, and interior changes that occur whenever systems join or leave the SoS, or whenever the objectives of the SoS change. Those changes usually can't be accounted for when constructing the SoS, and so, instead of focusing on the response of the SoS to changes in the environment, just like in traditional SE, SoS constructors must take interior changes into account as well. Comparisons between traditional systems and SoS, in (Gorod, Brian Sauser, and John Boardman 2008), (Mo Jamshidi 2008), and in (U.S. DoD 2008), show that SoS, with their differences over traditional systems, require a new engineering approach to address the challenges that face SoS. This new approach, which is starting to gain more and more attraction in the field, should take the different challenges that face SoS into account (discussed in the next Section 2.7.1).

2.7.1 SoS Challenges

The development of reliable and durable SoS must take into account the eventual challenges that will face these SoS during design and deployment. Some of these challenges come from the fact that SoS are complex systems and thus inherit the same challenges, others are specific to SoS due to their nature and their close interaction with their environment. Here we list some of these challenges to highlight the difference between SoSE and SE.

2.7.1.1 Social technical Equilibrium (Maier 2005)

SoS involves beside the technology, a human factor that plays a very important role in the SoS. The majority of CS are managed by humans, and in other cases, humans are the clients that will use the technology of the SoS. Designing a SoS, means studying the technology and choose the best option that suits SoS objectives. That is why it is very important to study the social factors that affect the SoS.

2.7.1.2 SoS Management (U.S. DoD 2008)

The managers of SoS do not have control over the CS, and stakeholders in the SoS have interests beyond the SoS. Despite all of that, SoS managers have to manage the SoS in a way that satisfies all stakeholders, and still achieve the SoS objectives. Note that CS are the systems that belong to the SoS, while stakeholders are all systems that do not belong to the SoS, but they are either affecting it or affected by it.

2.7.1.3 Emergence (U.S. DoD 2008)

Emergence is a key characteristic of SoS and was discussed in the Section 2.3.1. All SoS are built for their emergent behavior, but emergent behaviors in SoS are not always desirable. The challenge with emergence behaviors in SoS is to treat predictable undesired behaviors, and to predict unpredictable undesired behaviors in the system before deployment, to prevent system failures and even destruction.

2.7.2 Discussion

SoSE is a rising discipline that aims to guide the development of SoS just as SE guides the development of traditional systems. The development of this discipline is a process that involves people from both the academic and industrial worlds. IEEE Systems of Systems Engineering Conference[16] is a yearly conference dedicated to showcasing the SoS community's effort towards establishing SoSE discipline, which involves researchers from universities and research centers, as well as managers from a wide range of industries. On a parallel effort, the International Council on Systems Engineering (INCOSE) has dedicated a working group for this purpose (Systems of Systems Working Group[17]), to advance and promote SoSE. Furthermore, several books have been published that were either fully dedicated to SoSE, or tackle the subject as part of the SE discipline [(Mo Jamshidi 2008), (Luzeaux and Jean-René Ruault 2010), (Sage 2011), (Luzeaux, Jean-Rene Ruault, and Wippler 2013), (Dickerson and Mavris 2016)].

2.8 Conclusion

SoS are large complex systems, that result from the interactions between different independent systems. In our connected world, SoS emerge naturally due to the increasing interactions between independent systems, and the service-based business models adopted by various systems. Since they are the result of complex systems interacting, SoS are themselves complex systems, and thus they inhibit emergent behaviors just as any complex system does. SoS are dynamic systems in the sense

[16]http://sosengineering.org

[17]https://www.incose.org/incose-member-resources/working-groups/analytic/system-of-systems

that CS might join or leave the SoS during operation time, add to that their exposure and close interactions they have with their operational environment, and we are in front of complex systems that face challenges typically not observed in traditional systems. And that is why developing a SoSE discipline is important to engineer and build better and more resilient SoS. Several taxonomies might be used to group different SoS, depending on what aspects of the SoS are we interested in: management taxonomies group SoS based on their management style, whereas anticipation taxonomies describe the degree of anticipation of engineering activities in the SoS during operations, and finally contracting taxonomies refer to the way SoS acquire their CS, and how the CS contribute to the SoS. The importance of SoS can be proven by looking at the diversity of SoS applications across different domains: SoS could be found in healthcare applications, economics, defense and security, and in weather monitoring, etc. And that is not shocking since SoS the natural result of different systems operating in the same environment.

2.9 What's Next?

The key aspect to achieving a reliable SoS is by acquiring CS, and controlling them in a way that the local actions on the CS level combined, result in a desired and predictable behavior on the SoS. This means that CS must know what to do, how to do it, and when to do it. The engineering branch that deals with this aspect of system is systems control, but since SoS requires a new engineering discipline to extend the traditional SE, does they also require the same thing regarding control? This will be the subject of the next chapter entitled: Traditional Systems & SoS Control.

Chapter 3

Traditional Systems & SoS Control

3.1 Introduction

We have touched a little bit about the topic of system management in the previous chapter, and how do systems know what they should do. We have also mentioned that systems have a predefined behavior that is set to tell systems how to do what they should do. In this chapter, we are going to talk in detail about this behavior, and how systems control their parts in order to achieve the overall desired behavior. In the context of systems control, we make the distinction between control law and control paradigm. A control law is the mechanism used inside a controller to make a decision based upon its knowledge of the environment, whereas a control paradigm is how the controllers will be distributed in the system in order to achieve the desired behavior while respecting the requirements of the system. After discussing the problem of control for independent systems, we are going to tackle the problem of control in SoS: how does SoS authority control the CS in order to achieve the global goals of the SoS?
This chapter is divided as follows: Section 3.2 introduces the problem of systems control in traditional systems, and the concepts of low and high level control. In Section 3.3 centralized and non-centralized paradigms are explained, with the different approaches adopted in the non-centralized paradigms, alongside some examples illustrating key points. Later on, Section 3.4 describes the problem of control in SoS, and highlights the difference between traditional systems control and SoS control. Finally, Section 3.5 concludes the chapter with a conclusion.

3.2 Traditional Systems Control

Systems are formed of multiple different parts, where each part is responsible for performing a task (or series of tasks) that is essential to the operation of the whole system, these tasks might be physical, i.e. moving something from point A to point B, or immaterial, i.e. computing certain values. Every part has parameters that describe its current states, a model that describes the relation between the different states, and describes their evolution over time, it has also inputs and outputs: inputs are sent to this part in order to produce desired outputs. The outputs of a part are then transmitted to others connected parts and play the role of either inputs or values that are used to produce inputs. Outputs are computed in what is called controllers. Figure 3.1 represents a typical controller system. Controllers usually

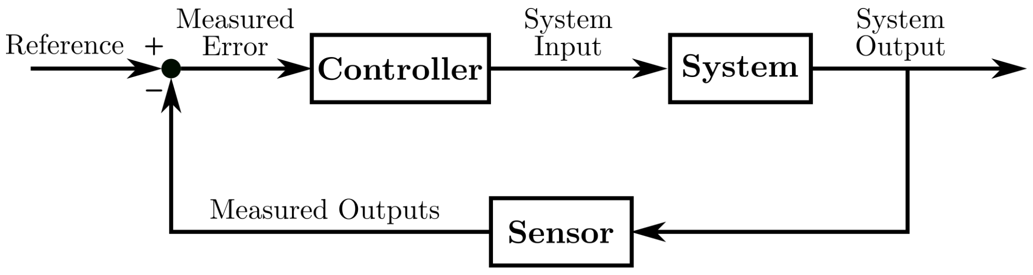


Figure 3.1: Typical system control: a controller computes the system input based on the error between the measured system output and the reference.

integrate mathematical models of the parts they control, and they compute the outputs of the parts using their models and the reference values on the outputs, or what we call also desired outputs. The content of controllers, or their logic in producing their outputs is what we call the control law. There are a myriad of control laws that are used to produce the desired outputs such as proportional-integral-derivative control (PID control) (Åström, Hägglund, and Astrom 2006), sliding mode contol [(Shtessel et al. 2014), (Azar and Zhu 2015)], optimal control [(Macki and Strauss 2012), (Bryson 2018)], etc. The choice of a particular control law to be used in a controller depends on the dynamics of the system, the desired behavior, and other factors.
On the other hand, the distribution of controllers in parts, how they communicate, and what information is shared between different controllers is what we call control paradigm. Sometimes a single controller controls all parts, and sends their inputs which are then transformed into outputs, in other cases, a system may have multiple controllers where each controller is responsible for a specific task in the system. There are numerous factors that define the choice of a particular paradigm in a system: the nature of the system, its size, the required computational capacity , etc. Systems control is a task that has two layers, defining the control paradigm in the system, and/or defining control methods that will be used inside controllers so that the system achieves its objectives, while having the desired behavior as well. The following example (example 3.2.1) illustrates the problem that we try to solve in control:

Example 3.2.1. *The system in Fig. 3.2 is a water boiling system, the objective of this system is to heat water to a certain desired temperature $T_{water/desired}$ to achieve a desired radiator temperature $T_{desired}$. It is composed of a burner, pipes in which water flows, and a radiator. To describe the behavior of this system, we need to measure the following properties, or states: the quantity of fuel burnt in a time δt_{burn}, the amount of air used to feed the fire in a time δt_{air}, and the temperature of water inside the tubes in the radiator $T_{water/radiator}$. Using these 3 measurements, we could say that using x amount of fuel per δt_{burn}, and y amount of air per δt_{air} , we could expect to reach a temperature of $T_{desired}$ in the radiator for z period of time.*
First question: what control method is the most adapted to this particular system in order to control its different states and achieve the desired behavior?
Furthermore, this system contains multiple actuators that translates controls into

[1]www.microgreening.com

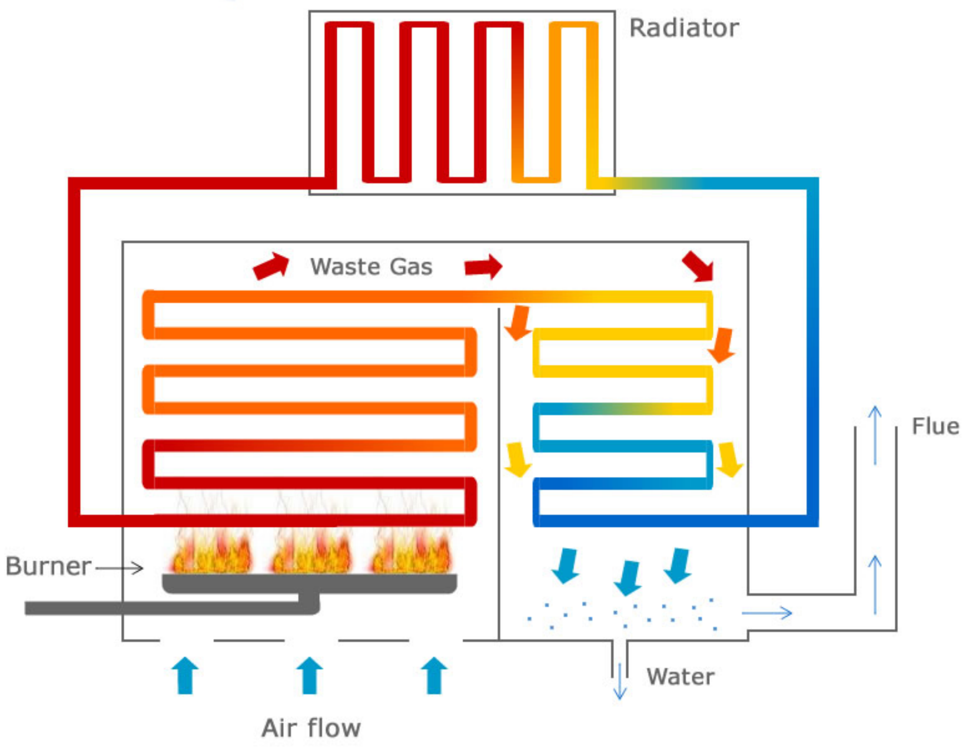


Figure 3.2: Water Heater Scheme[1].

actions in the parts. A fuel valve is responsible for controlling fuel intake into the burner, an air valve that controls the air flow into the burner, and a gate motor that controls the opening of the radiator from which we take the heat. There are multiple possible ways that this system could be controlled; We could control both valves (fuel and air) with a single controller, and the radiators opening with another one, or we could control the 3 actuators using 1 controller. Second question: what is the best paradigm to be adopted in this system to achieve the desired behavior?

The answers to the previous questions are provided by the discipline of systems control. Systems control in systems engineering is the application of automatic control theory to produce systems with desired behavior in control environment (Kilian 2006). In other words, systems control tells us what is the best control law to use in order to control a specific system, and what controller-actuator architecture is best suited to this system. The focus of this chapter is the different control paradigms that may be adopted in a system, however, before discussing paradigms, there are some notions that we must explain: low and high level control.

3.2.1 Levels of Control

3.2.1.1 Low-Level Control

Usually systems control is treated from the point of view of controllers, i.e. what are the states that we want to control and how to design controllers to do so. So in the previous example (example 3.2.1), when we tackle the subject of control, we think about the models of parts, how they are connected, and what is the best control method to be used in order to achieve the best performance from the system. This part of control is what we call “low-level control”, i.e. the final control boundary between the management of the system and its operations. If we go back to the figure 2.3 from chapter 2 which represents a generic representation of an independent system (operationally and managerially), low-level control is what occurs in the operational level, where the controllers get their reference values, or the desired mode of operation and translate it to commands sent to the actuators. However, the operational boundaries of systems are wider than the actual operation a system performs; the boiler in the previous example is used to boil water. However, it is capable of heating the water to a specific temperature as well. So even though the normal mode of operation of this boiler is when water is $100°C$, it could very well operate in the range of $10°C \rightarrow 120°C$. But who decides on the desired operational mode at time t? Well, this is where high-level control comes into play.

3.2.1.2 High-Level Control

High-level control is the process of choosing the best operation mode of a system, which occurs in the managerial level of figure 2.3. In high-level control, the operational level is regarded as a black box, which accepts the desired operational mode as input, and provides the required capabilities as outputs. In order to better understand the differences between high and low-level controls, let us expand our previous example of the water boiler and consider its use in a room heating system where the heat of the water is used to heat a room via the radiator (Fig. 3.3). In this case, the occupants of the room control the desired temperature of the room, which then translates into a desired temperature of the radiator, to arrive finally to the flow of fuel and air in the burner (fig 3.4). In this chain of relations, the choice of the room temperature is the reference value to the whole boiler system, and this is what we call high-level control. As far as it concerns the room occupants, the whole heating system is a black box, they do not know what are its parts, neither how it works. For them to control the room temperature (and the heating system in consequence), they need to provide the desired room temperature only, and the system will take care of the rest. In short, low-level control is what is applied to the actuators in order to translate the commands into actions, and high-level control is when the objectives of the system (or the parts) are defined.

3.3 Control Paradigms

Systems contain multiple parts that have similar or diverse sets of tasks to be performed in order to achieve the desired behavior of the system. Each part (or group of parts) is controlled via one or multiple controllers that send their inputs, which are then transformed into outputs (physical or immaterial). Before choosing the right control law that will be used inside each controller, system architectures must first choose the number of controllers, the responsibilities of each controller, and how these controllers are coupled, or what is called the control paradigm that the system will adopt throughout its operation. There exist two categories of control paradigms: centralized and non-centralized paradigms. As clear from their names, in centralized paradigms systems use a single controller to control all the parts, while in non-centralized paradigms systems use 2 or more controllers to do so.

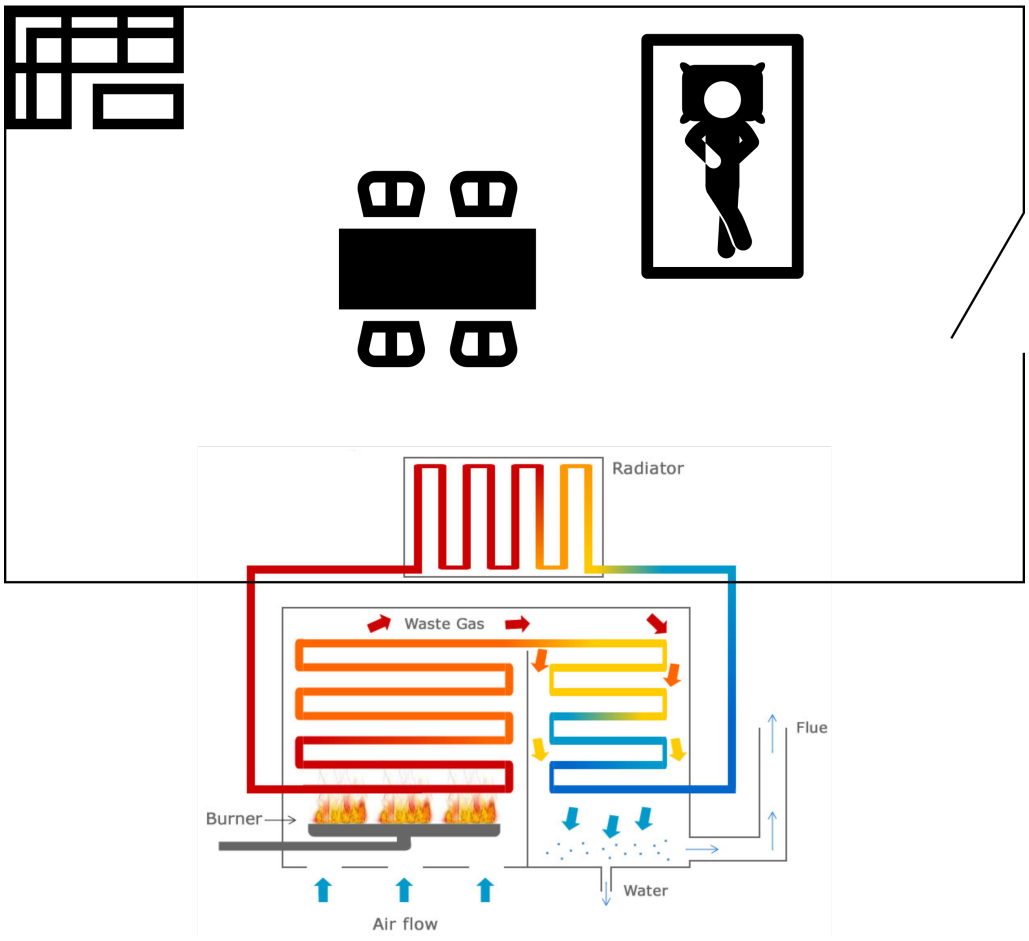


Figure 3.3: Boiler-Room system: room sends high-level controls to boiler, which are then translated into low-level controls inside the boiler.

3.3.1 Centralized Control Paradigm

A centralized control paradigm is adopted when the needs of the system on the control level could be met by using a single controller [(Díaz et al. 2017), (Zhou et al. 2017)]. In this case, the controller is connected to the different actuators and sensors in the system, it gathers all sensors readings, and adds to them reference values that are required by the system's management, in order to compute different inputs for different parts. This paradigm is usually adopted in small-scale systems, where the required computational capacity is moderate. Systems like the previously mentioned water heater, dish washers, and refrigerators all use centralized paradigms to control their parts to achieve their objectives. A centralized paradigm could be represented as shown in Fig. 3.5: a single controller C is responsible for gathering reference values and all sensors measurements $(y_1 \cdots y_n)$ from the different parts, and it computes the different inputs $(u_1 \cdots u_n)$ which are then used by the respective actuator A_x in order to achieve the desired operational behavior of the system. To demonstrate this paradigm let us take a close look on the water heater from the previous section (example 3.2.1 and fig 3.2). We are interested in its parts for now, of which we have the burner, the pipes, and the air flow intake. The actuators of each part are respectively a valve that controls the flow of fuel to the burner, another valve that controls the flow of water in the pipes, and a third valve that controls the air flow to the intake. In order to reach a certain temperature at the

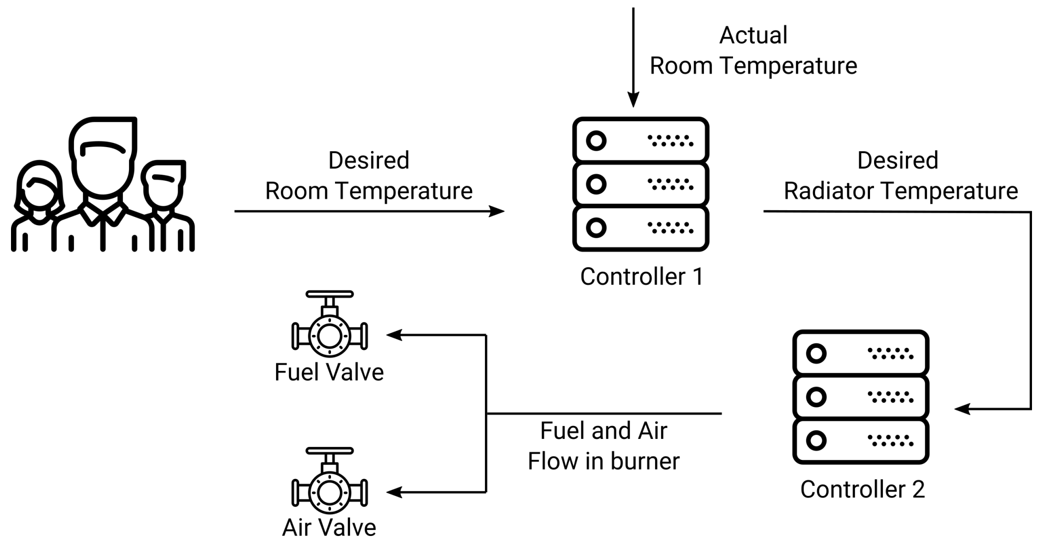


Figure 3.4: Boiler-Room control chain: from high-level to low-level control.

radiator each valve must be open in a precise value. According to what we said earlier, and assuming a centralized paradigm has been adopted, we could expect a single controller connected to each valve, connected as well to the thermometer at the radiator, and several other sensors in the system. This controller uses the model of the system, i.e. the different models of the different parts together, to compute for each valve an opening value that will be respected by the valve.

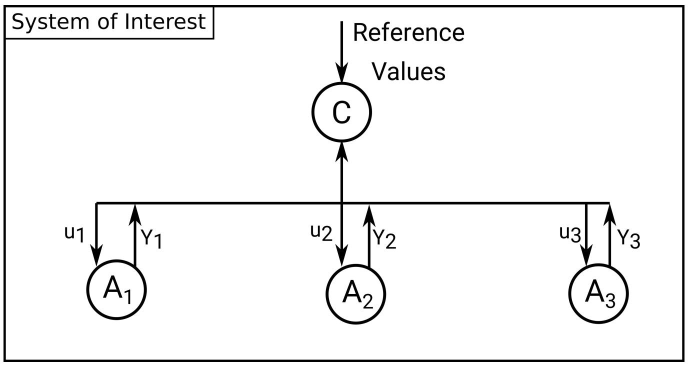


Figure 3.5: Centralized Paradigm: one controller and multiple actuators.

3.3.1.1 Discussion - Centralized Paradigm

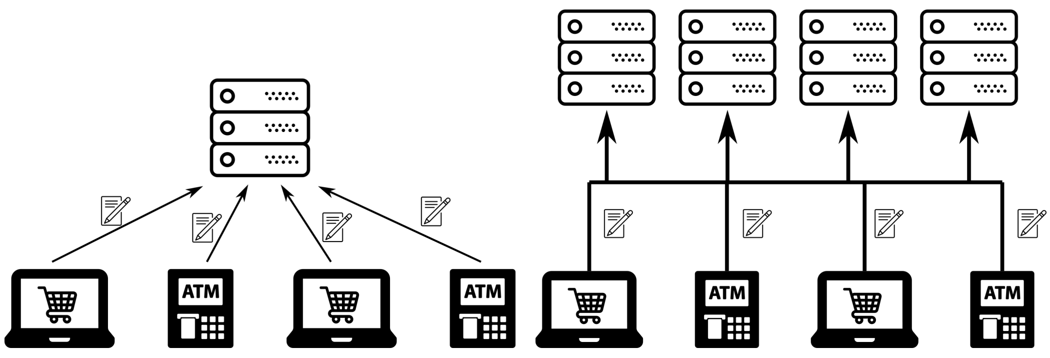
Just as the same as everything else in life, this paradigm has advantages and disadvantages. On the plus side, since in this paradigm a single controller is used to control the whole system, cost-wise this is a more affordable solution than to use 2 or more controllers. This depends, of course, on the computational capacity required by the system, which up to a certain limit, favors one controller. Above this limit, using single controller actually is not efficient cost-wise, since a single controller that could handle this load would cost much more than multiple smaller controllers, and therefore distributing the load on multiple controllers eventually leads to a lower overall cost. Another advantage is that by using one controller we are able to better control the system since we have all the needed values at one place. On the other hand, using a single controller may lead eventually in the case this controller stopped working for the failure of the system. This risk, also called a single point failure risk is very important to take into account when the failure of the system leads to disastrous consequences (nuclear power plant for example).

3.3.2 Non-Centralized Control Paradigm

Non-centralized control paradigms are adopted in systems either because they are non-centralized in nature, or to overcome some of the centralized paradigm. Decentralization is in the process of moving the authority from central to local[2]. A non-centralized system is a system in which parts have local authorities by themselves, in other words, parts take decisions based on local information without knowing the actual state of the global system. Systems that contain fleets of vehicles[(Vandermeulen, Guay, and McLellan 2018), (Dimarogonas, Frazzoli, and Johansson 2012)], or smart infrastructure [(Mahmud and Zahedi 2016), (Ruiz-Romero et al. 2014)] are decentralized systems in nature, whereas systems that use blockchains (Swan 2015) to secure transactions are decentralized by design to overcome the single failure point problem that we discussed in the centralized approach. In all of these systems, the central authority has delegated decision-making to the different parts that constitutes the system. Instead of having a central command center that has the knowledge of the states of the overall system, and issuing commands for every part like in centralized systems, in non-centralized systems parts integrate local rules or functions that they used to take decisions. The information available to the parts are local and they do not provide them with a global view of the state of the system, therefore the overall behavior of the system is an aggregation of the different behaviors of its parts. Whether a system is decentralized in nature or by design, a non-centralized control paradigm is adopted for controlling the system. To illustrate the difference between a centralized and a non-centralized paradigm, consider a simplified version of the transaction recording system used to record transactions in example 3.3.1.

Example 3.3.1. *A transaction recording system is used in banks to record transactions made by their customers. Every transaction made must be recorded and stored in a secure log to prevent any modification of the data. Every transaction is labeled with the amount, account number, and a transaction code indicating if its debit, credit, or other kind of transaction, and then it is added to the transactions log. The transactions log is used later-on to generate financial statements and for other important purposes. In this system, it is required that the logs are stored in a secure storage where they are safe from destruction or alternation. Figure 3.6 shows the two paradigms that the bank could adopt within the system (Fig. 3.6a and Fig. 3.6b). Centralized paradigm in Fig. 3.6a is adopted in which all transactions made by users are logged in a single central storage, whereas non-centralized paradigm in Fig. 3.6b is adopted in which transactions are sent to all data storage units or what is called a blockchain. Cost-wise, the centralized approach is more efficient than non-centralized approach because in the latter all storage units have the same copy of the transactions log, and therefore the amount of storage units present in the non-centralized approach is always higher than in centralized approach. However, if the system's designers value security and safety of the data more than cost, the non-centralized approach is a more secure architecture, because in order to alter data*

[2]en.wikipedia.org/wiki/Decentralization



(a) Centralized Paradigm **(b)** Decentralized Paradigm

Figure 3.6: Transaction recording system: centralized vs. non-centralized paradigm.

for example, one must change the data in all different copies, whereas in centralized approach it is enough to alter the data in the only storage unit.

There are mainly 4 different organizational shapes a non-centralized system may take [(Šiljak and Zečević 2005), (Scattolini 2009)]: hierarchical, distributed, and decentralized organizations. The fourth shape is a hybrid of two or more of the previous shapes. In hierarchical organizations, there are multiple levels of authority in a top to bottom distribution, where top level authority has more knowledge of the system and is responsible for more tasks than lower level authorities. Distributed systems are systems in which all parts have partial knowledge of the system, they share the same level of authority, and they communicate specific information to perform their tasks. The last shape, namely the decentralized organizations distribute tasks on parts in a way that gives each part a local role within the system, every part has a partial knowledge of the overall system, but unlike the previous shape, parts do not communicate information. In a hybrid organizational shape, the organization of parts is a combination of two or more shapes that were mentioned earlier, for example a hierarchical shape in which parts are grouped in a hierarchy, and inside each level groups of parts might use distributed or decentralized shapes.

3.3.2.1 Hierarchical Paradigm

Hierarchical systems get their names from the hierarchy they adopt in distributing tasks on parts of the system. In the top-down direction there are 2 or more levels in which parts belonging to a higher level have more authority and therefore knowledge of the system than parts belonging to lower levels. Parts belonging to the same level do not communicate, however, they are interconnected through an input/output relations: outputs of subsystems are input to other subsystems (Findeisen et al. 1980). Systems such as power plants and microgrids [(Vermillion et al. 2014), (Palizban, Kauhaniemi, and Guerrero 2014)], and enterprises (Vargas 2016) are examples of systems that adopt hierarchical organizational shapes. As illustrated in Fig. 3.7, on the lowest level of authority n, reside parts labeled as A_x which are responsible for the smallest tasks within the system. The knowledge of each part A_x is limited to what it is supposed to do. Going one level higher to level $n + 1$, we could see that parts on this level (C_x as in coordinator) have more authority in that they are responsible for sending reference values or objectives to their corresponding systems in the level n, and therefore they have larger knowledge of the system. At the highest level, level $n+2$, the supreme coordinator SC resides, which has authority and complete knowledge of the overall system. The process of distribution of tasks in the system, and which part is responsible for what task is very important and depends on the nature of the system itself as well as the architecture's decision in the system. In a factory for example, the choice of assigning the authority of the production line and product storage to the same authority is the choice of the system's architecture, on the other hand, on the production line itself, the task of filling the product is separated from the task of preparing the container since each task is performed physically by a different machine. The following example

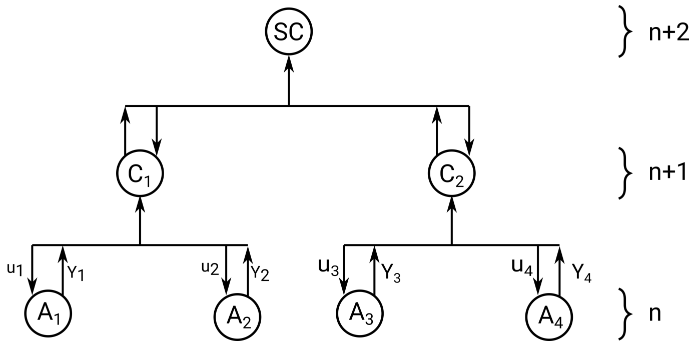


Figure 3.7: Hierarchical Systems: different levels of authority, no communication occurs on the same level.

shows how hierarchical control is applied to a large-scale system, which consists of multiple interacting subsystems. The method used (Rezaei and Jabehdar-Maralani 2012) is an improvement of the "Model Coordination Method", proposed initially by (Mesarovic, Macko, and Takahara 1970). This method consists of decomposing the global system into multiple subsystems, where the interactions between subsystems are parameterized on the subsystem level, where the objective is to find the optimal solution under those circumstances. The following example (Mo Jamshidi 1997) shows an application to this method on a 12 states large-scale system, which consists of 2 6-states subsystems.

Example 3.3.2. *Consider the following large-scale system, described by equation 3.1,*

$$x(k+1) = Gx(k) + Hu(k); \quad k = 0, \cdots, N-1 \tag{3.1}$$

where

$$G = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \tag{3.2}$$

$$H = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix} \tag{3.3}$$

with

$$G_{11} = \begin{bmatrix} 1 & 0.1 & 0.005 & 0 & 0 & 0 \\ -0.015 & 1 & 0.1 & 0 & 0 & 0 \\ -0.3 & -0.2 & 0.9 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.1 & 0.005 \\ 0.005 & 0 & 0 & -0.005 & 1 & 0.1 \\ 0.1 & 0.005 & 0 & -0.1 & -0.275 & 0.8 \end{bmatrix} \tag{3.4}$$

$$G_{12} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.005 & 0 & 0 \\ 0 & 0 & 0 & 0.095 & 0.005 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.005 & 0 & 0 & 0 & 0 \\ 0 & 0.09 & 0.004 & 0 & 0 & 0 \end{bmatrix} \tag{3.5}$$

$$G_{21} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.0045 & 0 \\ 0 & 0 & 0 & 0 & 0.085 & 0.004 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.005 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0.0047 & 0 & 0 & 0 \end{bmatrix} \tag{3.6}$$

$$G_{22} = \begin{bmatrix} 1 & 0.1 & 0.0045 & 0 & 0 & 0 \\ -0.0045 & 1 & 0.086 & 0 & 0 & 0 \\ -0.086 & -0.176 & 0.732 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0.1 & 0.005 \\ 0.005 & 0 & 0 & -0.0145 & 1 & 0.1 \\ 0.1 & 0.005 & 0 & -0.284 & -0.204 & 0.895 \end{bmatrix} \tag{3.7}$$

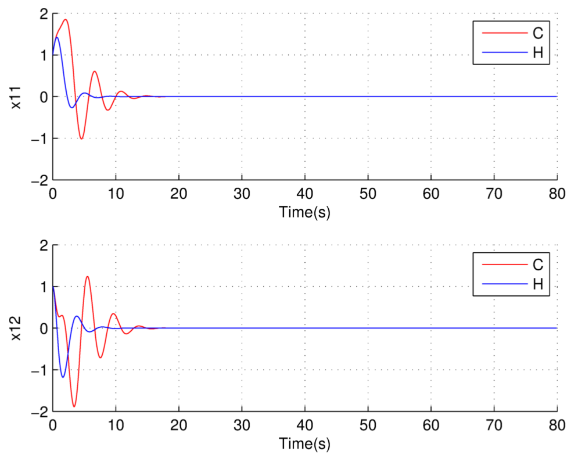
and

$$H_{11} = \begin{bmatrix} 0 & 0.005 & 0.1 & 0 & 0 & 0 \end{bmatrix}^T \tag{3.8}$$

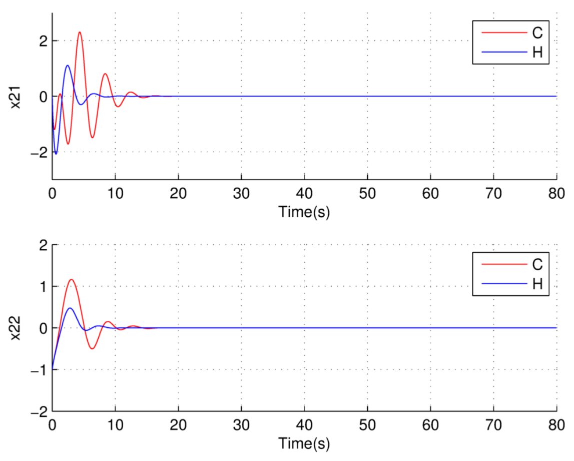
$$H_{22} = \begin{bmatrix} 0 & 0.005 & 0.09 & 0 & 0 & 0 \end{bmatrix}^T \tag{3.9}$$

$$H_{12} = H_{21} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T \tag{3.10}$$

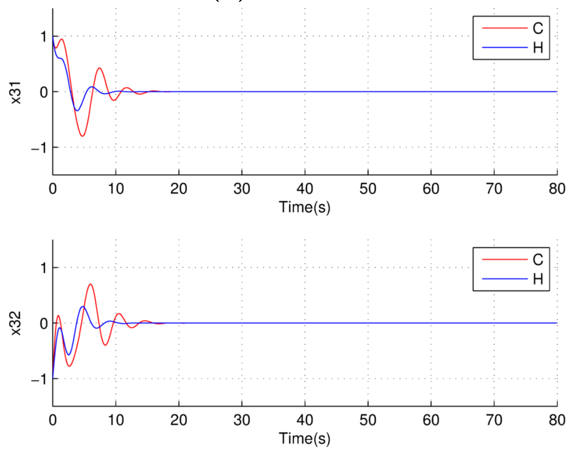
The vector $x(k)$ is the state vector at step k, and $u(k)$ is the input vector at step k. The matrices G_{ii} represent the dynamic of each subsystem $i, i = 1, 2$, whereas G_{ij} represent the interaction matrices that describe the relationship between subsystems i and j $(i \neq j)$. The method used control this system is taken from (Rezaei and Jabehdar-Maralani 2012), which uses the interaction prediction principle, and the model coordination method proposed by (Mesarovic, Macko, and Takahara 1970). The idea is to parameterize the interactions between subsystems on the subsystems level, where each subsystem computes the optimal solution for those parameterized interactions. Afterwards, the coordinator computes the real interactions between subsystems using their results. This process is then repeated until the error between the predicted and real interactions is less than a certain threshold ϵ. The algorithm that describes this method is the following (algorithm 1): For the final time $T_f = 80s$, sampling time of $T_s = 0.1s$, error threshold $\epsilon = 0.01$, $Q_i = \mathbb{I}_{(6\times6)}$, $R_i = 1$, and $S_i = 0.5\times\mathbb{I}_{(6\times6)}$; with $i = 1, \cdots, 6$, the results of the simulation using hierarchical and centralized methods are shown in figures 3.8a to 3.9b. As we can see, in some cases (Fig. 3.8a and 3.8b) the states reach equilibrium more smoothly in the hierarchical method, while other in other cases (Fig. 3.8c and 3.8d), the centralized method led to smoother transition state. However, it took 0.060295 *seconds for the centralized controller to stabilize the system, whereas it took* 0.095766 *seconds for the hierarchical controller to do so. Even though hierarchical control was slower to reach the overall stability, we should keep in mind that the computation load was distributed over two controllers in the hierarchical case, which have less capacity than the single controller of the centralized method.*



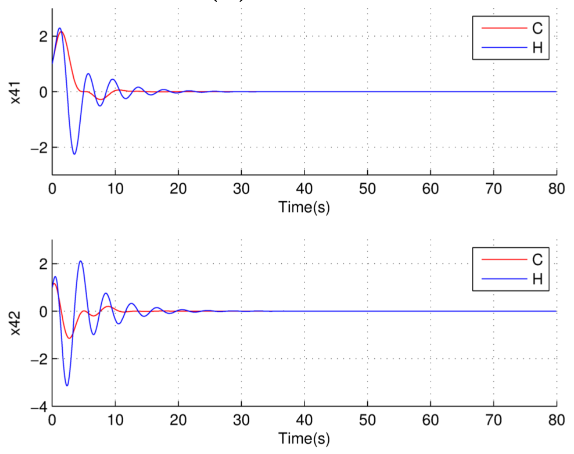
(a) State x_1



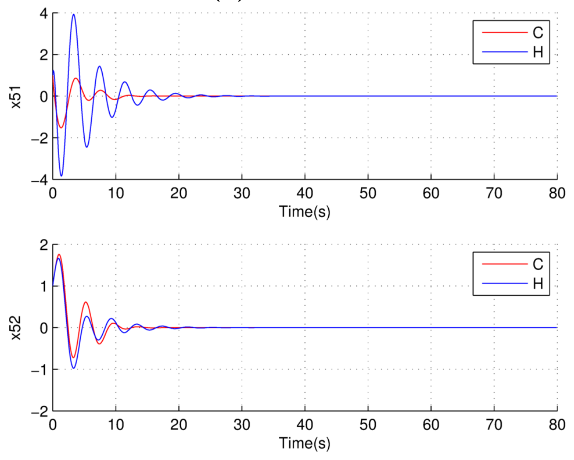
(b) State x_2



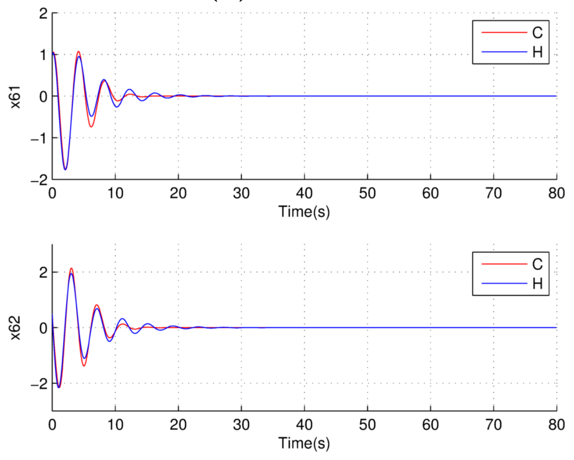
(c) State x_3



(d) State x_4

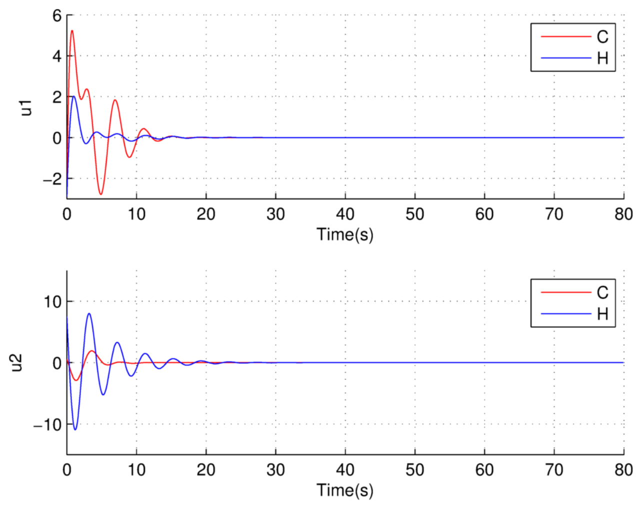


(e) State x_5

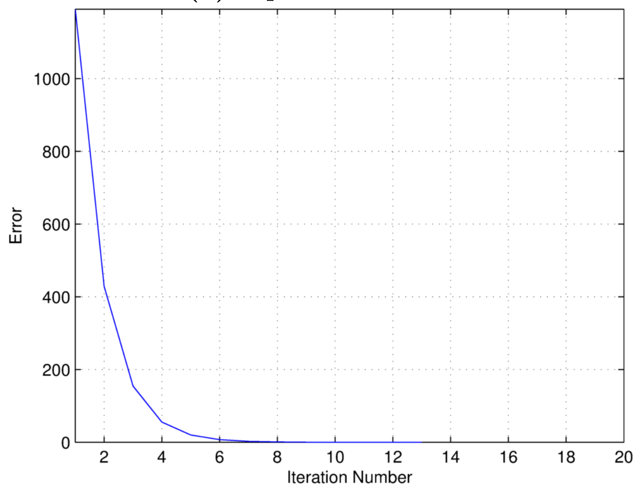


(f) State x_6

Figure 3.8: The states x_i of the simulated system.



(a) Inputs u_1 and u_2



(b) The error between the predicted interactions and the real interactions throughout the iterations.

Figure 3.9: The inputs u_i, and the evolution of the error throughout the iterations.

Algorithm 1 Two-level hierarchical control of a large-scale systems using interaction prediction principle.

```
 1: compute $K_i(k)$ and $P_i(k)$ with final condition $P_i(k_f) = S_i$ using:
        $K_i(k) = R_i^{-1} H_{ii}^T \big(G_{ii}^T\big)^{-1} \big(P_i(k) - Q_i\big)$
        $P_i(k) = Q_i - G_{ii}^T P_i(k+1) \big(\mathbb{I} + H_{ii} R_i^{-1} H_{ii}^T P_i(k+1)\big)^{-1} G_{ii}$
 2: store $K_i(k)$ for $k = 0 \to k_f$
 3: initialize the predicted interactions vector $z_{p_i}(k)$
 4: compute inputs $u_i(k)$ and states $x_i(k)$ for each subsystem using:
        $u_i(k) = -K_i(k) x_i(k) - (H_{ii}^T H_{ii})^{-1} H_{ii}^T z_{p_i}(k)$
        $x_i(k+1) = G_{ii} x_i(k) - H_{ii} K_i(k) x_i(k)$
 5: compute the real interactions using:
        $z_i(k) = \sum_{\substack{j=1 \\ j \neq i}}^{N} \big(G_{ij} x_j(k) + H_{ij} x_j(k)\big)$
 6: evaluate the overall error $e = Z - Z_p$ with:
        $SE = \frac{1}{2} e^T e$, where
        $Z = [z_1^T(k) \cdots z_N^T]$    and    $Z_p = [z_{p1}^T(k) \cdots z_{p_N}^T(k)]$
 7: if $|SE| \leq \epsilon$ then
 8:     stop
 9: else
10:     update the predicted interactions and repeat from step 4
```

3.3.2.2 Distributed Paradigm

Unlike hierarchical systems, distributed systems do not have different levels of authority, but rather parts take decisions based on their perception of the environment, and the information they get from other parts in the system (Christofides et al. 2013). The majority of Multi-agent Systems (MAS) (Olfati-Saber, Fax, and Murray 2007) adopt this organization: a surveillance fleet of drones and a fleet of autonomous delivery vehicles are examples of such systems. Information systems that perform huge computations or tasks might also adopt such paradigm (Hashem et al. 2015), where the main task is distributed over multiple computation units, and the overall result is an aggregation of the different results obtained by the different parts. In this paradigm, the global task is decomposed and distributed over the different parts just as in hierarchical paradigm, however, unlike hierarchical paradigms, in distributed paradigms controllers share some information with other controllers in order to make decisions. In distributed paradigms each part has partial knowledge of the global system that is sufficient for it to make decisions and perform its tasks. What differentiates this paradigm from the other two non-centralized paradigms is the communication that occurs between parts. Whereas in hierarchical paradigms, no communication occurs horizontally (between parts of the same level), and in decentralized paradigm parts do not communicate at all, in distributed paradigm parts communicate. The distribution of tasks and the design of controllers in such paradigm is done in such a way that parts will use information from other parts in order to make better decisions, and so communication is key in this paradigm. Which part share which information to which other part(s) depends on the nature of the system and the distribution of tasks. Figure 3.10 shows an example of a distributed system, composed of parts $A_1 \to A_6$. The unidirectional arrows indicate a unidirectional communication while the bidirectional arrows indicate a bidirectional communication. For example, communication that occurs between A_1 and A_4 is unidirectional, where A_1 sends information to A_4 which are later on used by A_4 alongside other information to make a decision, whereas between A_3 and A_4 a unidirectional communication occurs, in which both parts send and receive information from/to the other part. The stability of such systems is often studied using graph theories, where parts are represented by nodes, and communications by connectors between nodes. A lot of applications use distributed paradigms, most notably,

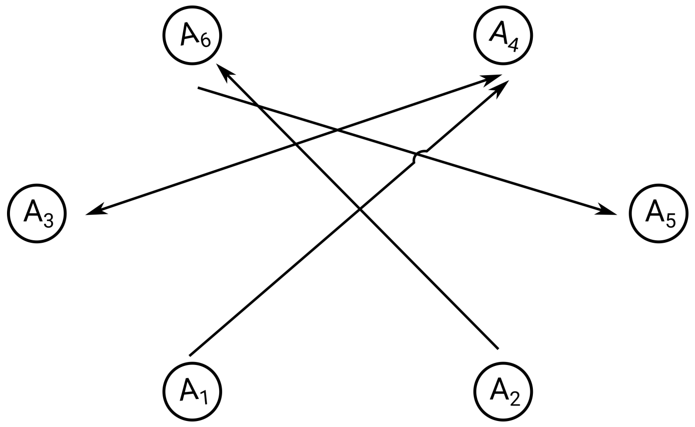


Figure 3.10: Distributed Systems: Same level of authority, different tasks for each part, and communication between parts.

consensus-based missions (Ren, Beard, and Atkins 2007), and vehicle navigation problems (Yuan and Li 2017) in MAS. Example 3.3.3 from (Assaad 2015), demonstrates how distributed control is applied on a system that consists of 4 holonomic robots, each has an objective of reaching a destination, while avoiding collision with other robots. The robots share their positions with other neighbors, but do not share their objectives (their destination). The control law used in each robot is based on a potential gradient function that attracts the robot towards its destination, and repulse it from other neighbors.

Example 3.3.3. *Consider 4 robots, r_i, $i = 1, 2, 3, 4$, with state vectors $q_i \in \mathbb{R}^2$ (equation 3.11), which represents the position vector of robot i.*

$$q_i = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \tag{3.11}$$

The potential gradient function φ_i in each robot i is depicted in equation 3.12

$$\varphi_i = \varphi_{a_i} + \sum_l \varphi_{r_{i,l}}, \qquad l \neq i, \qquad l = 0, 4 \tag{3.12}$$

where φ_{a_i} and $\varphi_{r_{i,l}}$ are the potential attractive and repulsive potential gradients, given in equations 3.13 and 3.14 respectively.

$$\varphi_{a_i} = ||q_i - q_{di}||^2 \tag{3.13}$$

$$\varphi_{r_{i,l}} = \begin{cases} \frac{S_l^2 - \gamma_{i,l}}{S_l^2 + \gamma_{i,l}} & \textit{if} \quad S_l^2 \geq \gamma_{i,l} \\ 0 & \textit{otherwise} \end{cases} \tag{3.14}$$

where $\gamma_{i,l}$ is the distance between robots i and l (equation 3.15), and S_l is the safety area of robot l, which is constant.

$$\gamma_{i,l} = ||q_i - q_l||^2 \tag{3.15}$$

The control input for each robot i is the vector $u_i \in \mathbb{R}^2$ (equation 3.16), that represents a velocity input to the robot.

$$u_i = \begin{bmatrix} v_{x_i} \\ v_{y_i} \end{bmatrix} \tag{3.16}$$

Since φ_i is a function that is null at the destination of robot i, therefore, the first-order gradient of this function represents a velocity vector pointing in the destination's direction. And in consequence, the control law that governs each robot's movement is

$$u_i = -k_{ai}\frac{\partial \varphi_{a_i}}{\partial q_i} - k_{ri}\frac{\partial \varphi_{r_{i,l}}}{\partial q_i} \tag{3.17}$$

Assuming that the initial positions of robots are

$$q_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad q_2 = \begin{bmatrix} -1 \\ 2 \end{bmatrix} \quad q_3 = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad q_4 = \begin{bmatrix} 1 \\ -3 \end{bmatrix} \tag{3.18}$$

and their destinations are

$$q_{d1} = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \quad q_{d2} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \quad q_{d3} = \begin{bmatrix} 0 \\ -1 \end{bmatrix} \quad q_{d4} = \begin{bmatrix} 0 \\ 2 \end{bmatrix} \tag{3.19}$$

For a safe area of $S_i = 0.75$, time-step of $T = 0.1s$, gains $k_a = 0.2$ for attractive term and $k_r = 1$ for the repulsive term, simulating the system of Simulink gives us the behavior shown in Fig. 3.11.

3.3.2.3 Decentralized Paradigm

Just as with any other non-centralized paradigm, the global task of the overall system is decomposed to smaller tasks, which are then distributed to the different parts. The parts of decentralized paradigms are all on the same level of authority, where each is responsible for a specific task (or set of tasks), and makes decisions using partial knowledge of the overall system. Unlike distributed paradigm where parts communicate, in this paradigm parts do not communicate, and therefore the tasks are decomposed in a way that each task may be performed without using information from another part [(Siljak 2011), (Kariwala 2007)]. This paradigm is useful when the parts in the system are distributed in a way that does not favor communication, or when communication is impossible to achieve. For example, a group of airline booking agencies that sell tickets for the same flight without communicating and coordinating their sales (Cai and Lim 2011). In Fig. 3.12, the overall process P is controlled via 3 controllers C_i, each has its input u_i and outputs y_i, $i = 1, 2, 3$. Applications that use decentralized control involve a formation of robots in which robot do not communicate while they try keep a formation and avoid collision [(Ray et al. 2009), (Dimarogonas, Zavlanos, et al. 2003)]. One decentralized method proposed (Mohammad Jamshidi 1996), called stabilization via multilevel control, consists of separating inputs in such a way that each local input contains 2 terms: a local term that has the responsibility of stabilizing that specific controller, while the second term contributes to the stabilization of the overall system.

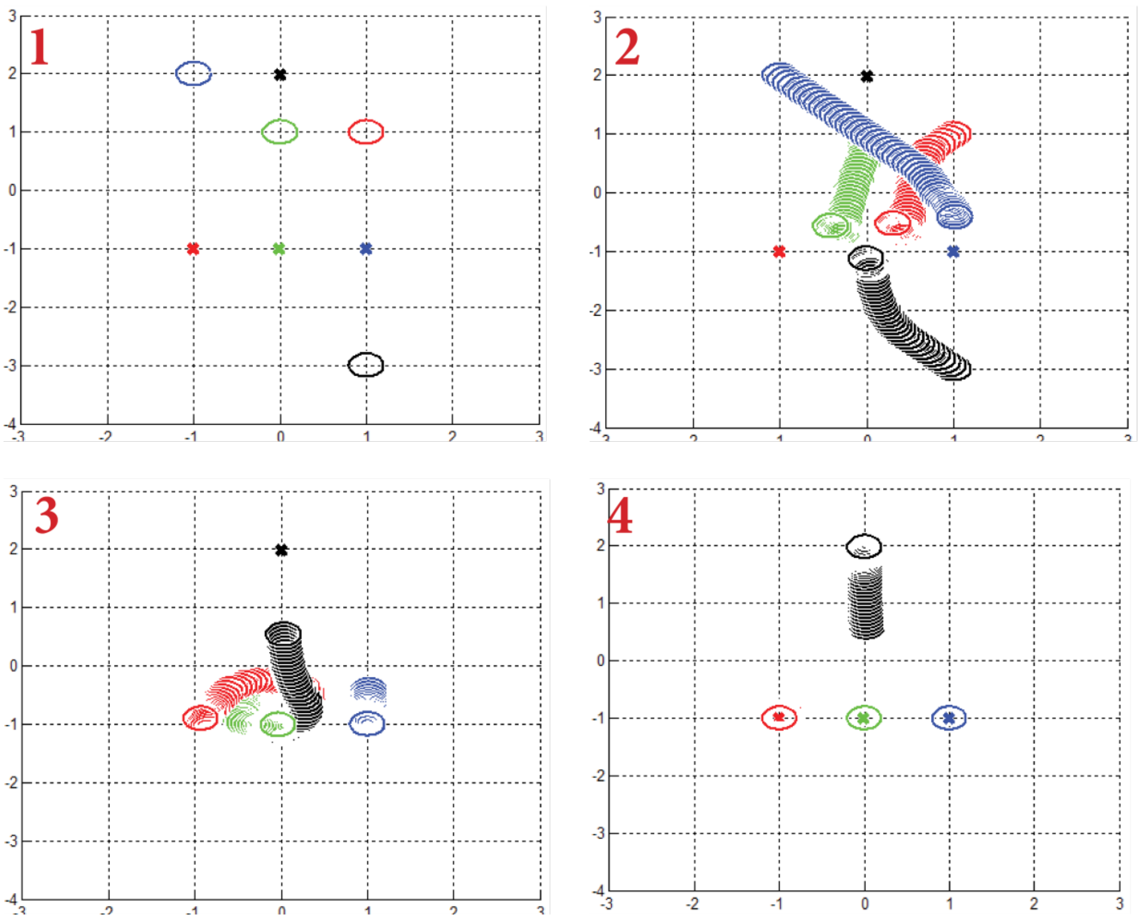


Figure 3.11: The paths followed by each robot: initial positions are depicted with circles in the first figure, and the destinations with an x mark.

Example 3.3.4. *Consider a large-scale system with N local control stations with the following model, in its input-decentralized form:*

$$\dot{x}_i(t) = A_i x_i(t) + b_i u_i(t) + \sum_{\substack{j=1 \\ j\neq i}}^{N} G_{ij} x_j(t) \tag{3.20}$$

Where A_i represents the model of subsystem i, b_i is the relation between the input u_i and subsystem i, and G_{ij} is the interaction matrix of subsystems i and j. We assume that all (A_i, b_i) pairs are controllable for all i.
The objective of decentralizing this system is to find N independent controllers for each control station, where each controller is responsible for finding a local solution based upon its local inputs, and such that the overall stabilization is achieved for the whole system. Therefore, the decentralized multilevel control input for each subsystem i is

$$u_i(t) = u_i^l(t) + u_i^g(t) \tag{3.21}$$

where $u_i^l(t)$ is the input responsible for stabilizing subsystem i, and $u_i^g(t)$ is the input that contributes to the overall stability of the system. Since (A_i, b_i) are controllable, the ith local control law may be written as

$$u_i^l(t) = -K_i^T x_i(t) \tag{3.22}$$

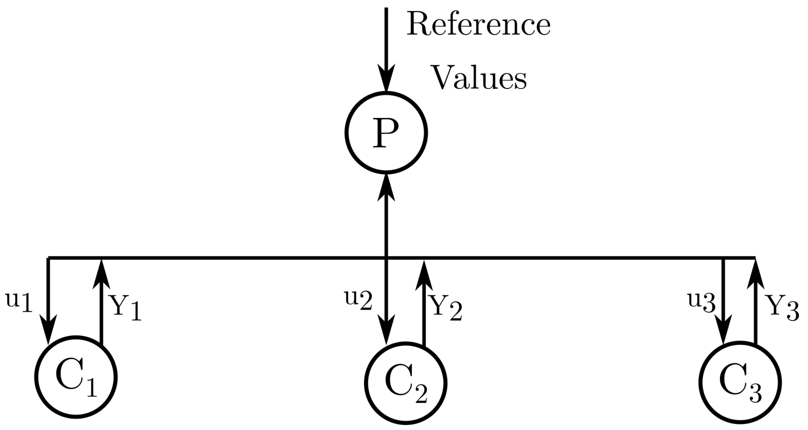


Figure 3.12: Decentralized Paradigm: multiple controllers that do not communicate perform individual tasks, the overall result is the aggregation of each controller's result.

Suppose that the global term $u_i^g(t)$ could be written as

$$u_i^g(t) = -\sum_{\substack{j=1 \\ j\neq i}}^{N} k_{ij}^T x_j(t) \tag{3.23}$$

If the subsystems are decoupled, therefore no global terms exist in the control inputs $u_i(t)$, and since we assume that the pairs (A_i, b_i) are controllable, then the local stability of each subsystem is achieved using the local control law $u_i^l(t)$. As for the global stability, we have to make sure that the aggregation of the global inputs for each subsystem will lead to the overall stability of the system.
Replacing the values of $u_i(t)$ into the input-decentralized form of the global system (equation 3.20), we get the following model:

$$\dot{x}_i(t) = \big(A_i - b_i K_i^T\big)x_i(t) + \sum_{\substack{j=1 \\ j\neq i}}^{N} \Big(G_{ij} - b_i k_{ij}^T\Big)x_j(t), \qquad i = 1, \cdots, N \tag{3.24}$$

Consider the overall system's model to be given by

$$\dot{\hat{x}}(t) = \Lambda_i \hat{x}(t) \tag{3.25}$$

The new system's equation is given by transforming each subsystem's equations using M_i matrices such that

$$x_i(t) = M_i \hat{x}_i(t) \tag{3.26}$$

$$\Lambda_i = M_i^{-1} \hat{A}_i M_i \tag{3.27}$$

where

$$\hat{A}_i = (A_i - b_i K_i^T) \tag{3.28}$$

After obtaining the transformation matrices M_i, and transforming matrices G_{ij}, and b_i, we can evaluate the interconnection gains k_{ij}^0 using

$$\hat{k}_{ij}^0 = \Big[\big(\hat{b}_i^T \hat{b}_i\big)^{-1} \hat{b}_i^T \hat{G}_{ij}\Big]^T \tag{3.29}$$

and finally the new transformed system, in addition to the aggregation of the transformed interactions gives the new transformed overall system's equation by

$$\dot{\hat{x}}_i(t) = \Lambda_i \hat{x}(t) + \sum_{\substack{j=1 \\ j \neq i}}^{N} \left(\hat{G}_{ij} - \hat{b}_i \hat{k}_{ij}^T \right) \hat{x}_j(t) \tag{3.30}$$

with $\hat{G}_{ij} = M_i^{-1} G_{ij} M_j$, $\hat{b}_i = M_i^{-1} b_i$, *and* $\hat{k}_{ij}^T = k_{ij}^T M_j$.

3.3.2.4 Hybrid Paradigms

A system is controlled with a hybrid paradigm is a system that uses a combination of 2 or more paradigms. In real life systems, often we witness a combination of hierarchical and decentralized/distributed paradigm in work [(Xin et al. 2016), (Lu et al. 2014)]. This is the case with a fleet of drones that adopts the leader-follower strategy (Hou and Fantoni 2015), where hierarchical paradigm is mixed with distributed paradigm, and a power plant in which hierarchical paradigm is used with decentralized paradigm. Consider the case of the fleet of drones, where the objective is moving in a triangular formation as shown in Fig. 3.13. In this system, the leader (black drone in the middle-front) gathers the positions and speeds of the followers drones, it proceeds then to computing the next inputs for each follower drone in a way that all the drones preserve the formation while moving. This adopted paradigm is a combination of hierarchical and decentralized paradigms. Hierarchical paradigm exists here since the leader has more knowledge about the system than other drones, and since it sends them objectives that they must achieve to preserve the formation, while decentralized paradigm exists on the lower level between followers, where each has the task of achieving the objectives sent by the leader, without communicating with any other follower drone.

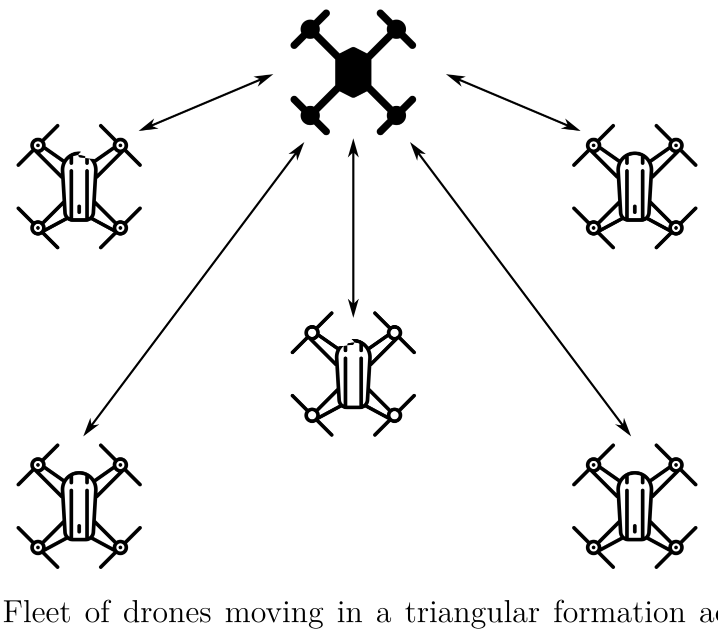


Figure 3.13: Fleet of drones moving in a triangular formation adopting leader-follower strategy.

3.3.2.5 Discussion - Non-Centralized Paradigms

Although non-centralized paradigms are more robust than centralized paradigms because they do not suffer from the single point failure problem, however, analyzing and predicting the behavior of the system in systems that use non-centralized paradigms is much more challenging. In centralized paradigm, the system is controlled by a single authority that has the knowledge of the state of the whole system and controls all of it, whereas here the tasks are distributed in a way that the global behavior is controlled by controlling the separated tasks in each part of the system. For example, how to make sure that vehicles in a fleet will move according to a predefined formation when each vehicle is partially blind to the state overall system? In such systems, simulation and lessons learned help a lot in defining the right distribution of tasks, and the information shared between parts if any. As for the choice of the best paradigm that suits a specific system, multiple factors affect this choice: the nature of the system, the physical distribution of parts, the tasks to be performed in the system, etc. are all factors that must be taken into account when choosing the paradigm. Furthermore, the requirements that system owners or users require from the system also affect this choice. Let us go back to the drones example from the Hybrid Paradigms Section 3.3.2.4 where a mix of hierarchical and decentralized paradigms was used to control the system. If the owners of this system require that they do not want to use the leader-follower strategy, another way to achieve the objectives of this system (moving in a triangular formation) is by using a distributed paradigm, where drones share speeds and positions with neighbors, and choose the next objective by their own (Fig. 3.14). Here we have the same system, with the same objectives, but with different requirements, which leads to adopting a different control paradigm.

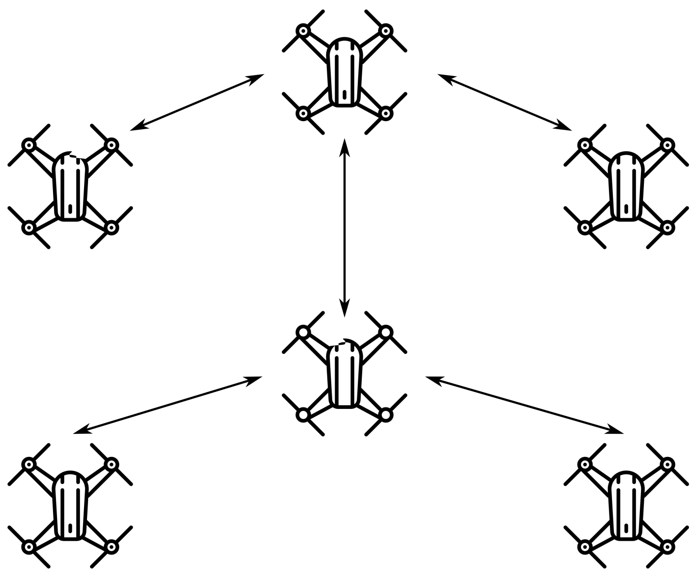


Figure 3.14: Fleet of drones moving in a triangular formation adopting distributed paradigm.

3.4 SoS Control

So far we have discussed control for single systems, whether they are small or large in scale, but what about SoS control? How are we going to use the previous information to control CS in SoS in order to achieve the desired performance from the overall SoS?

Our objective is to find a control law that could be used to control CS in SoS to achieve a certain objective. If we use the traditional SE approach, we must formulate a model that describes the relationship between different states in the SoS, and how they evolve over time, and another model that describes the architecture of the SoS and how its CS are integrated. These two models will than be compared to models of single systems, and hopefully we get a control law that could be used as a starting point to control the SoS. A generic architectural model for SoS is given in Fig. 3.15, where CS_x depicts the xth CS, the solid arrows indicate uni- or bi-directional communication between CS. The node SoS_A indicates the presence of an authority for this SoS that communicates with all CS. This node along-side its communications do not always exist in SoS, that is why they are represented using dashed lines. Furthermore, by using the different taxonomies of SoS given in the previous chapter

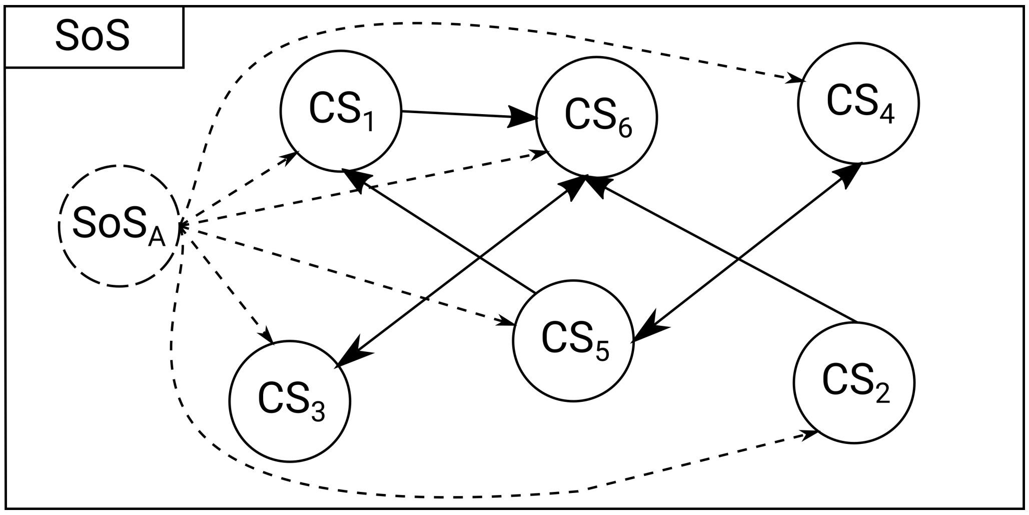


Figure 3.15: Simplified SoS Architectural Model: Generic Form.

(Section 2.5), we could refine this architectural model by providing different models for each different type of SoS. Starting from virtual SoS, we know that there exists no SoS authority, and that CS do not recognize the presence of other CS in the SoS, and consequently we assume that no communication occurs between them for the SoS. Using this information, we could eliminate SoS_A and all communication from the previous model to get Fig. 3.16. As for collaborative SoS, CS are aware of the presence of other CS in the SoS and therefore they communicate information, on the other hand, there exist no SoS authority in this type of SoS. Applying this to our previous generic model, we get the model in Fig. 3.17. Regarding acknowledged and directed SoS, this model becomes similar, where CS communicate, and SoS_A exists (Fig. 3.18). The reason why both acknowledged and directed SoS are represented in the same way is that this representation that we used to represent the architectural model of SoS does not show the difference between the different types of communication that might occur in the system, or the different relations that exist between CS, and between CS and SoS authority. In the current context, namely operational context, the differences between directed and acknowledged SoS are irrelevant, as they mainly occur during acquisition or development stages of either SoS or its CS. Let us analyze the two levels of control that we have mentioned earlier: low and high

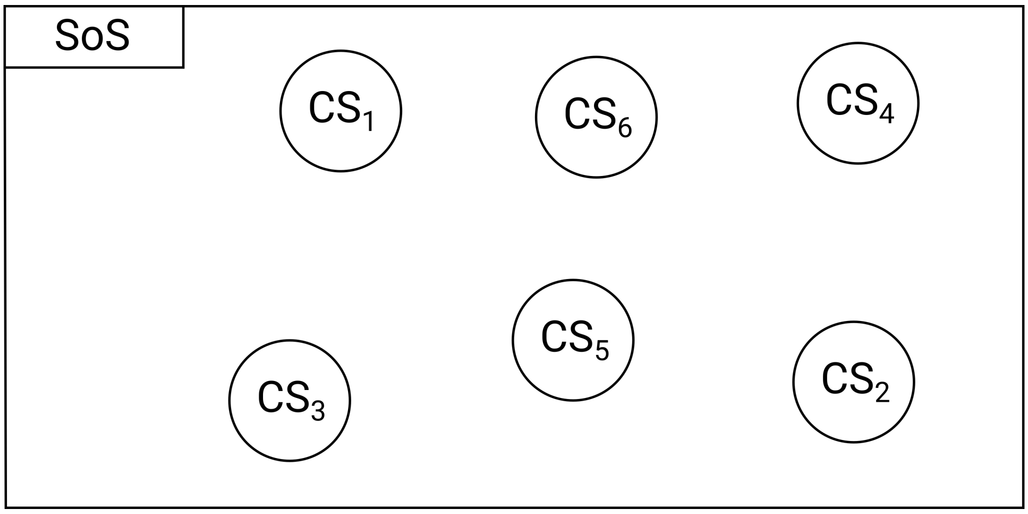


Figure 3.16: Simplified SoS Architectural Model: Virtual SoS.

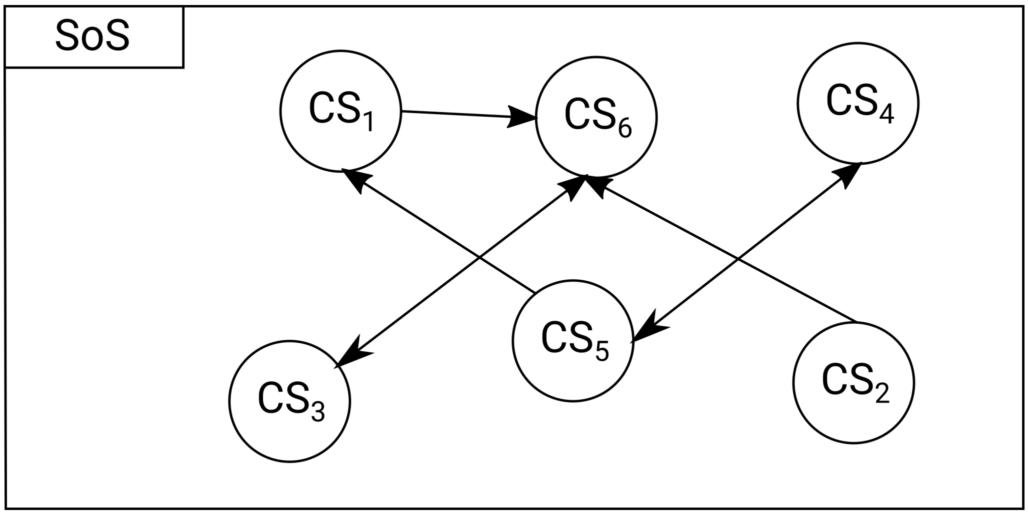


Figure 3.17: Simplified SoS Architectural Model: Collaborative SoS.

control levels. We have said that low-level control means controlling the states of the system so that it operates as expected, and high-level control means controlling the objectives of a system, or setting the reference values on its states. Each level of control is treated separately in SoS and in different ways. low-level control requires SoS authority to have complete knowledge of the different states in its CS, and how do they affect each other. Furthermore, this level of control requires access to the CS's actuators, and the internal real time values of its states. In SoS, these types of information are rarely accessible to SoS authority, furthermore, low-level control means controlling how the system does what it has to do, something outside the scope of SoS authority since CS are operationally and managerially independent.
Let us remember some specific characteristics of CS in SoS: CS are operationally independent and they belong to the SoS because this belonging offers them some kind of advantages. This means that CS know how to do what they want/have to do, and that they might leave the SoS in case they do not consider their belonging beneficial. Moreover, CS exist in SoS (or they are added to it) for a capability (or set of capabilities) that they offer to the SoS. There are several possible ways a

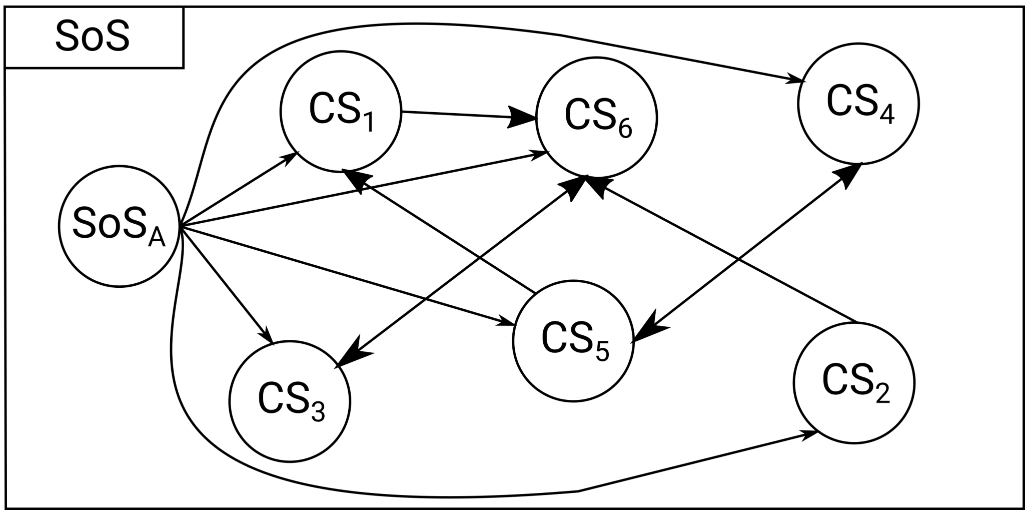


Figure 3.18: Simplified SoS Architectural Model: Acknowledged and Directed SoS.

CS could join a SoS, and they depend on the type of SoS in question. In directed and acknowledged SoS, CS are contracted either for the whole life-time of the SoS, or for specific missions that they carry on for the SoS. In this case, the questions of belonging and incentives are answered when the CS is contracted, and it must perform what it is asked by the SoS. In collaborative SoS on the other side, CS belong to the SoS as long as the reasons that they join it for are valid, and serve this particular CS. Furthermore, CS are not required to perform actions for the SoS even when they belong to it, so in this case, incentive must be thought of even after CS join the SoS.
In a SoS, control is the question of providing incentives to and requesting services from CS. Consider a crisis prevention and intervention SoS in a city (Fig. 3.19). Assuming that CS of such system are the police department, fire department, a surveillance center, the mayor and the city council, and lastly the citizens of this city. The objective of this SoS is to "predict and handle any crisis that might arise in the city before it happens, and if not possible, intervene and solve the crisis in the least amount of time possible". This SoS operates as follows: the surveillance center, police department and fire department are always evaluating the situation in the city to predict any possible crisis that might arise. In case they decide that a crisis is coming and they should act on it, the mayor and city council give the necessary permission and if needed, to provide logistical support. Citizens have 2 roles in this SoS: in case they detect any unfamiliarity they could report it to authorities, and in the case of a crisis, they might help authorities in some situations. Let us zoom on the citizens' first role in this SoS: report unfamiliar events for authority. Providing that the incentive of the well-being of the city is enough for citizens to report such events, and an event occurs, the question to be asked is: to whom citizens should report this event? And is there a format that must be respected in order to report this event?
The job of SoS authority is to inform its CS about what they should do, and to think about the communication methods to be used between CS in order to achieve the desired operations from the SoS. In the case of the previous example, SoS authority informs citizens that they should call a particular phone number to report an event, and informs the surveillance center that in case they detect something

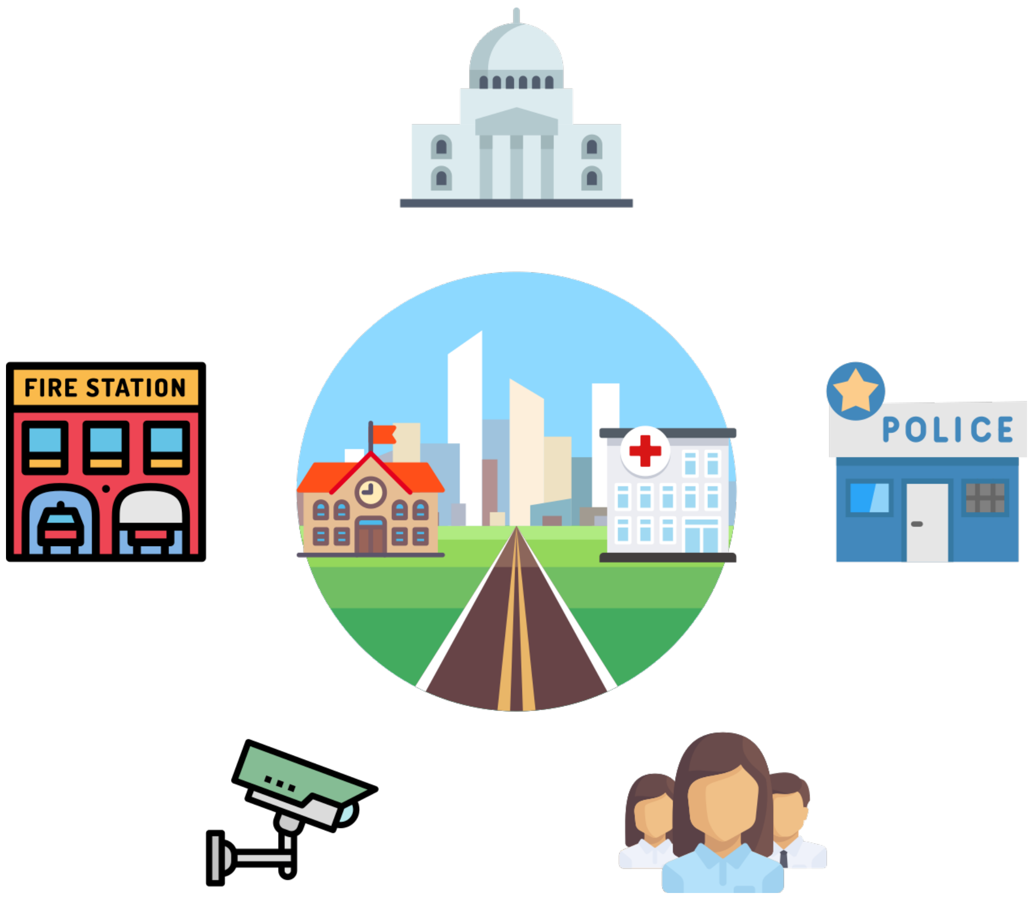


Figure 3.19: Crisis prevention and intervention SoS.

crisis related, they should contact the police department to investigate closely, and that all departments for example must be in direct contact with the city council when they operate in the SoS. The problem of control in a SoS is to manage the relationship between CS, and between CS and SoS, and to inform CS of what they should/must do, and when to do it when it comes to SoS.
Next chapter will discuss architectural frameworks, which are a tool for managing the relationships between different entities in a system, and which describes the behavior of said entities, as long as the overall system behavior.

3.4.0.1 Discussion - SoS control

Unlike control in traditional systems, control in SoS faces different challenges, and is still identified as part of the research gap in SoSE (DeLaurentis and Callaway 2004). Managing emergent behaviors when thinking about the interactions between CS is a challenging task, so is the task of verifying that the system would still be operational even if some CS do not perform what the SoS requested from them. Another important point that differentiates SoS control from traditional systems control is the fact that CS are operationally and managerially independent. The operational independence of CS means that SoS authority could only apply high-level control on them, and their managerial independence means that SoS authority may only influence CS management and is unable to apply rigid control over CS (Fang, Davendralingam, and DeLaurentis 2018), if we consider that control in systems range "from rigid control by one system over another to simply influence of one system over

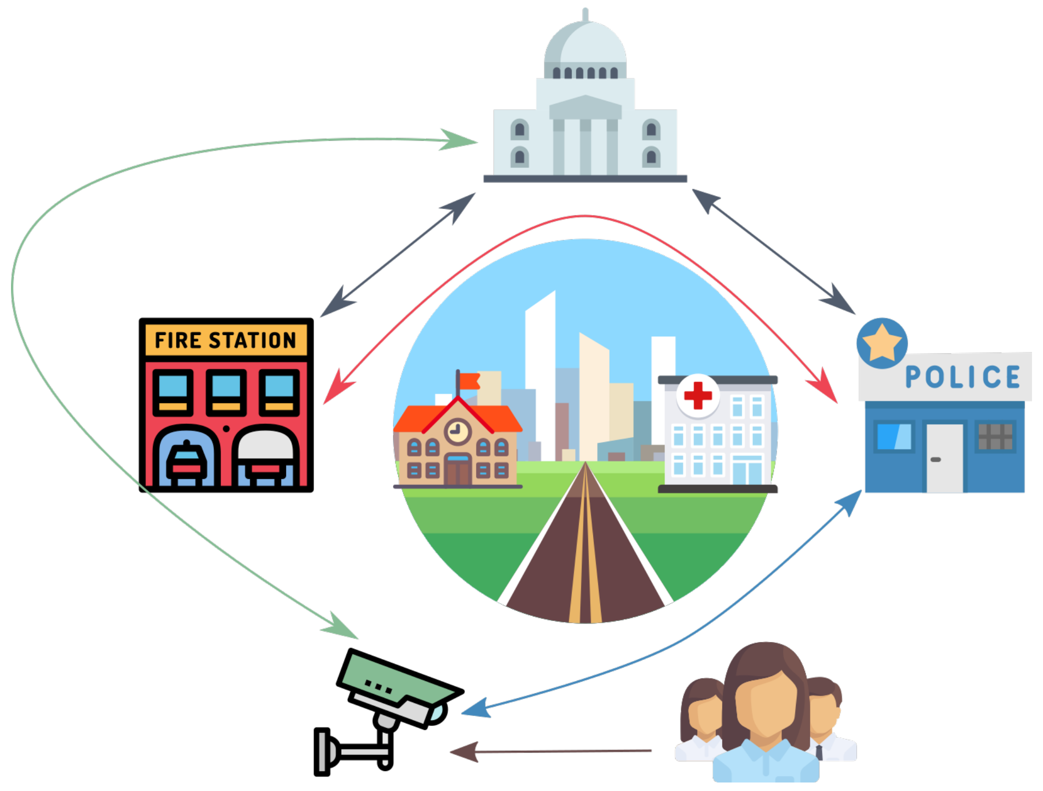


Figure 3.20: Crisis prevention and intervention SoS: each CS knows their role and how they relate to other CS in the SoS.

another" (Henshaw et al. 2013). In contrast, in traditional systems, parts could be operationally independent or not. However, they are not managerially independent, and that is the way we assume that parts will perform exactly what is requested from them, since they all share the same management.

3.5 Conclusion

Systems control is the discipline of systems engineering that handle the problem of controlling the parts of a system in order for it to respect the predefined behavior defined by its authority. Depending on the nature and requirements of systems, control may be applied in systems using a centralized or a non-centralized paradigms. In centralized paradigms a single controller is responsible for knowing the states of the overall system and for sending inputs to the different actuators in the system, whereas in non-centralized paradigms, whether be it hierarchical, distributed or decentralized, the task of control is decentralized and no single controller is able to control the whole system. Hierarchical paradigm contains different levels of parts, distributed based upon the authority each part has, no horizontal communication is allowed. Distributed paradigm lacks the vertical distribution of tasks, and it uses inter-part communication to obtain better performance from the system. Decentralized paradigm is similar to distributed paradigm in that it has no vertical levels of authority, and similar to hierarchical paradigm in that no communication occurs between parts, but rather each part is responsible over a local role that it must perform.
There are similarities between SoS and traditional systems on the organizational level, directed and acknowledged SoS often have certain hierarchy between CS, and

on the same level we might find communication between CS. Collaborative SoS are so much similar in organization to a graph of nodes, where nodes share information and each node is responsible for making its own decisions. Virtual SoS, on the other hand, are similar to a decentralized paradigm, where CS are not aware of the presence of other CS that exists in the same system. These similarities inspired us to study the control problem in SoS to conclude that in SoS, control is the process of defining the different relations between CS, and what actions should CS do in order to achieve a desired behavior on the SoS level. Finally, we have mentioned that frameworks (the focus of the next chapter 4) might be the solution for controlling CS in SoS.

Chapter 4

Modeling & Frameworks

4.1 Introduction

To describe an object, whether it exists or still an idea, we might use drawings to describe its form, equations, and charts to describe some behavioral aspect, and any other means that help us convey the correct idea. Combined together, these means of communication are called models. In systems engineering, models are used to describe the behavior of the system, its form, the data it retains, etc. For any particular aspect of a system, a model could be developed to describe this aspect. Models vary in types, mathematical models use equations to describe the modeled system, while conceptual models use concepts, shapes, and arrows to describe the modeled system. The terms modeling and simulation often come together in the literature [(Cantot and Luzeaux 2013), (Rainey and Tolk 2015), (Ouyang 2014)], and the reason is the ability to integrate mechanisms into models that dictate how a model evolves over time, and in result we could simulate the behavior of a system for whatever purpose. The life cycle of any system consists of a series of cycles which involve each a model or set of models, from the initial conception model that describes the initial idea of the system, to the different models that describe its behavior and architecture. Models are a powerful tool to convey ideas in a clear and non-ambiguous manner, if created right that is!

SoS are systems, and thus they are also represented using models that describe their different aspects. As we have seen earlier, SoS are different from other types of systems on different levels, and the same goes for modeling as well. Each SoS requires the cooperation of multiple interdisciplinary teams either of its constituents or the entity building and maintaining this SoS. To account for this, several models must be created to describe different aspects of the SoS, aspects like its architecture, behavior, and objectives must have models that describe them in a clear way in order to have a shared knowledge about the SoS in all interested parties. Frameworks are a good tool to achieve this objective, in particular, multi-view frameworks consists of views that describe whatever aspect we want in a system. As we will see later on, in addition to modeling, when using frameworks we can simulate certain aspects of a system, and most importantly we can perform traceability on different elements of the different views.

This chapter continues as follows: Section 4.2 describes models, their different types, and presents some best practices adopted in modeling systems, and the relationship between models that describe systems and simulations that use these models to simulate the behavior of a system. Afterwards, in Section 4.3 we describe how modeling SoS differs from modeling traditional systems to account for our specific needs, which leads us to multi-view frameworks, where we provide some examples used in SoS, and of course, different modeling languages used to build models. Finally, we end with a conclusion in Section 4.4.

4.2 What is a model?

A model is a human construct that represents our understanding of a concept, relationship, structure, system, or any aspect of the real world. In general, models are abstractions, which allow model makers to eliminate unnecessary details for the purpose of clarity and simplicity, while at the same time preserving a high degree of validity regarding the modeled aspect. The objectives of models are numerous, and include:

1. To facilitate the understanding of ideas and objects;
2. To aid in decision-making by simulating "what it" scenarios;
3. To aid in communicating complicated and complex ideas about something of interest;
4. To control systems, estimate unmeasured states, and predict their behavior.

4.2.1 Types of Models

The aspects described within a system are diverse in type, such as describing the architecture of a system, or its behavior, and therefore different modeling techniques are required to model different aspects of a system. Since different aspects require different representations, models come in 3 types: descriptive models, analytical models, and hybrid models. Descriptive models use drawings, charts and other graphical representations to describe how are things in a system, while analytical models use mathematical equations in order to describe mathematical relations in the system, whereas hybrid models contain both descriptive and analytical tools.

4.2.1.1 Descriptive Models

A descriptive model is a "model that describe things as they are"(Turban, Sharda, and Delen 2010). This type of model is used to describe aspects of systems such as their architectures, objectives, requirements, etc. Examples such as cubesat reference model (Kaslow et al. 2015) which describes a recommended architecture for building cubesats, and the architecture for industry 4.0-based manufacturing systems (J. Lee, Bagheri, and Kao 2015) which intends to guide the development of cyber-physical systems in support for industry 4.0 manufacturing systems. Charts, diagrams, and other graphical representations may be used in this type of model to convey the ideas. The purpose of descriptive models is to answer questions such as "what does it look like?" or "what does it do?", which may be either a drawing for the first question, or a text for the second one. Fig. 4.1 represents the model that describes the structure of stakeholders in the CubeSat reference model. Shapes and connectors are arranged in a hierarchical representation that shows the different stakeholders, and why each one exists.

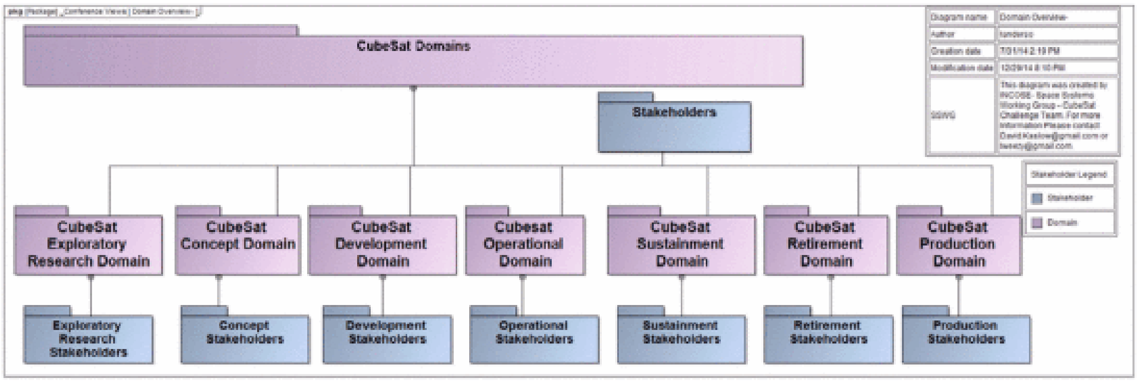


Figure 4.1: A descriptive model that describes the structure of stakeholders in CubeSat domain (Kaslow et al. 2015).

4.2.1.2 Analytical Models

While descriptive models describe objects, analytical models give us the ability to analyze different aspects of a system by using mathematical equations. Analytical models may be dynamic if they describe a time-varying state of a system, e.g. a model developed to describe the behavior of a vehicle over time (Jazar 2017), or static, in which case the model represents a time-invariant aspect of a system, e.g. a model that describes the capacity of railway lines (Weik, Niebel, and Nießen 2016) in order to perform analysis on the railway network. This type of model provides insights about performance, cost, and any other quantifiable parameter of the system, and as we will see later, analytical models are the corner stone of simulations in systems engineering. For example, the equation system 4.1 (Ren 2008) is a model that describes the dynamics of 2 objects of mass m, connected with a damper with coefficient b, and attached each to a fixed support by identical springs with spring constant k. This model describes the relation between position, speed, and acceleration of both masses. If we solve it for the time variable t, we could predict the different behaviors of the system with a certain degree of accuracy.

$$\begin{aligned} m\ddot{x_1} + kx_1 + b(\dot{x_1} - \dot{x_2}) &= 0 \\ m\ddot{x_2} + kx_2 + b(\dot{x_2} - \dot{x_1}) &= 0 \end{aligned} \tag{4.1}$$

4.2.1.3 Hybrid Models

In hybrid models, descriptive and analytical models are coupled together to make logical analysis on a certain aspect of a system. Usually such models involve higher levels of abstractions than the previous types, and they provide a higher level of description of the modeled system. For example, consider a system that consists of agents that choose to stay in the system or leave it based upon the evaluation of a certain cost function. To model the structure of that system at any given moment, we are going to need a descriptive model to describe the system's structure, as well as an integrated analytical model which describes the state of each agent at any moment, and whether they will stay in the system or leave it based upon the evaluation of the cost function. In contrast, if we are interested in modeling the behavior of agents at a given moment, without looking at the structure of the overall system, an analytical model is enough to do so.

4.2.2 Modeling and Simulation

During the process of designing and building a system, decisions about the system's structure, its behavior, the technologies to be used, etc. are in the majority made when the system is still a concept. Some of these decisions are made based upon previous experiences and lessons learned, other decisions are based upon standards and best practices. However, there are decisions that must be made in each system that need to consider that system's own environment, most precisely, how the system must act towards its environment, and how it must react to it. Such decisions are made after examining the system's structure, understanding the behavior of its parts, and predicting the system's behavior based upon the relationship between its parts and their behaviors.

Example 4.2.1. *Consider a manufacturing line, which consists of several machines (Fig. 4.2), and the decision about how many machines we must install in order to achieve a certain quantity of the product in a certain period. First, we have to identify the parts involved in this process, which are the machines and the belts connecting them. Then we have to know the behavior of these parts, so for machines, we are interested in knowing how many products they can produce in a period of time, and for transportation belts we have to know their capacity based upon their dimensions. After identifying the parts involved in our process, and understanding their behaviors, we then have to identify the relationship between them, and how each part contributes to the global outcome. Assuming that the machines are arranged in parallel, so that the overall outcome is the sum of the individual outcomes of those machines, then we could say that the relationship between the parts is linear, and therefore, we could choose the number of machines, as well as belt so that the total outcome is equal to the desired outcome.*

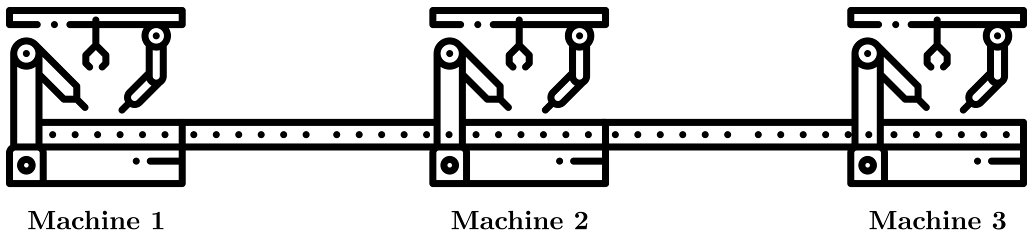


Figure 4.2: A production line that consists of 3 machines, related to linear relationships.

Example 4.2.1 shows a simple case where the relationship between the parts is linear, and the overall behavior may be predicted by performing a simple thought experiment that imitates the real system once it is deployed, to make a design decision about the system. This process of imitating the system is called simulation (Banks et al. 2004), and it is a very powerful tool in system design and verification. Simulation helps us predict the behavior of a system under different circumstances to a certain degree of accuracy, without the high cost affiliated with performing the test in real-life. For example, if we want to understand how a certain bridge reacts to a certain load, instead of applying that load to the bridge and risking its destruction, we could develop a detailed model that models the structure of the bridge, and create a computer simulation that imitates the bridge, and observe its reaction to that load, and any other load. Since simulation uses system model, its results depend heavily on the validity of that model. Therefore, before simulating a system and using the results to make decisions, we have to make sure that the model used in that simulation is valid and reflects the true nature of the system. When we wish to simulate a system that already exists, we can make sure that our model is valid by using it to simulate the system under known circumstances, and comparing the results with the observed state of the system: if the results and the real behavior match, then we can proceed to simulate the system under unknown circumstances, and use the results. However, when the system does not exist, this comparison cannot be made, and we should be very careful about the validity of our model that we used to simulate the system.

4.2.3 Model-Based Systems Engineering (MBSE)

Systems engineering processes include a variety of documents and models that describe the system of interest, from design to analysis and verification. Every document could be considered as a model that describes a certain aspect of the system in question, and therefore technically speaking, systems engineering discipline was and still a model-based approach. However, the tools used to build these models are textual documents, informal drawings, and spreadsheets in general. When numerous teams are involved in developing a specific aspect of a system, communication is a key factor for achieving reliable results. When using the traditional document-based approach, the tools mentioned earlier suffer from inconsistencies, and are difficult to maintain and be reused again. For example, when providing requirements using text, the specific meaning of words matters. Consider the following text which describes a hypothetical requirement of a system:

> The system's response to external requests must be **quick** in order to meet stakeholders' expectations.

The use of the word **quick** is ambiguous as it does not provide any time frame, and it represents a subjective opinion of the person who used it. This is an example of one of the various problems that we face when using documents as references in systems development. The objective of MBSE is to transition from document-based systems engineering, where the emphasis is on developing documentation of the system, to a process that emphasizes the use of models from the initial conceptual design, throughout all later life cycle phases (INCOSE 2015). In other words, MBSE tells us that models are no longer simply a good idea, but rather an integral part of the whole engineering process of engineering a system. In the world of traditional system engineering, there are an increasing number of approaches that address this topic [(Dickerson and Mavris 2016),(Estefan et al. 2007)], however, in the world of SoS engineering, there is a scarcity of such approaches on the SoS level (Lewis et al. 2009). In order to realize the full potential benefits of MBSE, there are 3 areas that must be addressed (Holt, S. A. Perry, and Brownsword 2011):

1. People: There must be properly educated, trained, and experienced people available who hold the appropriate competence for their roles;

2. Process: In order to realize MBSE capability, there must be an effective set of processes in place, which is properly deployed and available to all people;

3. Tools: Tools are required, particularly for automation, such as computer-aided

system engineering tools, notations tools, architectural frameworks, and so on.

4.3 SoS Modeling

The process of designing and developing SoS involves multiple teams that elicit requirements from the different stakeholders, and provide design solutions to achieve the SoS goals. During this process, numerous models are exchanged between all members working on the development of that SoS (Belkadi, Bonjour, and Dulmet 2004), as well as any interested stakeholder, where each model describes a specific aspect of the SoS in question. MBSE is being applied nowadays to the SoS development process since it provides a more efficient and viable solution: models that use specific and unambiguous elements to describe a certain aspect of the SoS, and that can be extended to allow for the integration of analysis in the model. For example, if we agree that a triangle represents a drone, a rectangle represents a car, and a line that connects 2 shapes represents a communication link, we can easily understand the relationships between agents in a SoS that consists of 3 drones and 2 cars if we look at Fig. 4.3. Furthermore, any person who is interested in this aspect of that SoS can understand these relationships simply by understanding the meanings of the semantics used in the model. The development of SoS is evolution-

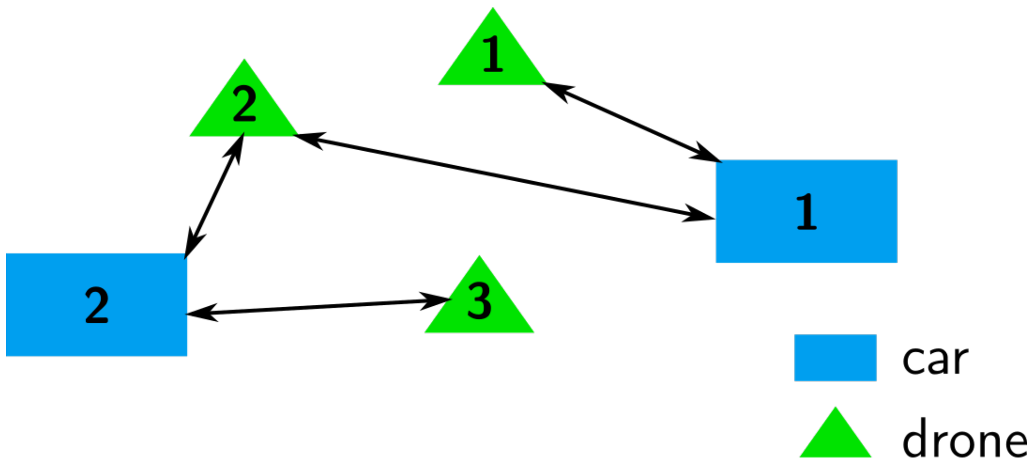


Figure 4.3: A model that describes the relationships between different agents in a SoS.

ary over time, because SoS are characterized by increasing complexity, emergent behavior, and uncertainty in requirements and context (e.g. evolving technologies., changing environment) (Andary and Sage 2010). Furthermore, for each aspect of the SoS, a model may be developed that addresses that specific aspect. A structural model may describe the structure of the SoS, whereas a behavioral model may be used to describe its behavior. Therefore, any tool that may be used to model SoS must have the ability to integrate the rationale behind every element that exists in every model, so that any change that happens in that SoS, whether it is an internal change (e.g. CS joining or leaving), or an external change (e.g. environment or stakeholders), may be taken into consideration in the overall model of that SoS. The complexity of SoS is somewhat similar to that of enterprises or large information systems, where the different models that describe a system in those categories are usually a part of a framework that contains all the necessary information needed in the system. A framework is a group of analytical and descriptive models that describe the components of a system, their objectives and behaviors, as well as their inter-relationships.

4.3.1 Architecture Frameworks

According to the ISO/IEC/IEEE 42010:2011 standard definition, an architecture framework is all "conventions, principles, and practices for the description of architectures established within a specific domain of application and/or community of stakeholders". They provide some conventions, rules, and practices for developing the architectural descriptions of SoS from multiple perspectives and on varying levels of abstraction, thereby allowing stakeholders to focus on specific aspects of interest, while keeping sight of the whole system (ISO 2011). In other words, an architecture framework is a set of models, which describes each an aspect of the SoS while abstracting the other aspects of interest in that SoS, and when looked at all together, they contribute to the understanding of the whole.
Many different architecture frameworks have been established over the last decades. One of the earliest and most well-known architecture frameworks, the Zachman Framework, published in 1987, defines an enterprise in a formal and highly structured way using a 2-dimensional classification matrix. The matrix is based on the intersection of six basic communication interrogatives (What, Where, Who, When, Why, and How) with six rows (contextual, conceptual, logical, physical, as built, and functioning). However, defense organizations were the main contributor to the development of architecture frameworks in the last two decades (Ota and Gerz 2011). The US Department of Defense (DoD) views architecture as the mechanism for the transformation of capabilities in the information age and, therefore, has issued a series of architecture frameworks, such as the Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance Architecture Framework (C4ISRAF) and DoDAF (Wagenhals and Levis 2009). C4ISRAF and DoDAF are also the baselines of other mature and formally adopted defense frameworks, such as the UK Ministry of Defense Architecture Framework (MODAF) (Partners 2005), and NATO Architecture Framework (NAF) (Handley and Smillie 2008). Moreover, civil organizations have employed these defense frameworks for designing architectures, and developed their own frameworks, like the Unified Architectural Framework (UAF)[1] of the OMG, The Open Group Architecture Framework (TOGAF) (Haren 2011), and the System of Systems Operational Management Matrix (SoSOMM) (Gorod, Gove, et al. 2007).
An architecture framework should contain a detailed description of the objectives of SoS, the components, the inter-relationship between them, and the rules that govern the SoS in question (the invariant). In addition, due to the complexity of SoS and the detailed description needed from a framework, it is hard to describe all the aspects of a SoS in a single-view framework (Cole 2008), that's why a good framework for SoS should use a multiview approach. A view (or viewpoint) in architecture frameworks context is a set of models that describe a specific aspect of a SoS. For example, one of the viewpoints in the DoDAF is the capability viewpoint, which consists of a set of models that address capabilities in the SoS of interest (Capability Taxonomy, Capability Phasing, Capability Dependencies, etc.), however, due to the complexity of SoS and the different aspects that must be addressed, architecture frameworks that describe SoS consists of multi-viewpoints. Even though the

[1]www.omg.org/uaf/index.htm

viewpoints that consist a specific framework vary depending on the modeled SoS, however, there are some major viewpoints that are usually in a SoS architecture framework such as the purpose view, which describes the objectives of that SoS, the form view, which describes the physical form of that SoS (CS, their relationships, etc.), and the behavioral view, which describes what the SoS does.
Another reason why frameworks are important in the development of SoS, and any other complex system, is their reusability (Zhang et al. 2012), in the sense that once a framework is developed for a particular SoS, it may be modified and adapted to another SoS in case both SoS have similarities between them. This enables the SoS designers to focus on the content, and saves a lot of architecture and design time, especially since frameworks are usually built using a dedicated language that is both human and machine readable, which uses defined semantics. For example, if we were to develop a framework for a SoS for an earth observation application, instead of starting from scratch, if we use the framework proposed in (Durbha et al. 2006) for instance, we start with multiple advantages:

1. We can use the knowledge presented in the mentioned framework to improve the design of our SoS;

2. We can add the capability for cooperation between our new SoS and the reference SoS, which increases the scope of the operation of both systems (in case both systems have similar domain of applications).

4.3.2 Systems Modeling Language (SysML)

Frameworks are not only a representation of a SoS, and they are not the end goal (Maier 2006). Although frameworks capture a static description of the SoS, it is essential that they are built in such a way that enables them to be used to perform dynamic analysis of how the CS interact with each other, and whether the SoS exhibits the desired behavior as modeled (Wang and C. H. Dagli 2011). To do so, frameworks must be built using specific languages that allow this kind of analysis to be performed. There exist multiple languages that support this type of operations[2], but in this section we will focus on a specific language: Systems Modeling Language (SysML) because of its widespread use in the industry, extensive tool support, and flexibility of use (Holt, S. Perry, et al. 2015). SysML is a general-purpose architecture modeling language for SE applications, it supports the specification, analysis, design, verification and validation of a broad range of systems and SoS. It was developed by the Object Management Group[3] (OMG), as an extention of a subset of the Unified Modeling Language 2 (UML2) which was built to support the development of software systems, as an effort to promote MBSE, and to move from document-centric to model-centric in systems engineering. SysML specification was developed based on the UML for SE request for proposal (RFP), which was developed jointly by the OMG and INCOSE[4]. SysML contains 9 type of diagrams, distributed into 3 categories (Fig. 4.4): structure diagrams, behavior diagrams, and requirement diagrams. Structure diagrams are used to represent all information about the structure of the system, as well as the relationships between components, whereas behavior diagrams are used to represent behaviors in the system such as activities, interactions

[2]www.compass-research.eu/Project/Deliverables/D22.6.pdf
[3]www.omg.org
[4]www.omgsysml.org

(using sequence diagrams), and life cycles (using state machine diagrams), finally, requirement diagrams are used to represent text based requirements.
There exist a lot of software that supports the use of SysML for modeling and analysis of systems, such as No Magic[5], and Visual Paradigm[6]. By using software that enables SysML support, we can focus on modeling the system, and perform more detailed analysis, while at the same time making sure that the model we are developing may be reused and shared with any entity interested in that system: another witness on the benefits of MBSE approach in engineering systems. For detailed

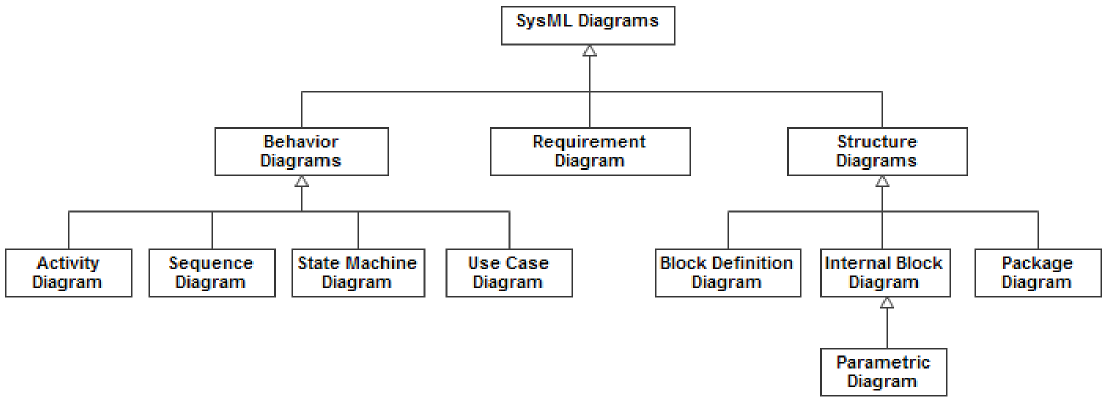


Figure 4.4: SysML diagram taxonomy [7].

information about SysML, readers may read the tutorial provided by the OMG at `www.omgsysml.org/INCOSE-OMGSysML-Tutorial-Final-090901.pdf`

4.3.3 SoS-ACRE Framework

To illustrate how frameworks are developed and used, in this section we are going to describe a framework built on the basis of MBSE, that uses requirements from the different stakeholders, SoS authority, and CS to build a SoS framework on the basis of Model-Based Requirements Engineering (MBRE) (Holt, S. A. Perry, and Brownsword 2011). This framework was adopted by the COMPASS[8] project, and has been used successfully on a number of industrial projects and on the European projects OPENCOSS[9] and iFEST[10]. The Approach to Context-based Requirements Engineering (ACRE) is a MBSE approach that describes requirements ontology that is used to generate the different needed requirements (Holt, S. Perry, et al. 2015). SoS-ACRE was built on top of the ACRE approach, in order to benefit from its best practices.

4.3.3.1 ACRE Ontology

The first element of this approach is its ontology, or the set of concepts, terminology, and the relations between the different concepts. Figure 4.5 represents the ACRE ontology using SysML block definition diagram, where each block represents

[5] www.nomagic.com
[6] www.visual-paradigm.com/features/sysml-diagram-tool
[7] www.sysml.org
[8] www.compass-research.eu
[9] www.opencoss-project.eu
[10] www.artemis-ifest.eu

a concept. For example, the concept “Need” has three types: requirement, goal, or capability. The relationship between “Need” and “source element” is noted on top of the connector that connects two blocks, and it states that one or more “need” is elicited from one or more “source element”, and the connector between “need” and “rule” tells us that rules constrain requirements. On the other side, we have two “context” types: “system context”, which represents the need for its source system, and “stakeholder context”, which represents the context of a stakeholder. A “Use case” describes the context of each need, and it is validated via one or more “scenario”, which might be “formal” or “semi-formal”.

4.3.3.2 ACRE Framework

After presenting the ontology, now we present the ACRE framework, using SySML block diagram, in Fig. 4.6. The ACRE framework consists of 7 views described briefly below:

Source Element View: which contains all the information source required to specify the system requirement. This view records all the origins of requirement sources in the system.

Requirement Description View: contains structured descriptions of each need (requirement, goal, capability) elicited from the source elements.

Definition Rule Set View: contains all the rules that may be applied to the needs in the previous view.

Requirement Context View: contextualizes requirements by looking at them from a specific point of view.

Context Definition View: identifies the different points of view that may be choosen to look at certain requirements.

Validation View: contains all validations that are needed to ensure that all needs are met in the system.

Traceability View: contains all the relations that exist between elements of different views.

4.3.3.3 SoS-ACRE Framework

The previous ontology and framework does not contain any mention of SoS, because ACRE approach was originally developed to be used in a single system. To expand this approach so that it may be applied to SoS, first the ontology had to be expanded, to what is called COMPASS ontology (Fig. 4.7), which builds upon the ACRE ontology and adds SoS to the system context. The idea behind this is that a context may represent a system from a specific point of view, and that SoS is a higher level of a point of view that looks at multiple systems (namely CS) together. Using this extended ontology, finally we could present the SoS-ACRE framework (Fig. 4.8, which is also an extension to the ACRE framework, and contains 2 extra views in addition to the previous 7 views: the validation interaction view, and context interaction view.

Context Interaction View: contains the interactions between the different contexts (CS context and SoS context).

Validation Interaction View: represents the relations that exist between the set of validation on SoS level, with validations on the CS level.

4.3.3.4 Discussion

This description of this framework was extracted from (Holt, S. Perry, et al. 2015). A more detailed description of this framework and other related information could be found on the COMPASS website[11]. In the next chapter, we are going to use this framework as a basis to build our own framework. There are 2 reasons behind this decision:

1. It is compatible to be applied on relatively small scale SoS (Holt, S. Perry, et al. 2015);
2. It is intuitive and easy to understand, and in consequence, we can modify it to suit our purposes.

4.4 Conclusion

Modeling is a process that we use in order to understand a certain aspect of a concept or an object. By abstracting the aspects that are of no interest to us, we are able to focus on the aspect that interests us without dealing with the overall complexity of the modeled entity. Models are the result of the modeling process, and they are used as means of communication, analysis, and decision-making aiding tools. When dealing with systems, there are 3 types of models: first descriptive models which consist of drawings, charts, etc. are used to answer logical questions about the modeled system, such as questions about its structure, objectives, etc. The second type of models is the analytical models, which use mainly mathematical tools in order to answer quantitative questions about the modeled system, such as performance, cost, etc. And finally, the third type is hybrid models, which represent a combination of the two previous types, and they provide a higher level description of the system.
When we want to predict the behavior of a system, or understand the outcome of a certain action in a complex system where the overall outcome is not the sum of the outcomes of the parts, models are very helpful because they give us the ability to perform computer simulations by using them as inputs, coupled with the rules and mechanisms adopted in the system, to predict and analyze behaviors. Due to the complexity and dynamic nature of SoS, frameworks which consist of multiple models, grouped in viewpoints based upon what aspect they describe in SoS, are a tool that is very useful in the context of designing reliable evolutionary SoS. They provide us with the ability to represent all the different aspects that we wish to describe in SoS, and to perform computer simulations to test the adopted solutions, especially when in the case of SoS, it is very complicated, and often unfeasible to perform the tests in the real world. Of course this process of simulating the SoS by using its framework is possible when it is built (the framework) using a language that enables this simulation, such as SySML, ArchiMate, etc.
After understanding what are SoS, their definitions, different types, how to control their CS in order to realize their objectives, and how to model and analyze them, it is now the time to apply that to a real use cases, by developing a SoS that enables

[11] www.compass-research.eu/Project/Deliverables/D211.pdf

cooperation between autonomous vehicles in different types of maneuvers, called the Cooperative Maneuvers Manager for Autonomous Vehicles (CMMAV), which is the focus of the next chapter.

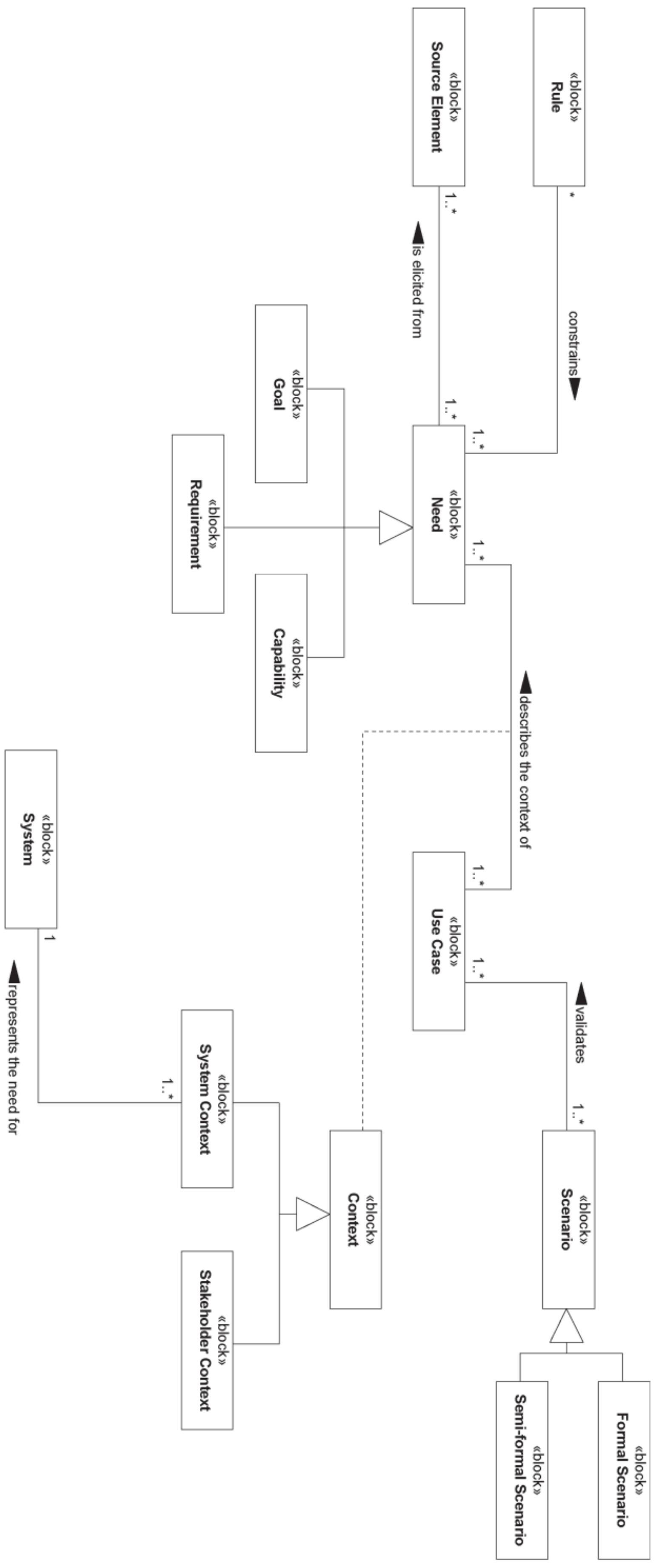


Figure 4.5: ACRE Ontology (Holt, S. Perry, et al. 2015).

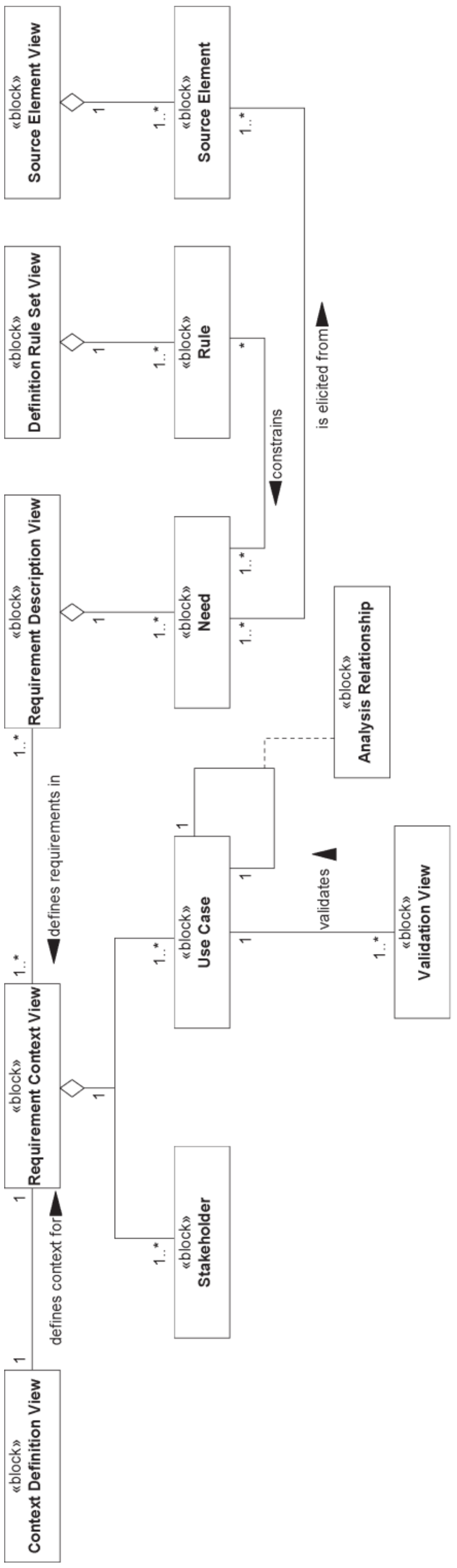

«block» Context Definition View
«block» Requirement Context View
«block» Requirement Description View
«block» Definition Rule Set View
«block» Source Element View
«block» Stakeholder
«block» Use Case
«block» Need
«block» Rule
«block» Source Element
«block» Analysis Relationship
«block» Validation View
defines context for
defines requirements in
constrains
is elicited from
validates
1
1..*
*

Figure 4.6: ACRE Framework (Holt, S. Perry, et al. 2015).

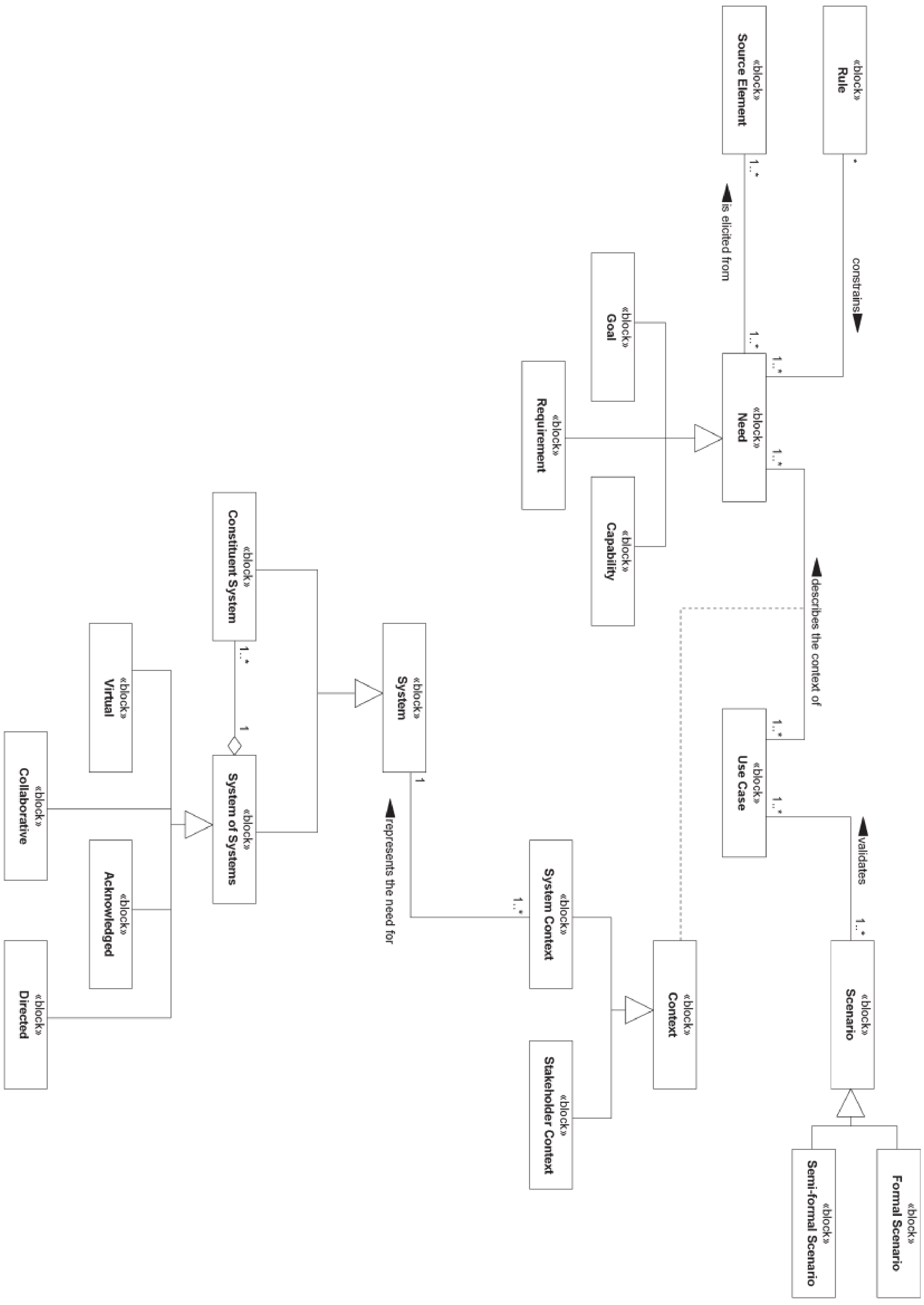


Figure 4.7: COMPASS Ontology (Holt, S. Perry, et al. 2015).

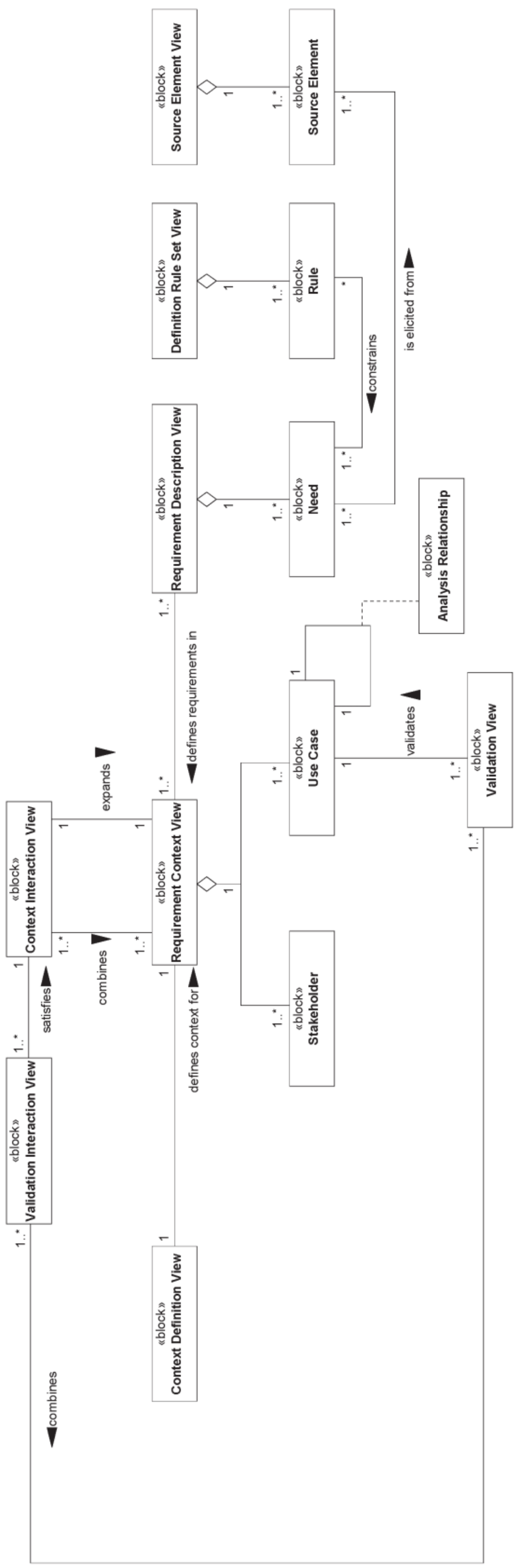

«block» Validation Interaction View
«block» Context Interaction View
«block» Context Definition View
«block» Requirement Context View
«block» Requirement Description View
«block» Definition Rule Set View
«block» Source Element View
«block» Stakeholder
«block» Use Case
«block» Need
«block» Rule
«block» Source Element
«block» Analysis Relationship
«block» Validation View
combines
satisfies
combines
expands
defines context for
defines requirements in
constrains
is elicited from
validates
1
1..*
*

Figure 4.8: SoS-ACRE Framework (Holt, S. Perry, et al. 2015).

Chapter 5

Application: Cooperative Maneuvers Manager for Autonomous Vehicles (CMMAV)

5.1 Introduction

Autonomous cars have been studied since at least the 1920's, with the "phantom auto" was promised to tour the city of Milwaukee in the US [1]. They showed up in science fiction movies and stories long time ago as well. The idea of a car driving itself while the driver is free to do something else is compelling. It gives us the feeling of importance and for some people security. On the other side, some people fear this kind of technology, either because they do not trust machines enough for performing this type of tasks, or simply because they like to drive by themselves and they enjoy doing it. Regardless of this, autonomous driving is the subject of many academic studies, as well as commercial development. Autonomous driving (or navigation) is not exclusive to cars or ground vehicles, aerial vehicles, such as Unmanned Aerial Vehicles (UAV), even though most of them are controlled from the ground, are a technology that exists for a long time now. Satellites and space probes usually have several components for autonomous navigation. As for ground transportation, autonomous trains are now the focus of different manufacturers and railway network operators, and even autonomous bicycles are a thing now!
Autonomous navigation in each of the previous domains differ in the degree of maturity: we could find a fully autonomous commercial UAV, while there exist no commercial fully autonomous cars, for example. This difference in maturity levels could be attributed to the complexity of the environment in which a particular system operates. While any vehicle would need to self-locate and perceive the surrounding environment, among other functionalities, the safety requirements for different vehicles are different (in case of a failure, an autonomous UAV might be allowed to crash in an inhabitable area, whereas an autonomous car crashing in an inhabitable area might not be allowed, since there are passengers on board). Moreover, the surrounding environment in the case of a UAV is usually empty from any type of valuable assets that it would need to conserve, which is not the case for cars, where they share the same environment with pedestrians, other cars, even store and markets in the case of urban navigation. To complicate things further, a car would

[1]https://news.google.com/newspapers?id=unBQAAAAIBAJ&sjid=QQ8EAAAAIBAJ&pg=7304,3766749

not only need to account for the safety of other entities in its surroundings, but it should as well be able to cooperate with them. For example, cooperative adaptive cruising on a road increases its capacity (throughput) and decreases fuel consumption in participating vehicles (Vander Werf et al. 2002).
In this context, and as a contribution to the autonomous navigation of ground vehicles, in particular, autonomous cars, this chapter introduces the Cooperative Maneuvers Manager for Autonomous Vehicles (CMMAV). CMMAV is a framework that defines the logical structure to any user that would like to develop a cooperative maneuvers manager in an autonomous vehicle. However it does not impose any implementation or technological constraints. CMMAV uses Systems of Systems Approach to Context-Based Requirements Engineering (SoS-ACRE) (Holt, S. Perry, et al. 2015), in which the needs (requirements, goals, functionalities) of all constituent systems and stakeholders are the source of the different logical elements presented in the framework. It uses use cases as a starting point to identify stakeholders, uses these stakeholders to develop the framework, and then uses user feedback to refine it. The objectives of this chapter are:

1. Present the CMMAV: Motivations behind it, and its objectives;
2. Justify the relation between CMMAV and SoS, as well as ITS;
3. Explain the framework;
4. Present a general discussion about CMMAV, different possible use-cases, and SoS challenges that it might face.

5.2 Motivations

In order to achieve reliable autonomous ground navigation (AGN) for commercial cars, they must have different functionalities such as self-localization, self-monitoring, etc. We have identified 3 groups of functionalities that need to exist in Ground Vehicles (GV) (Fig. 5.1):

Base Functionalities, a set of functionalities that enable the subject vehicle to navigate autonomously, in a closed, static, and controlled environment. This type includes self-localization and cruise control, for example.

Environment Functionalities, a set of functionalities added to the subject vehicle in order to safely operate in a dynamic, uncontrolled environment. Functionalities such as detecting obstacles, and vehicle-to-X (V2X) communications belong to this type.

Collective Functionalities, a set of functionalities that improve the performance of one or more previous functionalities, or the aggregation of which gives birth to new functionalities for the overall system. An example is cooperative formation driving where vehicles decide on a specific formation in their movement.

Base and environment functionalities are studied, developed, and some of them are commercialized. However, there is a gap in applications that target collective functionalities. The ability for autonomous cars to communicate requests and to exchange services are not well established, and not many studies on the subject exist.

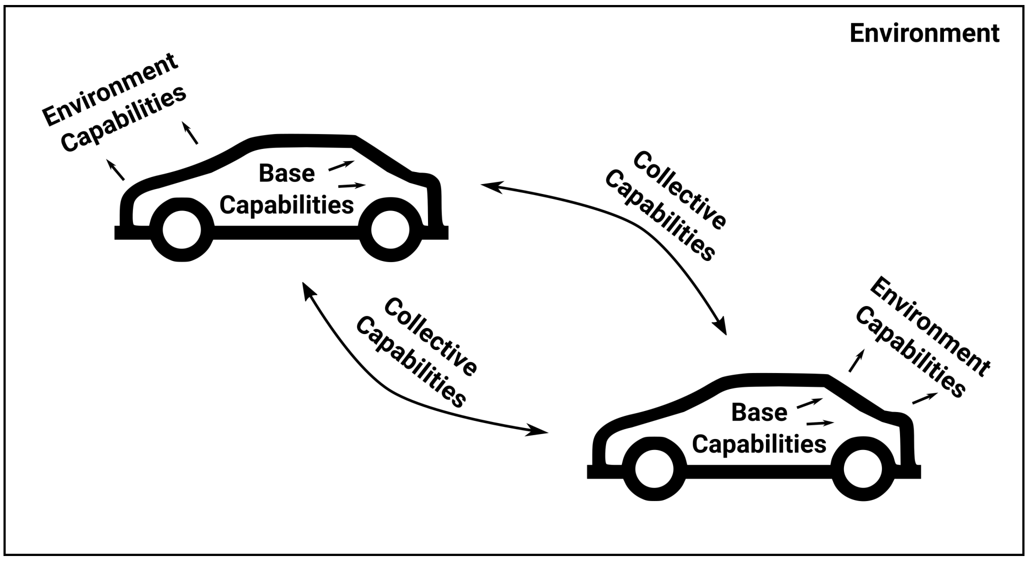


Figure 5.1: Functionalities types: base, environment, and collective.

This is due to the fragility of such functionalities towards the surrounding environment. For example, the presence of a human-driven car, with the unpredictability of the driver alongside autonomous vehicles might affect the operation of such functionalities. Moreover, to safely enable collective functionalities, we must ensure the resilience of both environmental and base functionalities. One more reason is the heterogeneity of the players in the system: there are different car manufacturers, where each uses different technologies and methods to enable the different functionalities in their vehicles. One more challenge that faces these functionalities is the different objectives of the autonomous cars that share the same road. Each car has its objectives set by its owner, which might be in conflict with neighbor cars. This subject is treated by game theory studies [(Jaramillo and Srikant 2010), (Su et al. 2007)], which contributes to the development of such functionalities. The deployment of such functionalities is the natural next step after developing robust environment and base functionalities. There are several projects[2] that use autonomous vehicles to prove and refine the technology, such as Autonomy[3] and AutoC-ITS project[4].
On the other hand, standards have proven their importance in environments that contain divers technologies, actors, and systems interacting all together. Standards are a way to harmonize these types of environments, while at the same time respect the nature of its environment. For example, in the mobile phone's world, there are numerous manufacturers that each uses different parts, and different software, while at the same time they offer the ability to work with routers, reception towers, and phones from other manufacturers. The reason why they are capable of doing this is because they follow standards. On this aspect, transportation world is very similar to mobile phones world, and not surprisingly, there are standards that guide things in it as well.
The majority of the challenges that face collective functionalities are similar to some of the challenges that face SoS: different constituent systems, with conflicting objectives, that operate in a dynamic environment that they share with other systems

[2]imovecrc.com/smart-mobility-projects-trials-list/
[3]www.autonomy.paris
[4]project.inria.fr/autocits

that are also independent and have different objectives. This similarity inspired us to use SoS approach, to develop the CMMAV, as a standard, or framework that guides the development of collective functionalities in autonomous vehicles.

5.3 CMMAV Description

CMMAV is a framework that proposes a logical architecture which could be used by any entity that might be interested in developing cooperative functionalities in autonomous vehicles. The importance of a unified architecture is proven by the presence of the diverse organizations that set standards in different domains (ETSI standards[5], IEEE standards, ISO standards, etc.). CMMAV could be considered as the first step towards a standard that guides the development of cooperative functionalities for autonomous vehicles. Being a logical architecture, it does not set implementation constraints either on the software or the hardware used to provide these functionalities. For example, CMMAV requires that CS must be able to locate nearby vehicles. However it does not mention how they must achieve that. Consider the users U_1 and U_2 who are developing the functionalities C_1 and C_2 for their CS respectively. By using the CMMAV, they could focus on developing their own functionalities, while at the same time, if later on they decide to develop a third functionality C_3 which allows their CS to cooperate, their systems are already compatible for integrating C_3.

> ***CMMAV is a framework that guides the development of collective functionalities in autonomous vehicles.***

CMMAV treats mainly two points: providing incentive for cooperation, and addressing emergent behaviors resulting from potential cooperation. More detailed discussion around these points will be provided later in this chapter. Three types of stakeholders exist in the CMMAV: maintainers, constraining stakeholders and users stakeholders (Section 5.7). Maintainers are responsible for developing and maintaining the CMMAV, constraining stakeholder such as the state and car manufacturers provide rules that must be respected by CMMAV, while user stakeholders are any entity interested in using CMMAV for a specific use case, this type might include universities, logistics companies, or transportation companies. CMMAV uses users use-cases as the starting point in its drafts. Use-cases provide stakeholders, which mainly provide requirements and feedback for the entity maintaining the CMMAV development in order to refine it and improve it. The process of developing CMMAV is described in Fig. 5.2: For each new identified use case (see Section 5.9),

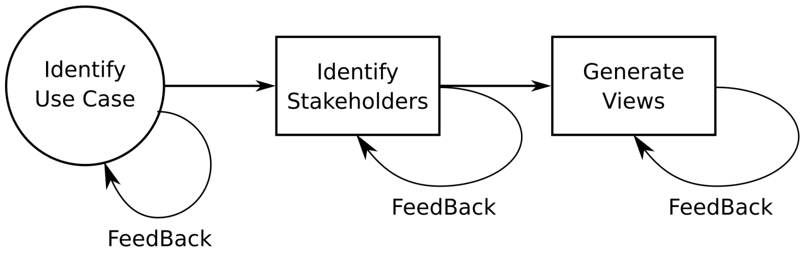


Figure 5.2: CMMAV development process: use-case based.

different involved stakeholders are identified, which are then used as an input to the

[5]The European Telecommunications Standards Institute

second process, to generate the different views of the CMMAV (see 5.11). We chose the "overtaking on high ways" (see appendix A.1.2) use case as a starting point to develop the first draft of this framework. The logical blocks proposed by this draft, however, may not be suited to be applied to another use case, such as "multi-vehicle delivery systems" for example. Later on, when new use cases are provided, CM-MAV maintainers integrate its newly added stakeholders and/or requirements to refine it. The loops shown on each process represent the feedback that could happen at any stage of development: use cases, stakeholders and requirements might change, and for each change, the process itself as well as the subsequent processes must be reviewed to ensure that the CMMAV is up-to-date.

5.4 CMMAV & Intelligent Transportation Systems

CMMAV is designed to standardize the development of collective functionalities in ground vehicles or systems in their environment. When working with any transportation system, it is very important to understand the placement of this system in a larger, global SoS, called the Intelligent Transportation Systems (ITS). This is important because it provides us with a very important knowledge about the ecosystem of the systems that we are dealing with, and boundaries about what we should respect and follow during the development of the CMMAV. The world is con-

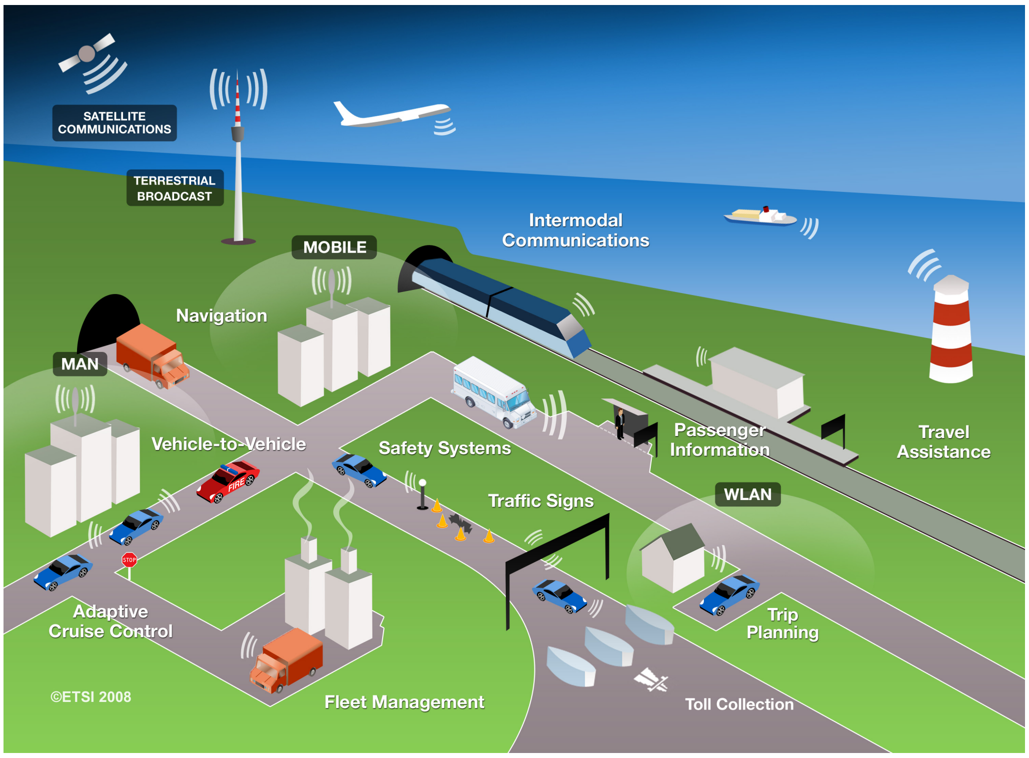


Figure 5.3: Intelligent Transportation Systems: a global SoS (ITS 2008).

nected by a large transportation network. Everyday, thousands of trips are made to transport goods or people from a place to place on water, ground, and in air. All the trips made have the same objective: deliver something from point A to point B. However, each trip has different stakeholders that have different requirements: when transporting goods often safety is required, while when transporting people, comfort

is required in addition to safety. The mode of transportation (train, airplane, etc.) also affects the trip: traveling by plane is faster than doing so in a train for example. Furthermore, the source and destination of a trip may constrain the trip as well, because every place has different laws and regulations that must be respected by the trip (the transported thing and/or the transporting entity).
Every stakeholder has a different interest in transportation systems, and affect it in a certain way. Countries have regulations on transportation systems, manufacturers produce transportation vehicles, travelers use the system to travel, service providers develop technologies to improve the system and gain profits, employees work in transportation systems. The list of stakeholders is very big and diverse, but since they all share the same objective, and each has different other set of objectives that might be conflict, a global transportation SoS emerged that they all form part in.

5.4.1 Intelligent Transportation Systems (ITS)

One particular SoS that belongs to the global transportation SoS is ITS. ITS are any system, technology, service, or organization that has the objective of improving the state of any transportation system. ITS by itself is not a system, but it is the placeholder of myriads of applications, infrastructure, standards, and technologies, that seek to make more intelligent roads and vehicles [Figueiredo et al. 2001].

Example 5.4.1. *Mobile applications that use the positions and speeds of different vehicles in an area to deduce the state of traffic in that area, and to provide suggestions to users about best routes belong to ITS.*

5.4.2 ITS Stakeholders

The stakeholders of ITS include mobile app developers, transportation companies, vehicle manufacturers, governments, syndicates, citizens, standardization organizations, etc. (Figure 5.4). Each stakeholder involved has different objectives, incentives, and requirements, and different types of relation to ITS. While a mobile app developer develop an app to provide routing services for travelers, a government sets traffic laws to guarantee the safety of its citizens, laws that must be respected by travelers and manufacturers in the ITS. Recognizing stakeholders of the ITS is important during identifying the stakeholders of the CMMAV (see Section 5.7).

5.4.3 ITS Categories

The systems that belong to ITS seek to improve transportation by targeting different parts of the system. Some systems target vehicles by developing technologies and applications to improve localization or control of vehicles for instance, while others target users by providing routing services. This allows us to separate ITS into 6 major categories (Wu and P.-J. Lee 2007):

Advanced Traffic Management Systems (ATMS): The focus of this category is to provide the necessary information about traffic to the infrastructure systems that can be used to better control traffic flow.

Advanced Travelers Information Systems (ATIS): This category provides real-time traffic information to the travelers, to enable them to make better decisions regarding routing, which help in the reduction of congestion.

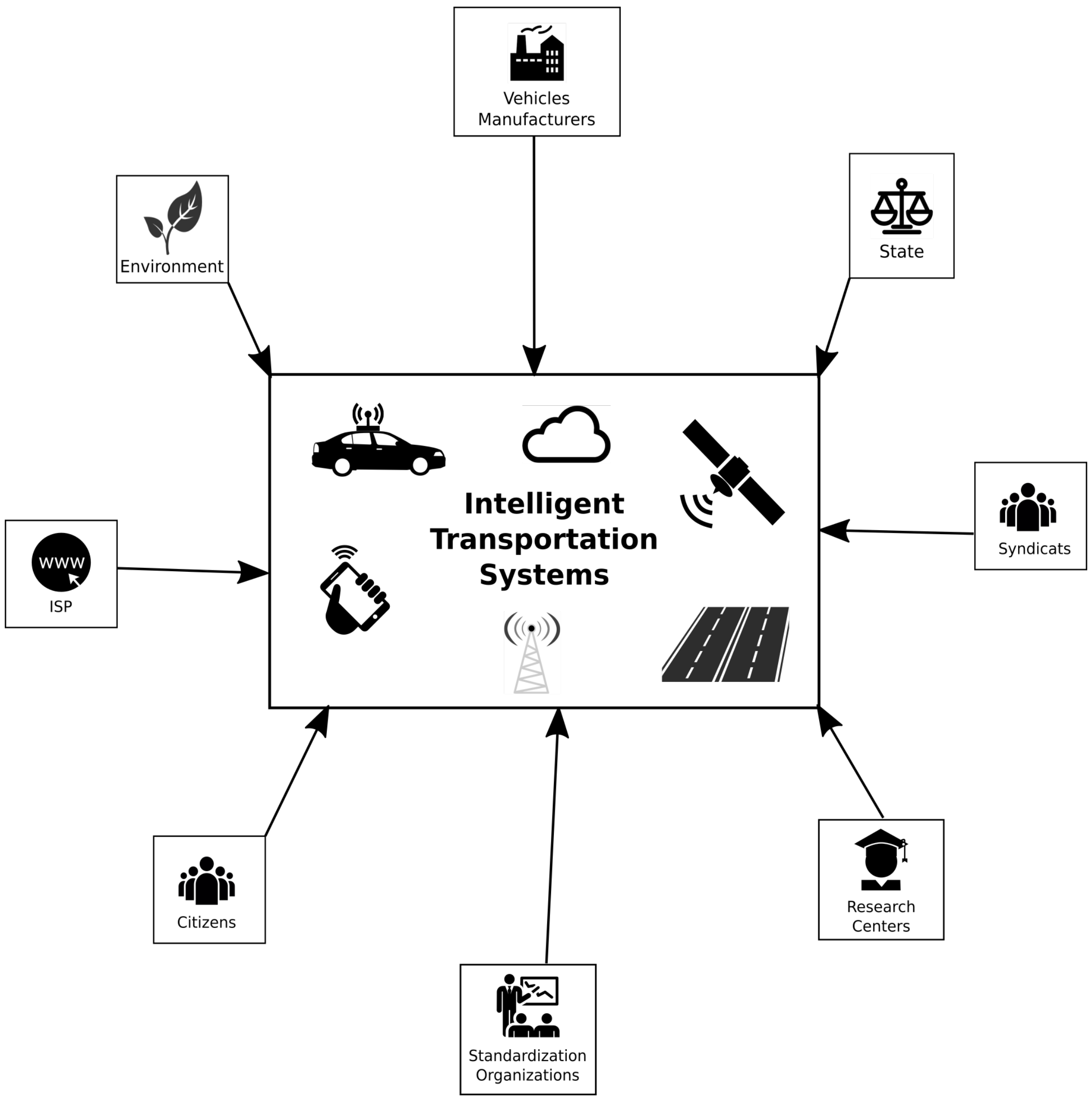


Figure 5.4: Intelligent Transportation Systems: Stakeholders.

Commercial Vehicles Operation (CVO): CVO's main beneficiaries are mainly logistics companies, and any entity that has commercial fleets. CVO systems allow these entities to better control their fleets, which results in better goods deliveries.

Advanced Public Transportation Systems (APTS): This category uses the ATMS and ATIS systems in order to optimize public transportation such as buses and trains.

Advanced Vehicles Control Systems (AVCS): AVCS is where in-vehicle sensors are used to assist drivers either by informing them, or by taking control over some functionalities in the vehicle.

Advanced Rural Transports Systems (ARTS): ARTS use different services from the previous categories in order to solve some problems in rural zones (alerting the driver about a blind curve for example).

5.4.4 CMMAV in ITS

Based on the previous Section 5.4.3, to locate CMMAV in the ITS, we can identify its potential users. CMMAV guides the development of applications that could be used to provide collective or cooperative functionalities in a system of interest (SoI). Just as any standard that exists in the ITS and guides the development of a particular group of products that might be used in several categories, the products developed by following CMMAV's recommendations might belong as well to different categories. In its current format, CMMAV targets AVCS, since it only takes passenger cars into consideration during its development (see Section 5.6). However, if a use case was to be added that involves a public transportation system for example, CMMAV then could be classified as CVO system as well. This placement helps us identify the practices used in that category, which will provide us with verification, deployment, and operational requirements.

5.5 CMMAV & SoS

We have mentioned in the motivations that CMMAV faces a lot of the challenges faced by SoS: from a wide range of different stakeholders, to constituents that have conflicted objectives. In fact we consider the CMMAV to be a case study on the development of such systems. By considering this, we enrich ourselves with lessons learned from this domain, and we seek to give back by contributing to the development of SoS. Sharing the same challenges is not enough to classify the CMMAV as a SoS though, but proving that Maeir's characteristics apply to the CMMAV is enough (Mostafavi et al. 2011): prove the operational and managerial independence of the CS.
As discussed in Chapter 2, Maier's characteristics contain two primary and three secondary characteristics. Starting with primary characteristics, in CMMAV every CS is an independent system that operates by itself: every AV operates without the need to rely on any other system. Thus CS are operationally independent. As for managerial independence, every AV is managed by two parties: the owner, who manages the logistical part (providing the vehicle with energy, choosing a destination and route, etc.), and the manufacturer, who manages the technical part (automated parking, adaptive cruise control, etc.). Based on the way we intend to deploy our system, it is up to the owner/provider of the vehicle to install this software in the vehicle. It is true that by doing so, the owner loses some managerial independence for the SoS, but this loss is covered by the gains from belonging to the SoS. So CS in CMMAV are operationally and managerially independent, thus the CMMAV can be classified as SoS. As for the secondary characteristics, another distinction that helps CMMAV development is defining its type in SoS, since this gives an insight about the best type of relation the CMMAV must use with its different stakeholders in order to succeed. CS cooperate using sensing information as well as communications, and thus, the system is geographically distributed. As for the evolutionary development, CS, namely autonomous vehicles in our case, and the environment in which the SoS exists (infrastructure, ITS services, etc.) all evolve independently of the SoS, and thus the SoS should be in constant evolution to keep up with these changes. The second important classification is the category to which our SoS belongs. We classify our system to be a collaborative SoS because the objectives are shared between the SoS and its CS, and the SoS is the result of the collaboration

between its CS.

5.6 CMMAV: Constituent Systems

CS are any physical system that contains software developed based on the CMMAV, and they are defined in use cases. They are important to the CMMAV because different types of CS have different stakeholders (passenger cars and commercial cars for example), and because they are the physical representation of the CMMAV. From "cooperative overtaking" (see appendix A.1.2) use-case, CS are identified as vehicles that use highways and perform overtaking maneuvers, which will be mainly autonomous electrical passenger cars. Let's take a closer look on the ecosystem of such vehicle (Fig. 5.5): AVs use infrastructure (e.g. roads and road signal),

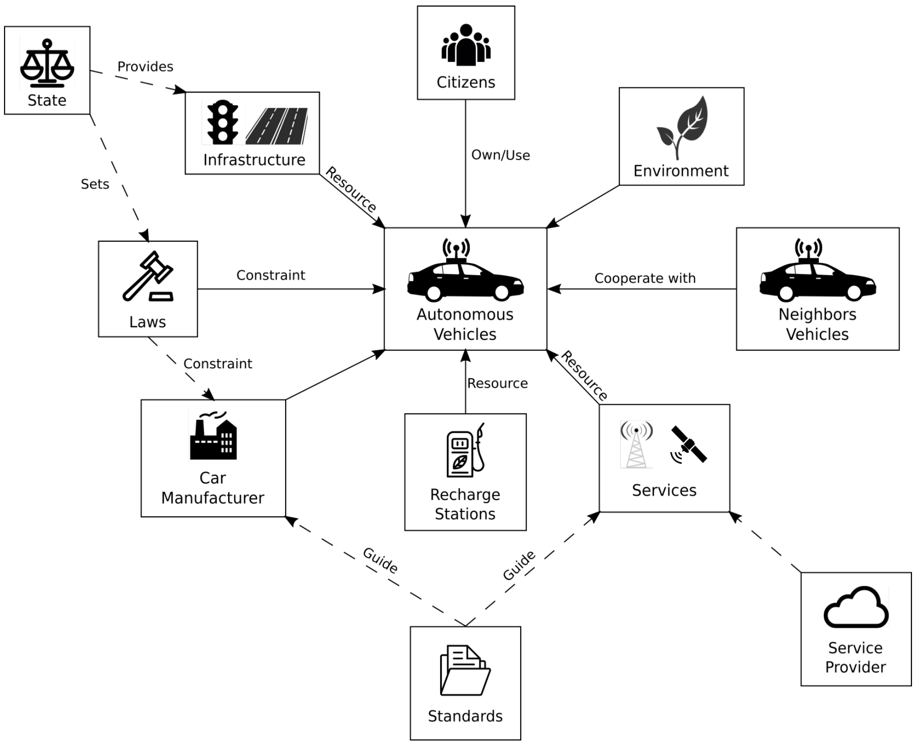


Figure 5.5: Autonomous vehicles ecosystem: different stakeholders and resources.

services (e.g. GPS, cloud services), and recharge stations. By using these resources, they are abiding the constraints and rules set by these resources providers, hence the dashed lines relations, which show a second degree relation to the AVs. For example, the relationship between the state and autonomous vehicles is indirect and it materializes through traffic laws. One relation of interest in this section is the citizens/Autonomous Vehicle relation, described by "own/use". There are multiple scenarios that could define this relation, we consider here two: ownership, and pay-per-use.

Ownership: the vehicle is owned by the person who uses it, namely the owner. The owner decides what kind of applications, services, and resources he/she wants to use in their vehicle, as well as setting the objectives for road trips.

Pay-per-use: the vehicle is owned by an entity (company, organization, etc.), namely the provider. The provider decides what kind of applications, and resources they use in their vehicle. However, the entity that pays to use the vehicle, namely the user, is the one responsible of setting the objectives for road trips.

The reason behind this distinction between the two types is that since each vehicle is a system, it is important to identify the different entities that manage that vehicle, which will be useful later (defining incentive mechanisms, identifying stakeholders, etc.).
Finally, if we apply the representation from 2.3 to an AV as a SoI, using Fig. 5.5, and different functionalities types from Section 5.2, we get Fig. 5.6. Note that we do not

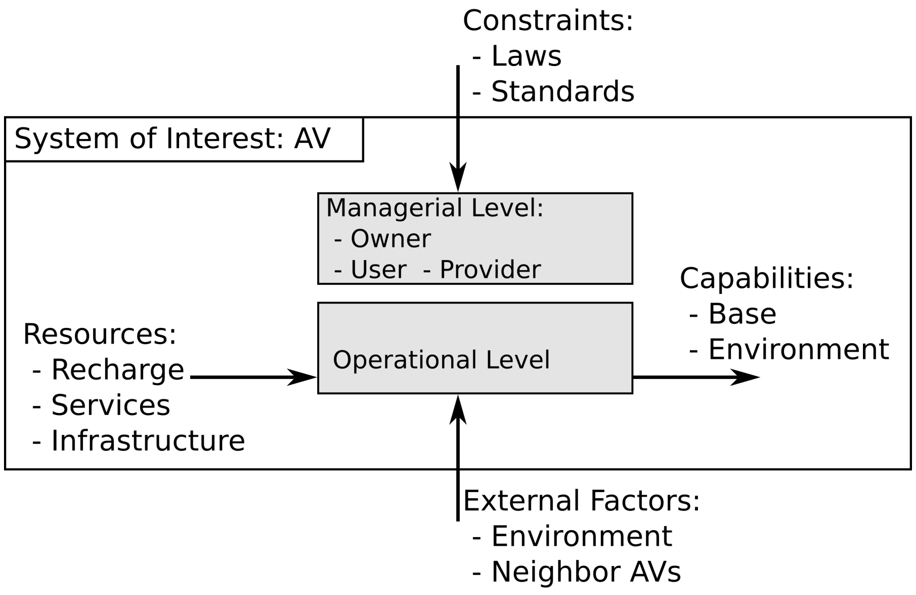


Figure 5.6: Autonomous Vehicle as an independent system: management, functionalities, resources, external factors, and constraints.

consider the operational part of the AV, but rather the functionalities it provides. Furthermore, we consider that the AV does not have any collective functionalities, since this is the objective of this framework (or have some of them but they were not CMMAV based). As for other types of functionalities, CMMAV contains a list of the different base and environment functionalities it requires to operate (Section 5.11.5).

5.7 CMMAV: Stakeholders

Figure 5.5 contains the different stakeholders involved in AV's ecosystem: the state, manufacturers, service providers, etc. Some of them have direct relation to the AV (manufacturer, owner), others have indirect relation with the AV (service providers). We separate stakeholders into 3 different groups: constraining stakeholders, user stakeholders, and maintaining stakeholders. This separation is based on the types of relationship each stakeholder has with the CMMAV: CMMAV is developed and maintained by its maintainers, it respects the requirements of both user and constraining stakeholders, and it is used by user stakeholders to produce a product or a service. Stakeholders can move from one category to another, for example, if a constraining stakeholder developed a product based on the CMMAV, it becomes a user stakeholder (e.g. a standardization organization adapted the CMMAV in its standards, it moves from constraining stakeholders to user stakeholders). CMMAV's stakeholders are not limited to those of its CS. Since the CMMAV is intended to be a standard, it should respect the requirements of the specific standardization organization that might adopt it, and being based on the MBSE discipline, it has to include its requirements (verification and validation for example) in its development, and of course, belonging to SoS means it uses SoSE guidelines as well. The complete list of current stakeholders in CMMAV may be found in Appendix (Fig. A.1).

5.7.1 Constraining Stakeholders

Constraining stakeholders are any stakeholder that does not use the CMMAV. Car manufacturers are constraining stakeholders since they produce cars that use applications developed based on the CMMAV, and therefore, they affect the deployment, and operation of those applications. CMMAV uses the requirements of these stakeholders either as guidelines and indications, or as rules that must be followed. For example, using ETSI[6] standards in communication is recommended in applications that run on European soil, but users may not use these standards depending on their use case, however, in some use cases users must follow traffic laws of their country.

5.7.2 User Stakeholders

User stakeholders are any entity that uses the CMMAV to produce a product or a service. For example, a logistics company that deploy software based on the CMMAV in its vehicles to perform a mission is a user stakeholder that affects the CMMAV by providing requirements related to their use case, but at the same time gets affected by it through adopting its guidelines. User stakeholders differ from constraining stakeholders in that in addition to affecting the CMMAV, they are affected by it as well. They have a special relation with the maintainers stakeholders in that they help the latter refine the CMMAV throughout its constant development cycle by providing feedback about the performance of their applications (Fig. 5.2). User stakeholders are usually added with every new use case, as they are mainly the ones providing those use cases. A user stakeholder may provide multiple use cases.

5.7.3 Maintaining Stakeholders

Maintaining stakeholders are any entity that develops and reviews the CMMAV. Upon any change in the requirements of one or more of the previously mentioned stakeholders, and in case any new use case was submitted, the job of maintainers is to modify and refine the CMMAV to ensure that it meets its requirements at all time. Right now the only maintainer stakeholder in the CMMAV is the Heudiasyc Laboratory. If any other organization collaborated with the maintainers in developing the CMMAV, they become maintainers as well.

5.8 Incentives for cooperation

The distinction we made between ownership and pay-per-use of the CS in Section 5.6 is very useful when thinking about inciting cooperation between CS. CS are independently managed. By using CMMAV based functionalities for cooperation, they are expected to ask for and provide services from/to other CS. They are not, however, obliged to do so. To incite cooperation between CS, incentives for cooperation must be provided. To cooperate, a CS sends a request to another CS, asking

[6]www.etsi.org

for a service. The second CS responds by either acceptance or refusal of the request. Incentives are given to favorite the acceptance of a request (or set of requests). They might be in the form of points gained for each accepted request from a CS that could be spent by the owner on goods, or that could be used by other CS to define their responses to the requesting CS (cooperating more with others means others will cooperate more with you). Incentives are very important in any cooperation, and therefore CMMAV maintainers require users to prove that the incentive mechanisms they have chosen are valid (see more in emergent behavior Section 5.10.

5.8.1 Cooperation Decision-Making

The decision that defines the acceptance or refusal of a request in CS might have different sources in different systems. In AGV, the decision is expected to be made inside the vehicle, while in a UAV, it might come from a ground control center. The position of this source in a system differs from system to system. However it always comes from the owner of the system. For example, an AGV driver who owns their vehicle might decide to take each decision by themselves, or let the vehicle take the decision via a specific application they installed. Think mobile phone users, by allowing different apps to know the position of their phones, they are providing services to these applications (whether they know it or not). The allowance to a certain app to access the position of the mobile phone and denying this access from another by the owner of the phone is similar to the driver taking each decision by themselves. However, when the phone's owner chooses a predefined mode in their phones that automatically grant or deny access to applications based on the chosen mode, this is similar to the driver letting the vehicle take the decision by either setting a predefined mode (very cooperative, cooperative, selfish for example), or installing applications that take this decision based on their preferences.

5.8.2 Examples of Incentives

There are several incentive mechanisms that might be used to favorite cooperation between systems. Stakeholders in the system play an important role in defining those mechanisms. Here are some examples to make things clearer.

- In cooperation between different civilians AV, consider some sort of "social points" system between vehicles, where a profile that represents the vehicle is accessible to all neighbors. This profile mirror the level of cooperation of a certain CS. More cooperation means better profile, which consequently means more chances that other CS will collaborate with you.

- In cooperation between vehicles owned by a logistics company and a service provider company when the first company pays a fee to the latter. CS do not need incentives as they are required to cooperate by their respective owners. The incentive of cooperation in this case resides within the owners, and takes the form of a paid fee in exchange for this cooperation.

- In the "shared parking" use case, incentive is given in the form of points to parking owners in exchange for the services they provide, that could be spent to gain services from service providers (charging stations, cloud access, etc.).

The marketing domain contains a lot of incentive mechanisms that could be used that are well studied and applied (Gao et al. 2015). More information on this subject is provided in Section 5.10.

5.9 Use Cases

As discussed earlier in Section 5.3, use cases are the source of CMMAV's requirements and stakeholders. A use case provides stakeholders, as well as requirements to the CMMAV. Use cases are provided to the CMMAV by users who decide to adopt its guidelines in the development of some of their functionalities. A user who wishes to submit a use case fills out the specific template provided by the CMMAV, and sends it to CMMAV maintainers. Maintainers verify that the submitted use case requirements are not treated in the CMMAV, and then proceed to perform the steps described in Section 5.11 to adopt this new use case. The first use case treated in this framework is the overtaking use case (described in "Overtaking use case submission document" in the Appendix A.1). It consists of enabling AGV to perform cooperative overtaking on high ways.

5.9.1 Use Case Examples

Here we list some of the potential use cases of the CMMAV.

5.9.1.1 Localization of a vehicle not transmitting its proprioceptive information

Communicating through wireless communication mechanisms face the problems of packet loss. Sometimes vehicles might lose the functionality of communicating important information about their location and speed. However, using the different positions and speeds estimated by neighbor vehicles and possibly infrastructure, the position and speed of the non-communicating vehicle could be estimated by a margin of error. This use case requires the vehicles to be able to identify neighbors, estimate their positions and speeds, and share this information.

5.9.1.2 Shared Parking

Consider an urban area, where private parking places are empty during day time (owners are at work). An empty AV (does not have any passenger) which is searching for a place to park while their owner is doing something else, could search for an empty parking spot. Based on different potential incentive criteria (social points, pay-per-use, etc.), private parking places accept to let the empty AV park for a period of time. User stakeholders in this use case might include parking spot owners, vehicle owners, commercial stores, service providers, etc.

5.9.1.3 Framework Process Use Case

This example is different than the above. This is a use case that uses the process of developing the CMMAV to develop a framework that caters to a specific system's needs. Consider a project to build a system that involves different teams, each is responsible of the development of a part of the system. By using tasks as use cases, and developing the framework by adopting the process described in this chapter, the managers of this system and all involved stakeholders will have concrete and clear knowledge about their system.

5.10 Emergent Behaviors

By considering SoS approach while building the CMMAV, one important aspect must be taken into account during conceiving, building, and deploying it: Emergent

Behaviors. This topic is discussed earlier in Chapter 2, and here we address its manifestation in CMMAV. By using the CMMAV to develop and deploy collaborative functionalities in CS (AV, infrastructure, etc..), rather than creating new interactions in the environment, we are refining them. Consider for example the overtaking use case, overtaking happens everywhere between human-driven cars, and sometimes there is cooperation between drivers: when we signal the intention to change a lane, often drivers on the other lane slow down a little bit or keep constant speed to allow us to execute the maneuver. What we are really doing, is changing the cooperation decision-making process, or rather providing standard mechanisms for this cooperation to happen. Since this type of cooperation happens already, it is very important to study the existing emergent behaviors in such ecosystem to account for it when developing these collaborative functionalities. In the current version of the CMMAV, we have identified two emergent behaviors that must be treated: Traffic shocks due to the accordion effect, and incentive compatibility. Emergent behaviors are required to be treated in CMMAV based applications by their maintainers, using verifications from the verification view (Section 5.11.6) that users must respect. The first behavior is related to the physical manifestation of the cooperation desired by the users, while the second is related to the incentive mechanisms used in such applications.

Traffic shocks and the accordion effect: This effect happens in any physical system that contains flowing elements. In traffic, it manifests when in traffic flow, a vehicle slows down, which leads to the following vehicle slowing down as well. This creates a backward ripple of vehicles slowing down, each more than the previous, which is similar to the accordion, and usually leads to congestion in the back of the traffic flow.

Incentive compatibility: In economics, incentive compatibility constraint (ICC) is considered in problems where an agent is expected to provide a service to other agents (Mehta and Vazirani 2018). ICC ensures that the expected utility of exerting high effort is higher or equal to the expected utility of exerting low effort. This is related to the CMMAV since it considers the interactions between CS as services exchanged. The problem is how to incite CS to provide services to other CS, and the incentive compatibility behavior might occur if there exist no incentives for cooperation for example, then it is expected that all CS will request services, but none will provide them.

5.11 The Framework

Using SoS approach, the best way to represent or model a SoS is by developing a framework with different views that provide each stakeholder with the necessary information regarding the SoI. Using the same approach, we have decided to use SoS-ACRE, the framework proposed by (Holt, S. Perry, et al. 2015) as a basis for CMMAV: Using a model-based approach for requirements engineering, we build a multi-view framework that maps the requirements of different stakeholders involved in a SoI. The use of this framework comes after choosing the stakeholders (second part in Fig. 5.2), which are used to populate the needed views. The CMMAV contains 7 views:

1. Use-Cases View

2. Organizational View
3. Sources View
4. Requirements View
5. Functionalities View
6. Verification View
7. Traceability View

These views are related together via a set of different relations. Each view (or set of views) is responsible for populating a subsequent view (or set of views). “Use Cases View” provides the “Organizational View” with stakeholders, which in turn gives sources to “Sources View”. The complete process of populating and refining the CMMAV is shown in Fig. 5.7. At the end, the first six views provide relations to the traceability view to create the different trees representing the different relations. Two main models are used to represent the different views: SysML diagrams and

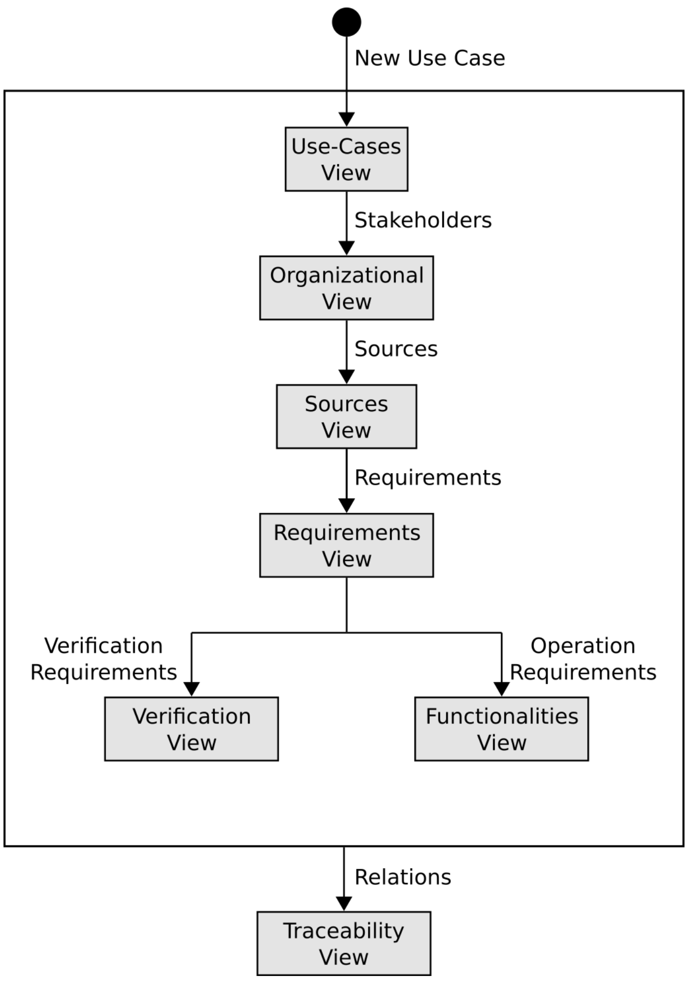
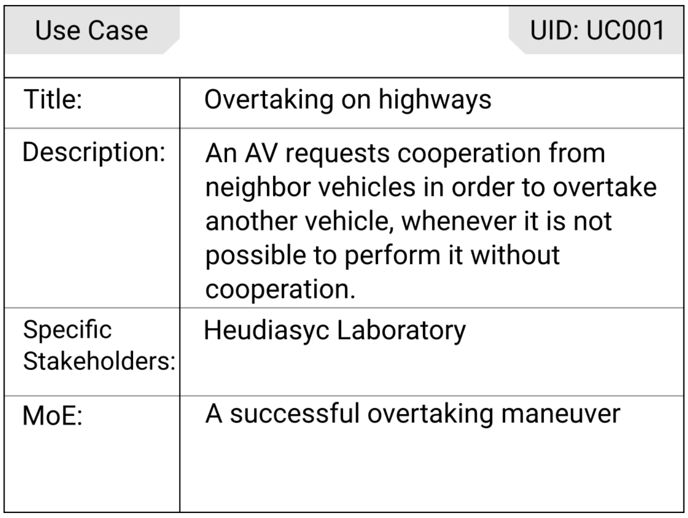


Figure 5.7: CMMAV framework process.

forms. SysML diagrams are used to give a high-level view of an aspect of the system, particularly to show relations between different elements of a view, while forms are used to provide a detailed description of each element of a view. In the high-level view, elements in dashed boxes are not considered in the current version of the CMMAV, but added to highlight potential use. In each view, elements are represented in a form defined by a template specific to that view, which contains several attributes used to describe these elements. One of the uses of the feedback mechanism shown in Fig. 5.2 is that user stakeholders may suggest modifications not only to the content of CMMAV, but to the templates as well, by adding or removing specific attributes from specific view(s). In addition to the different views, CMMAV contains a document that describes the purpose of each view, and the template used to represent elements in that view, and another document that contains all the acronyms used within with their meanings.

5.11.1 Use-Cases View

CMMAV is based on use cases. They provide it with requirements that come with the different stakeholders involved in a use case. The use cases view contains all use cases that were used to build the CMMAV. A use case describes how a certain user stakeholder intends to use the CMMAV. Each use case is described by a form that has the following attributes: a unique identifier, a description, and specific stakeholders. The first use case, "overtaking on highways" is shown in Fig. 5.8, as an example of a use case in this view. More potential use cases are presented in Section 5.9.



Use Case	UID: UC001
Title:	Overtaking on highways
Description:	An AV requests cooperation from neighbor vehicles in order to overtake another vehicle, whenever it is not possible to perform it without cooperation.
Specific Stakeholders:	Heudiasyc Laboratory
MoE:	A successful overtaking maneuver

Figure 5.8: Use Case: Cooperative Overtaking.

5.11.1.1 Unique identifier

This unique identifier (UID) is added for the purpose of enabling better traceability in the CMMAV. It is used to link any source, requirement, rule, or view of the CMMAV to the specific use case it is related to.

5.11.1.2 Description

Every use case must have a description that describes the intended use of CMMAV by a user stakeholder. This objective of this description is to explain the use case in a general way, so it must be short and it must convey the general idea of the use case.

5.11.1.3 Specific Stakeholders

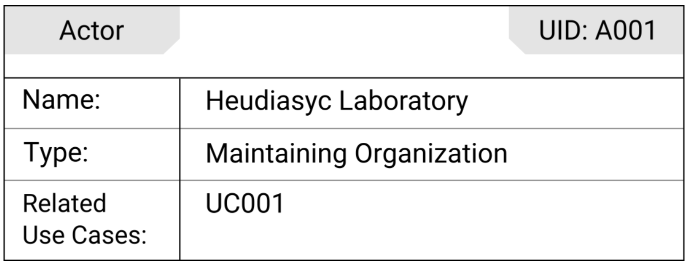
Specific stakeholders are any stakeholder that relates to the CMMAV through this use case. This is added mainly because whenever a use case is modified, it might be useful to know which stakeholders are affected, and if needed, should be consulted.

5.11.1.4 Meaures of Effectiveness

Measures of Effectiveness (MoE) are all the criteria that define the success of the use case. They are usually defined by specific stakeholders of the use case, they might be formulas or formal conditions.

5.11.2 Organizational View

This view describes the architecture of CMMAV from an organizational point of view: maintaining organizations, constraining stakeholders, user stakeholders, and CS. The maintaining organizations are the group of organizations that are responsible for developing and evaluating the CMMAV, stakeholders include constraining and user stakeholders, and CS are any physical system that deploy software based on the CMMAV. Each element in this view has a UID, a name, a type (CS; User Stakeholder; Constraining Stakeholder; Maintaining Organization), and use case field containing UID of all of the use cases that this element is involved in. Figure A.1 (Appendix) is the current version of the organizational view, and Fig. A.1 shows the detailed forms of the different stakeholders. Figure 5.9 shows an example of an element of this view. These stakeholders were chosen for the following reasons:



Actor	UID: A001
Name:	Heudiasyc Laboratory
Type:	Maintaining Organization
Related Use Cases:	UC001

Figure 5.9: Actor Details: name, type, and related use cases.

User Stakeholders: The Heudiasyc laboratory, by providing the use case used to build the current version of the CMMAV, is considered as user stakeholders.

State: Considered because the use case requires the respect of state laws during all maneuvers, and so the state becomes a constraining stakeholder.

ETSI: By adopting ETSI standards for telecommunication between CS (for applications operating in Europe), ETSI becomes a constraining stakeholder.

Maintainers: By developing the CMMAV, Heudiasyc is considered maintainers stakeholders.

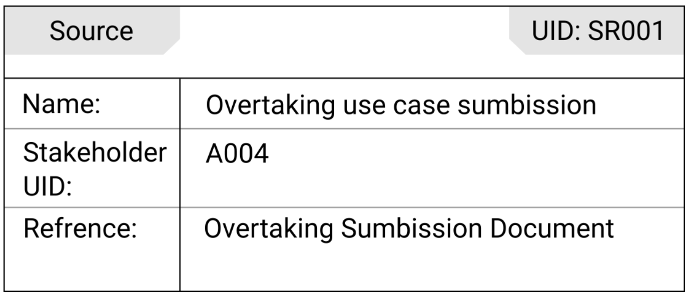
CS: AVs are considered as CS as described in Section 5.6.

Service Providers: Added to highlight incentive potential, see Section 5.8. Not considered in the current version.

Manufacturers: Added to highlight potential deployment requirements from manufacturers. Not considered in the current version.

5.11.3 Sources View

This view provides the sources of all requirements in the CMMAV. Each source provides the requirement(s) of an element of the previous view (Section 5.11.2), which may have multiple sources in this view. A source has a UID, a name, and the UID of the corresponding stakeholder, and a link to the document (or website, or email exchange) that this source represents (Fig. 5.10). The complete list of sources is in Section A.2.2 (Appendix). Traffic laws define the requirements of the state. ETSI EN 302 637-2 V1.3.2 (2014-11) defines the requirements of ETSI concerning communication between CS. User stakeholders provide the CMMAV with a use case document that contains their requirements (Appendix A.1). Exchanges represent any communication that might occur between users and maintainers concerning adding or modifying requirements. Maintainers provide requirement document that contains their requirements to the CMMAV. Figure A.2 (Appendix) is the current version of the sources view, and Fig. A.7 (Appendix) shows the detailed forms of the different sources.



Source	UID: SR001
Name:	Overtaking use case sumbission
Stakeholder UID:	A004
Refrence:	Overtaking Sumbission Document

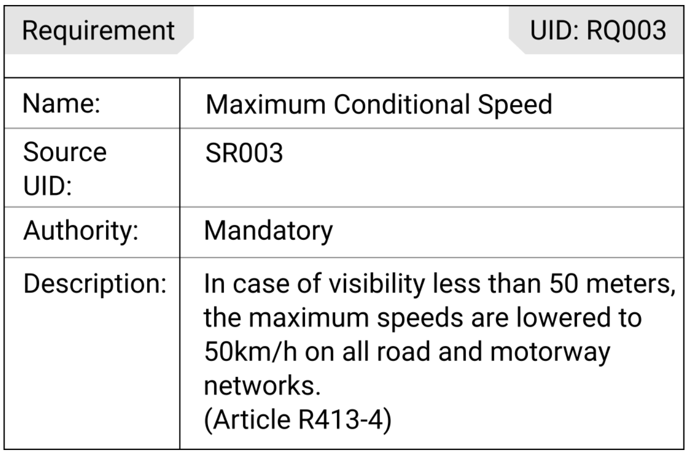
Figure 5.10: Requirements source form.

5.11.4 Requirements View

After identifying the sources of requirements from each element in the organizational view, this view contains all the requirements in the CMMAV. Each requirement has a UID, a description, a UID of the corresponding source from the requirements sources view (Section 5.11.3), and the requirement's level of importance (mandatory or recommended). The last attribute defines whether this requirement must be respected (e.g. state laws) by a user stakeholder, or it is up to them to respect it or not (using incentive to incite cooperation for example). Fig. 5.11 is an example of a requirement that comes from traffic laws that constrain the speeds of CS, it is mandatory for CS that use state's infrastructures.

5.11.5 Functionalities View

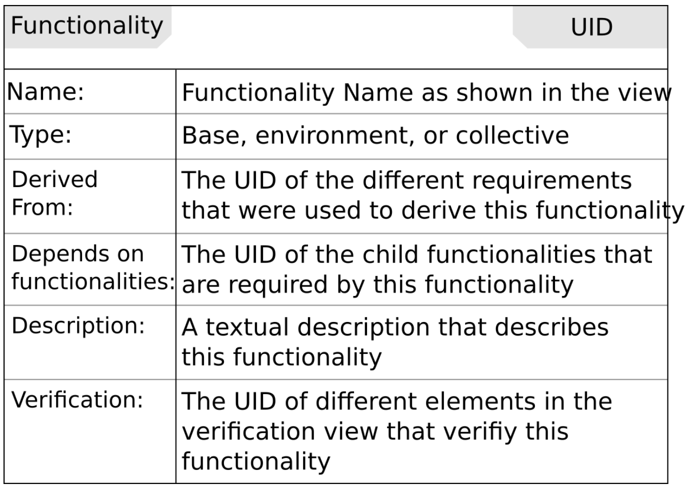
The functionalities view consists of a set of elements that describe the logical architecture proposed by the CMMAV to its users. This architecture consists of a set of recommended and required functionalities that need to exist in CS in order for it to cooperate with other CS. For Users interested only in using the CMMAV and not in its development, it is sufficient to look at both this view, and Verification View (Section 5.11.6) in order to develop their applications. The full functionalities view is in the Appendix (Fig. A.4). Base and environment functionalities are required to ensure that the CS is capable of performing the basic tasks, they are expected to exist on all CS before adding CMMAV based functionalities, they show in this view



Requirement	UID: RQ003
Name:	Maximum Conditional Speed
Source UID:	SR003
Authority:	Mandatory
Description:	In case of visibility less than 50 meters, the maximum speeds are lowered to 50km/h on all road and motorway networks. (Article R413-4)

Figure 5.11: Requirement form.

because they are the requirements of CMMAV maintainers. On the other hand, CS are not expected to have collective functionalities, or have them but not to the CMMAV standards, they are necessary to ensure cooperation between CS. Each element in this view (functionality) is described in detail in a separate form (Fig. 5.12). Figure 5.13 is an example of the “Neighbor CS identification” functionality from collective functionalities. The full list of descriptions of all functionalities could be found in the Appendix (Section A.3.4).

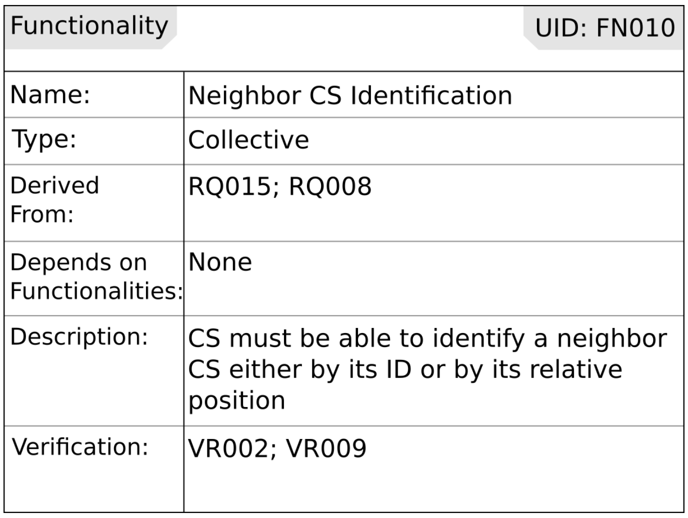


Functionality	UID
Name:	Functionality Name as shown in the view
Type:	Base, environment, or collective
Derived From:	The UID of the different requirements that were used to derive this functionality
Depends on functionalities:	The UID of the child functionalities that are required by this functionality
Description:	A textual description that describes this functionality
Verification:	The UID of different elements in the verification view that verifiy this functionality

Figure 5.12: Functionality Form: several attributes to describe a functionality.

5.11.6 Verification View

The verification view describes the necessary set of verification tests, or indicators that, when validated by a user in its CMMAV based application, the application is safe to be used. User stakeholders provide elements as well through MoE included in the submitted use case. Regulations of such applications are expected to arise in the future, and they could be added when the CMMAV is revised to refine it. For Users interested only in using the CMMAV and not in its development, it is sufficient to look at both this view, and Functionalities View (Section 5.11.5) in order to develop their applications. Each element (or set of elements) relates to a functionality, or a requirement. There are two types of verification that exist in



Functionality UID: FN010

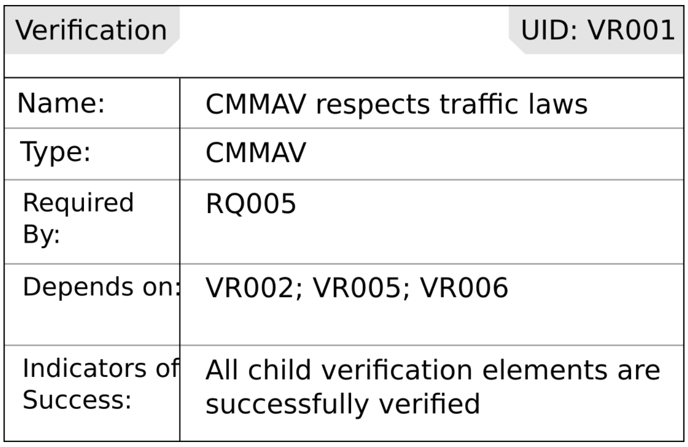
Name:	Neighbor CS Identification
Type:	Collective
Derived From:	RQ015; RQ008
Depends on Functionalities:	None
Description:	CS must be able to identify a neighbor CS either by its ID or by its relative position
Verification:	VR002; VR009

Figure 5.13: Functionality example: Neighbor CS identification.

the CMMAV: CMMAV verification and functionalities verification. The complete verification view is in the Appendix A.5.

5.11.6.1 CMMAV Verification

These are the necessary verification elements that are related to the CMMAV, they exist to ensure that CMMAV related requirements are respected. For example, verifying that a certain functionality or requirement exists in the system. Fig. 5.14 shows the verification element that validates the respect of law requirements. It validates the requirement that comes from the "overtaking on highways" use case about respecting French traffic laws.



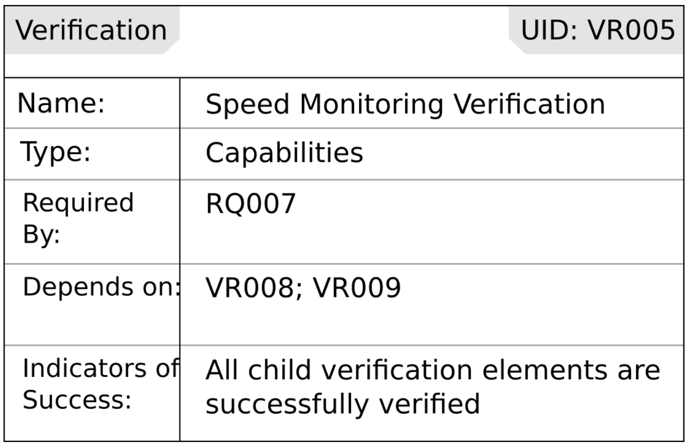
Verification UID: VR001

Name:	CMMAV respects traffic laws
Type:	CMMAV
Required By:	RQ005
Depends on:	VR002; VR005; VR006
Indicators of Success:	All child verification elements are successfully verified

Figure 5.14: "CMMAV respects traffic laws" verification element.

5.11.6.2 Functionalities Verification

These are the verification elements that are used by users to validate that a certain functionality they developed respects the CMMAV. For example, "Speed monitoring Verification" (Fig. 5.15) describes the expected outcome and behavior from the "Speed monitoring" functionality. By looking at this element, users are able to identify whether what they develop respects the CMMAV or not.



Verification	UID: VR005
Name:	Speed Monitoring Verification
Type:	Capabilities
Required By:	RQ007
Depends on:	VR008; VR009
Indicators of Success:	All child verification elements are successfully verified

Figure 5.15: Speed Monitoring Verification element.

5.11.7 Traceability View

Every view in the framework is related to one or more views, use cases view provides the organizational view with actors, and sources view provides requirements views with requirements. To users that only develop applications based on the CMMAV, these relations are most likely irrelevant, as they are only interested in functionalities and verification views. But to stakeholders that are interested in the development of the CMMAV (e.g. maintainers), it is very important to understand the rationale behind the elements it contains. It is important to know which source is related to which functionality, and which stakeholder provided a certain verification requirement. The only objective of the traceability view is to provide all the relations that exist in the CMMAV. The complete view is not provided in this document due to its size. Each relation is unidirectional, has a type, a source element, and a destination element (Table 5.1). We have identified 4 types of relations in the CMMAV:

Table 5.1: Traceability view relation: a name, source, destination, and description.

Attribute	Description
Type	The type of the relation. Possible types: satisfies, requires, provides, and refines.
Source Element	A source element is the element from which the arrow starts in the view. Each type of relations has a set of permitted elements.
Destination Element	A destination element is the element to which the arrow arrives in the view. Each type of relations has a set of permitted elements.

Satisfies: The source element ensures that the destination element is respected in the CMMAV. Speed monitoring functionality ensures that speed traffic laws are respected, which in turn ensures that the source element from Sources View is respected, and so on (Fig. 5.16).

Requires: The source element requires the respect of the destination element in the CMMAV. Continuing with the previous example, maximum speed on high ways requires the presence of speed monitoring functionality (Fig. 5.16).

Provides: The source element provides the destination element to the CMMAV. This relation exists between organizational elements and sources elements, and between sources elements and requirements. For example, speed traffic laws provide the maximum speed of high ways requirements (Fig. 5.16).

Refines: The destination element is refined and then used as a requirement. For example, RQ005 states that drivers should advertise their intentions, but does not specify how. RQ015 which states that CS must advertise their intentions to neighbors using communication and light signals refines RQ005 to be suitable to its source use case.

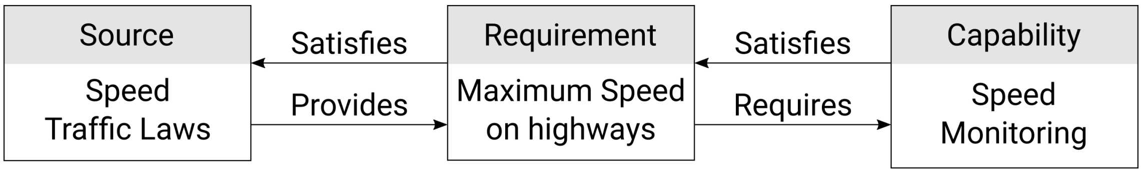


Figure 5.16: Validates, Requires, and Provides relations.

Any element might have several types of relations with different elements of the CMMAV at the same time.

5.12 CMMAV Horizons

The CMMAV is designed and built to cater for different users building different collective functionalities, while at the same time being able to inter-cooperate. Since different actors in transportation systems are connected via use cases, it is expected that each use case will be related to another one via its stakeholders. For example, an AV-cloud use case that consists of vehicles using cloud services is related to overtaking use case by having AV in common as CS. This relation means that these use cases might benefit from each other in the incentive mechanisms, for example: points gained from providing services in the overtaking use case could be spent in the cloud requesting services. This is a new use case that might be added to the CMMAV, but requires, however, eliciting new requirements from both stakeholders. Furthermore, the framework contains different views and relations. Stakeholders who wish to look at the framework might be interested in one type of relation, or elements related to a specific stakeholder. To make full use of the CMMAV potential, we propose the development of a tool that consists of 2 parts: a backlog that contains the data of the different sources and views, and a user interface that provides each user with the tools to look at and interact with the CMMAV. The backlog contains use cases, sources, and views that form the CMMAV. It is populated by the CMMAV maintainers and partly users who submit use cases. While the user interface is used by users to view the CMMAV on different levels, or based on different filters: elements that are related to provided relation, elements that trace to a specific stakeholder, requirements under reviewing, etc.

5.13 Conclusion

The development of collective functionalities in transportation systems is one of the main benefits of achieving commercial autonomous driving. Even though they are not required at the start, it is important to think about them beforehand, to build

systems in a way that support cooperation between different CS. This chapter introduced the CMMAV, a framework conceived to guide the development of such functionalities in different transportation systems. To account for the dynamic environment of CMMAV, and the diversity of stakeholders and CS, CMMAV process is based on use cases that define the stakeholders and CS of a specific use case, and a SoS approach to map the requirements of different stakeholders into a set of functionalities and verification requirements that may be used by any user who wishes to develop CMMAV based functionalities in their systems. To represent the system, a seven view framework was derived from SoS-ACRE: use case view, organizational view, sources view, requirements view, functionalities view, verification view, and traceability view. Each view contains a high-level representation that uses a hierarchical tree to represent the different relations within this view, and a detailed view via forms that describe each element in this view, and is used to generate the relations that exist between different views in the traceability view.
Besides developing CMMAV based functionalities, several potential uses for the CMMAV may arise: it could be considered by a specific standardization organization to produce a standard related to cooperation between vehicles for example. In this case, it is up to its maintainers to ensure that the requirements of this organization are met, and to include this organization into its user stakeholders. Another use case comes from users who might be interested in adopting the CMMAV process to manage projects that have diverse independent stakeholders that need to cooperate in order to achieve their goals.
The first use case that was adopted to build the actual draft of the CMMAV is also considered to validate it, and that is what we are going to discuss in the next chapter: The Cooperative Lateral Maneuvers Manager (CLMM), which is the first application that uses the CMMAV framework in its development.

Chapter 6

Application: Cooperative Lateral Maneuvers Manager (CLMM)

6.1 Introduction

Overtaking, lane changing, platooning and merging are all different types of maneuvers constantly performed by vehicles on roads. These maneuvers are very complex in that they require the driver to perform multiple tasks in a short period of time, from the perception of the surrounding vehicles, to the estimation of speeds and distances, etc. The main benefit of driving automation is to perform these tasks in a safer manner, and to eliminate human errors from the process of driving and decision-making. Cooperation using Vehicle-to-X (V2X) communications was proven to be useful in the literature (Guériau et al. 2016), as well as in live demonstrations (Englund et al. 2016). Acquiring an accurate knowledge about the states and intentions of the surrounding vehicles enables a better decision-making process about the next action for a vehicle.
Studies concerning lateral maneuvers such as overtaking and lane changing treat the topic from different points of view. From the modeling aspect, Zheng (Zheng 2014) reviews studies about lane-changing models, and distinguishes two groups of models: models that capture the lane-changing decision making process, and others that study the influence of lane-changing on nearby traffic flow. Always in the modeling aspect, Hidas (Hidas 2002) introduces SITRAS, a multi-agent simulation system that could be used to evaluate ITS applications, and studied models treating lane-changing and merging. Petrov et al. (Petrov and Nashashibi 2014) modeled autonomous vehicle overtaking, and proposed a nonlinear control scheme that uses only the relative position and orientation with respect to the overtaken vehicle acquired from onboard sensors. Other studies such as (Satzoda and Trivedi 2014) contributed to the subject by proposing a vehicle detection assistance system, that uses appearances to detect overtaking and receding vehicles. Concerning the actual maneuvers themselves, different control laws were proposed to govern the maneuvers in autonomous vehicles [(Murgovski and Sjöberg 2015), (Nguyen et al. 2017)]. The former studies deal with the maneuvers from the subject vehicle's perspective, and do not treat the cooperative aspect. Vehicle to X communications (vehicle to infrastructure, vehicle to vehicle) opened new potential to automated driving applications, and notably cooperative applications such as platooning and cooperative merging applications, which benefit from the increased accuracy of information ac-

quired by vehicles. Luo et al. (Luo et al. 2016) proposed a trajectory planning method that uses vehicle-to-vehicle (V2V) communication to plan a reference maneuver trajectory that avoid collisions, while Nie et al. (Nie et al. 2016) proposed a decentralized approach to the lane-changing maneuver, where V2V communication is used to make decisions about lane-changing in autonomous vehicles. While the last two studies use V2V communication for planning trajectories, they do not consider the cooperation aspect and how to exchange services between communicating vehicles.
In the previous chapter, we introduced the CMMAV, a SoS framework that aims to enable cooperation between autonomous vehicles (chapter 5). The purpose of the CMMAV is to be used to build applications on autonomous vehicles that aim to enable cooperation in certain situations between multiple vehicles, and the first use-case that was used to populate the framework was the overtaking maneuver on highways. This chapter introduces the first application that was built using the CMMAV framework that caters for the overtaking maneuvers. The Cooperative Lateral Maneuvers Manager (CLMM) is the application developed for this purpose, and is the subject of this chapter. The objective of the CLMM is to allow equipped vehicles to perform actions for one another in order to ease the execution of overtaking maneuvers. CLMM does not tell the vehicle how to perform the overtaking maneuver, but rather it tells it when to do it, and if this maneuver could not be performed due to conflict with another CLMM equipped vehicle, it enables the cooperation between the 2 vehicles in order to enable that maneuver.

The objectives of this chapter are:

1. Explain the procedure and the strategy we use to perform overtaking maneuvers;
2. Describe the adopted architecture of the CLMM, and the developed application in the autonomous vehicles;
3. Provide the different validations performed to validate the approach.

6.2 CLMM Objectives

Similar to mobile phone applications that exchange messages and services (e.g. automatic reservation applications), the vehicles equipped with the CLMM, i.e. CMMAV vehicles, exchange messages in a request/response mechanism in order to cooperate during overtaking maneuvers. The CLMM gathers information from the subject vehicle about its environment, messages broadcasted by its neighbors, and information related to the subject vehicle itself (such as speed and position). Whenever the initial conditions are met, the CLMM takes control over the decision making of the subject vehicle in order to perform the overtaking. In the cases where the presence of neighbor vehicles obstruct the execution of the maneuver, for example, performing the maneuver entails a collision risk with the top-front neighbor, in such cases, and when these neighbors are also CMMAV vehicles, the CLMM establishes connection with that specific neighbor, requesting it to perform a certain action that enables the subject vehicle to execute the maneuver with less waiting time. If the request is accepted by the neighbor vehicle, its CLMM tells the subject vehicle's decision making center the action it must perform for the subject vehicle. After that, the subject vehicle may overtake its front neighbor, and eventually continues its trajectory. Furthermore, on SoS design aspect, the CLMM is based on the CMMAV, which is a SoS framework. An important assumption that we make implicitly in the CMMAV is the assumption that regardless of the methods used by the different CS to perform the overtaking maneuvers, and that with the different technologies they use, we are still able to achieve cooperation and a safe operating SoS just by applying high level control over the CS. In summary, the objectives of the CLMM are:

1. Enable vehicles to perform overtaking maneuvers on highways in cooperative manner;
2. Validate the CMMAV, and in consequence that the method used to develop it is valid.

6.2.1 CLMM Evaluation

To evaluate the CLMM, we have to make sure that its objectives are attained. In other words, does the performance of the CLMM meet our expectations of the system? The CLMM has two objectives, and therefore two evaluations must be performed.

6.2.1.1 Evaluating the Performance

This first evaluation step is necessary in order to guarantee that the CLMM does what it is supposed to do, which validates the first objective. This evaluation is done using the three validation techniques presented in Section 6.5, where we test it and we observe the behavior of the system. The first criterion we use to evaluate the CLMM is the "no collision" criterion, which is a trivial requirement that states that there must be no collisions between vehicles during the operation of the CLMM (i.e. during overtaking maneuvers).
The objective of the CLMM is to enable cooperation between vehicles under the assumption that this cooperation will lead to less overall maneuver time from the moment a vehicle decides to perform the maneuver until it is finished. In consequence of this cooperation, the vehicles that perform actions for the subject vehicle are changing their trajectory speeds for the period of the maneuver. It is interesting to compare the overall time gained by the vehicles performing the maneuvers, with the overall time gained or lost by the vehicles performing the actions, to evaluate the behavior on the SoS level. Also it is interesting to compare the same values for each vehicle individually, to study that effect on the CS level. However, this evaluation requires long period studies to capture the effect on a meaningful scale, and could be performed in simulation.

6.2.1.2 Evaluating the CMMAV

As we will see later, the analysis that we present when we present our strategy for cooperation (Section 6.3.5), and when we validate the CLMM using formal scenarios (Section 6.5.1), the knowledge about the way the CMMAV vehicles perform certain maneuvers will not show up, whereas relative speeds, distances, and intentions will show up. This is an indicator that when we assume that CS know how to do what they are supposed to do. For example, CS know how to change the lane when the decision is made, we can achieve cooperation only by making decisions, rather than

also providing operational constraints on the CS (in the case where we define the way CS must change lanes during cooperation for example), which validates our claim in Section 6.2.
Furthermore, if we develop the functionalities required by the CMMAV, and perform the necessary verifications, we can observe if all the requirements of stakeholders are considered in the CMMAV, and that there are no requirements left unfulfilled. For example, if the maneuvers performed by the CLMM are uncomfortable to the passengers, this means that the comfort requirement of the passengers is either not well implemented in the CLMM, or is not mentioned in the CMMAV.

6.3 Overtaking On Highways

This section introduces our view on overtaking maneuvers addressed in the CLMM, and the strategy adopted to handle such maneuvers, and introduces the cooperation aspect into the process. More precisely, the overtaking maneuvers addressed throughout this chapter are the maneuvers executed on 2-lane or more, one-way highways. Fig. 6.1 represents the environment in which the overtaking maneuvers addressed herein take place, the arrows on the far right represent the direction of the traffic flow on the different lanes which in our case are the same on all lanes. The red vehicle is the subject vehicle, which is the vehicle that intends to overtake its front neighbor, the teal vehicle.

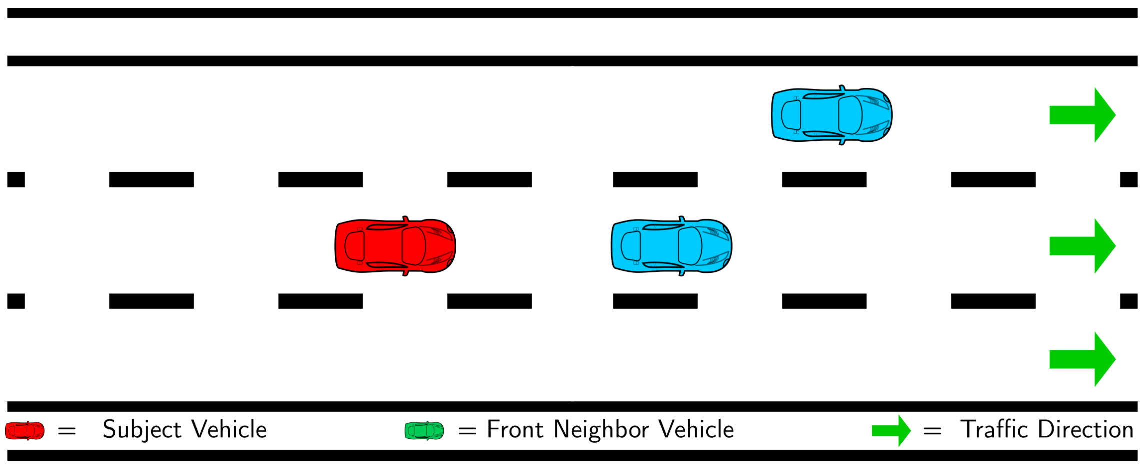


Figure 6.1: Overtaking configurations addressed in the CLMM: two or more lanes and same traffic flow direction.

6.3.1 Subject Vehicle, CMMAV Vehicle, and other terms

Throughout this chapter, we will mention the term vehicles a lot to refer to the different vehicles involved in an overtaking maneuver. For this reason, different names are going to be used to refer to different types of vehicles, and below the list of different names that we are going to use:

Subject Vehicle: The vehicle performing the overtaking;

Neighbor Vehicle: Any vehicle surrounding the subject vehicle directly; i.e. the front neighbor or any side neighbor;

Adjacent Vehicle: Is any vehicle situated at the first right or first left lane of the subject vehicle;

CMMAV Vehicle: Is any vehicle equipped with the CLMM and able to exchange messages with neighbors using the CLMM format; A CMMAV vehicle could be the subject vehicle, or any neighbor; All CMMAV vehicles are communicating vehicles;

Communicating Vehicle: Is any vehicle able to communicate with other vehicles using wireless communication;

Non-Communicating Vehicle: Is any vehicle that is not able to communicate with other vehicles; No non-communicating vehicle may be a CMMAV vehicle;

Human-Driven Vehicle: Is any vehicle driven by a human driver, it could be communicating or non-communicating vehicle;

Autonomous Vehicle: Is any vehicle that uses software to drive itself to move from position A to position B; All autonomous vehicles are communicating vehicles;

6.3.2 Distances

Figure 6.2 represents two distances that we use in the CLMM: the communication radius, and the security distance. Here we explain what they represent.

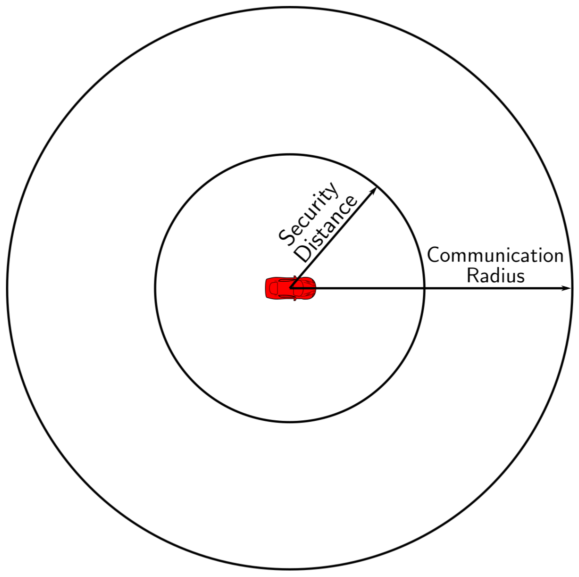


Figure 6.2: Two Measures of Interest: security distance, and communication radius.

6.3.2.1 Communication Radius

It is the maximum distance between two vehicles, beyond which, vehicles are not able to communicate anymore. Any neighbor vehicle identified by the subject vehicle is necessarily at a distance smaller than the communication radius from the subject vehicle. The communication radius in the CLMM is 800 meters in open areas, and is equal to the range of wireless network coverage of the antenna used inside the vehicle, a Cohda modem that serves as an UDP IPv4 to 802.11p gateway for sending and receiving messages (Xu et al. 2018).

6.3.2.2 Security Distance

It is the minimum permitted distance between the subject vehicle and its neighbor vehicle situated in the front, and on the same lane. Unlike communication radius which is defined by the range of the wireless antenna used by the vehicle, the security distance is variable and depends on the actual speed of the subject vehicle. At any moment, it is the responsibility of the subject vehicle to keep a distance with its front neighbor that is greater or equal to the security distance. Currently we use the formula in equation 6.1, to compute the desired value of the security distance, which is a requirement that comes from traffic laws (requirement RQ004 in A.3.3), which states that the minimal distance between two consecutive vehicles must equal the distance traveled by the subject vehicle in 2 seconds.

$$d_s = 2 * V_{sv} \tag{6.1}$$

Where d_s is the security distance, V_{sv} is the subject vehicle's speed, and 2 is the reaction time factor. Note that this law was made for human-driven vehicles and accounts for the reaction time of a human driver (McGehee, Mazzae, and G. S. Baldwin 2000). In the future, different laws may come out for autonomous vehicles, and therefore the formula used to compute the security distance may change as well.

6.3.3 Relative Positioning

At any given moment on a highway, let the “neighborhood degree” be the number of vehicles that separate 2 vehicles assumed on the same lane, so for example in Fig. 6.3, the neighborhood degree of the green vehicle with respect to the red vehicle is 0, that is, if we assume that the red and the green vehicles share the same lane, there are 0 vehicles that separate them. The neighborhood degree of the blue vehicle with respect to the red vehicle is 1, and the orange's is 2. In the CLMM, we only consider vehicles that have 0 neighborhood degree with the subject vehicle (see Section 5.10). Therefore, if we combine this with the previously mentioned distances (Section 6.3.2), we could obtain what we call a relative map that shows the possible different neighbors for a 3-lane highway (Fig. 6.4). In this map, there are 8 possible relative positions that might be occupied by a neighbor vehicle, represented by the blue vehicles. Figure 6.5 is a grid that represents the labels used to label neighbors, with **F**, **R**, **T**, and **B** representing respectively the front, rear, top, and bottom directions with respect to the subject vehicle. **TF**, **TR**, **BF**, and **BR** represent respectively the top-front, top-rear, bottom-front, and bottom-rear neighbors. A neighbor is considered top-front neighbor if it is on the left lane of the subject vehicle, in front of it, and on a longitudinal distance bigger than the security distance, while the top neighbor is the neighbor to the left of the subject vehicle, but on a longitudinal distance smaller than the security distance. Since we consider only neighbor vehicles with a neighborhood degree of 0 with respect to the subject vehicle, on a 4 or more lanes, as far as the CLMM is concerned, the subject vehicle may have at most 8 neighbors, which are represented in Fig. 6.5. Whereas on 2-lane highway, this map may be reduced to match that specific case. That is why in what follows, all our work uses figure 6.5 when referring to neighbors.

6.3.4 Strategy Overview

The whole overtaking maneuver is represented using a flowchart in Fig. 6.6. The maneuver starts with the subject vehicle cruising on a trajectory lane with a trajec-

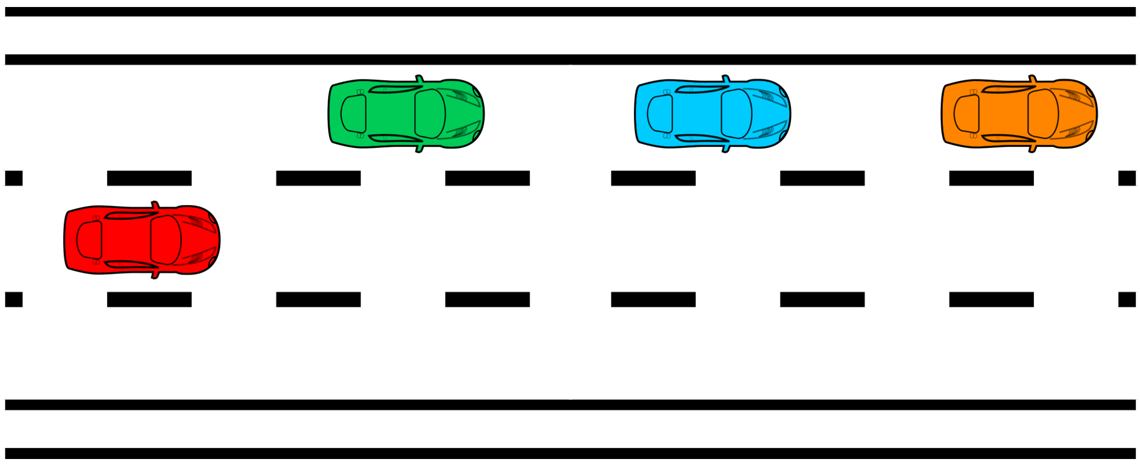


Figure 6.3: Neighborhood Degree: a measure that reflects the degree of proximity of different vehicles.

tory speed. Whenever the initial condition and the first safety conditions are met, it then proceeds to the first lane change, where it waits until the second safety conditions are met. When that happens, the subject vehicle may go back to its trajectory lane by performing a second lane change. The decision to perform an overtaking is made when the subject vehicle, which is moving at a speed which we call "trajectory speed" denoted by V_{sv}, encounters a front neighbor which is moving at a speed V_{fn}, where $V_{fn} < V_{sv}$, and that is not performing any overtaking maneuver itself. When this enabling condition is met, the subject vehicle intends to overtake its front neighbor. In principle, the subject vehicle could overtake its front neighbor either on its right or its left (that is changing lane to the left or right lane) if these lanes exist (i.e. the subject vehicle is on neither the far right nor far left lane), but since we adopt the french traffic laws, requirement RQ010 (Fig. A.8, Appendix) tells us that the subject vehicle is only allowed to overtake on the left side of the front neighbor (with one exception in requirement RQ011 in Fig. A.8, Appendix). Therefore, before making the overtaking decision, the subject vehicle must make sure that it is safe to move to the left lane, and this is done by verifying that there are no risk of collision with any neighbor that occupies its left lane, and that there is a left lane to change to it. More precisely, if we consider the three possible neighbors positions from Fig. 6.5, **TR**, **T**, and **TF**, the first safety conditions that must be met are:

First Safety Conditions:

1. There is no vehicle in the **T** position;
2. There is no vehicle in the **TF** position, OR there is a vehicle there but the time-to-collision (TTC) between it and the subject vehicle is bigger than a certain threshold;
3. There is no vehicle in the **TR** position, OR there is a vehicle there but the TTC between it and the subject vehicle is bigger than a certain threshold;

When these three conditions are met, the subject vehicle may start the overtaking maneuver, where it performs the first lane change maneuver. According to requirement RQ014 (Fig. A.8, Appendix), there should be a speed difference sufficient

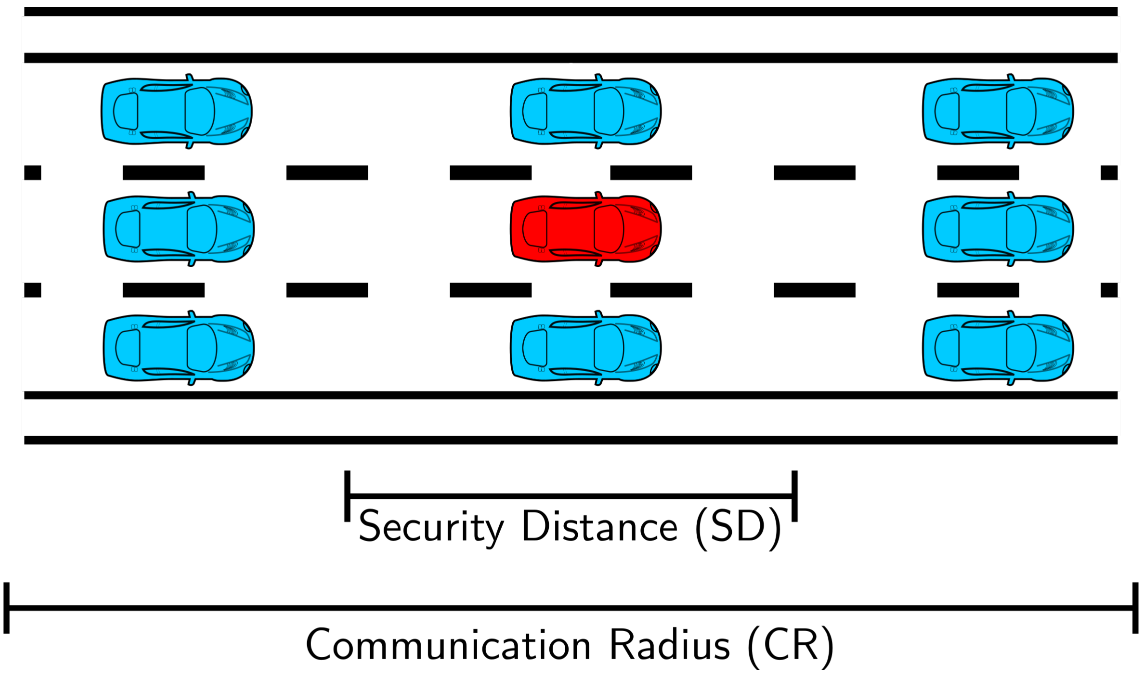


Figure 6.4: Neighborhood map: the different possible existing neighbors on a 3-lane highway.

enough so that the overtaking maneuver takes time in a relatively short time. We chose this speed difference to be at least $20km/h$ in this version of the CLMM. When the subject vehicle finishes the first lane changing maneuver, there is a period of time where it is cruising beside the overtaken vehicle. At the end of this cruising period, it must perform another lane changing, but this time to the right lane, to go back to the initial lane. This second lane changing maneuver is considered as the same as the first one, but in the opposite direction, however this time, the neighbors of interest are the bottom neighbors. Therefore, the second safety conditions that must be met in order to perform the second lane changing are:
Second Safety Conditions:

1. There is no vehicle in the **B** position;
2. There is no vehicle in the **BF** position, OR there is a vehicle there but the TTC between it and the subject vehicle is bigger than a certain threshold;
3. There is no vehicle in the **BR** position, OR there is a vehicle there but the TTC between it and the subject vehicle is bigger than a certain threshold;

Whenever these conditions are met, the subject vehicle may perform the second lane changing to go back to its initial lane, where it also goes back to its trajectory speed as well.

6.3.4.1 Time-To-Collision (TTC)

The time-to-collision (TTC) is an important measure that the majority of safety conditions depend on (equation 6.2). It is defined as the time required for two vehicles to collide if they continue at their present speed and along the same path (Hayward 1971). In (Laureshyn, Svensson, and Hydén 2010), several methods for computing TTC between two vehicles were presented, depending on the type of the

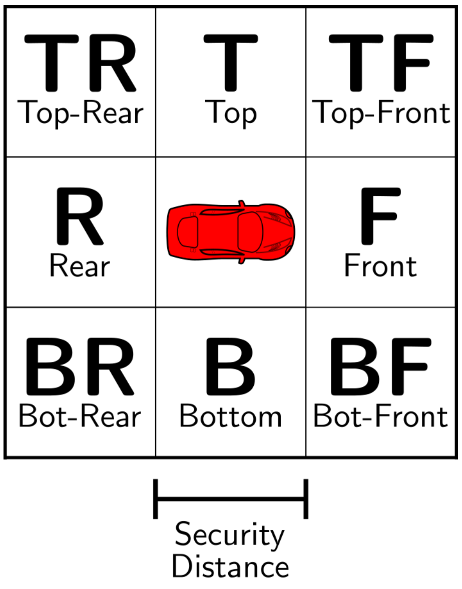


Figure 6.5: Neighborhood relative map: the labels used to identify different neighbors.

anticipated collision (right angle collision, head-on collision, etc.). In our case, the collision (if happened) would be a rear-end collision, meaning that a vehicle will collide to the rear-end of another moving vehicle, where both vehicles are moving in the same direction. In this case, TTC is calculated as

$$TTC = \frac{d_{12}}{v_2 - v_1}, \qquad \text{if} \quad v_2 > v_1 \tag{6.2}$$

Where $v_2 - v_1$ represents the speed difference between the two vehicles, with v_1 and v_2 representing the speeds of the front and rear vehicles respectively, and d_{12} represents the distance between the front-end of the rear vehicle, and the rear-end of the front vehicle (Fig. 6.7). The first thing to notice about the TTC is that it is unnecessary to consider it if $v_1 \geq v_2$, since in this case, the two vehicles will never meet. However, when $v_1 < v_2$, TTC will be a positive value that represents the remaining time until both vehicles collide, that is, if they are moving in the same lane. All this section discusses TTC under the assumption that the two vehicles are moving at a constant speed, which is a reasonable assumption to make when we consider vehicles moving on highways. The TTC is considered in the CLMM as we will see later on to check if it is safe to perform lane changing maneuvers.

6.3.4.2 Double Overtaking Maneuvers, and More

Highways are a dynamic environment, therefore the subject vehicle's neighbors are changing constantly. Sometimes during an overtaking maneuver, on 3-lane or more highways, when the subject vehicle performs the first lane change, it encounters a new front neighbor on the new lane which is moving at a lower speed. In this case, the subject vehicle must overtake this new front neighbor in order to complete the first overtaking, and this is what we call a double overtaking maneuver. This process could repeat itself multiple times depending on the number of lanes, so the way we handle this type of overtaking maneuvers is the following:
During the second cruising process in the single overtaking flowchart, the subject

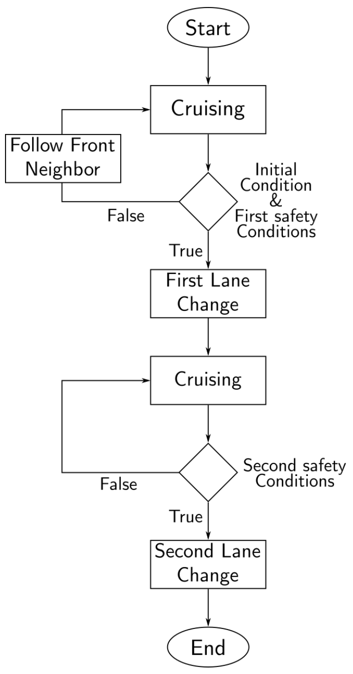


Figure 6.6: A single overtaking maneuver flowchart.

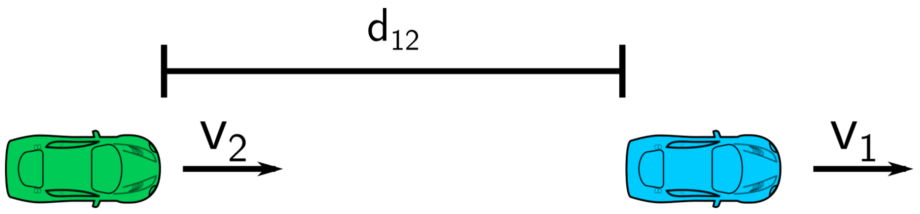


Figure 6.7: Rear-end collision.

vehicle checks constantly for the initial condition. When this condition is met, it repeats the single overtaking process until it overtakes the new front neighbor and gets back to its lane, only then it could resume performing the first overtaking maneuver and returning to its trajectory lane. In this case, the flowchart of a double overtaking maneuver becomes as follows (Fig. 6.8), where the **H** symbol represents a history element that stores the number of overtaking maneuvers performed alongside their different parameters. This way, several overtaking maneuvers may be performed in a recursive manner.

6.3.5 Cooperation in Overtaking

As we have discussed in the overtaking strategy Section 6.3.4, a typical overtaking maneuver may involve multiple neighbors. Neighbors appear in the different conditions that are examined before transitioning from a process to another one in the overtaking maneuver flowchart, and therefore they play an important role in the maneuver's execution time and even its execution in general. For example, the front neighbor shows in the initial condition, whereas the top neighbor shows in the first safety conditions. In order to achieve cooperation between the different vehicles, let us explore what are the possible actions neighbors may perform in order to help the subject vehicle performing the overtaking:

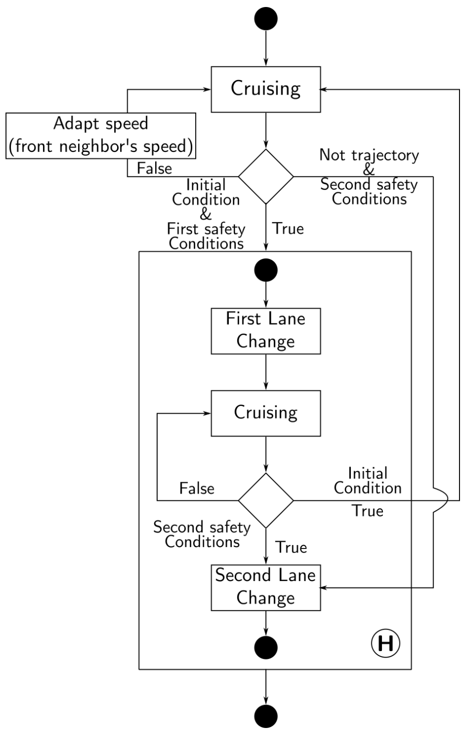

Cruising
Adapt speed (front neighbor's speed)
False
Initial Condition & First safety Conditions
True
Not trajectory & Second safety Conditions
First Lane Change
Cruising
False
Initial Condition
True
Second safety Conditions
True
Second Lane Change
H

Figure 6.8: Multiple overtaking maneuver flowchart.

6.3.5.1 Subject Vehicle - Front Neighbor

The decision to overtake is taken when the subject vehicle encounters a front neighbor moving at a less speed than the subject vehicle (Section 6.3.4). This tells us that in order to avoid overtaking, the front neighbor has to change its lane, or speed up to match the speed of the subject vehicle. Let A_i^n be the action i that the neighbor vehicle with label n may perform for the subject vehicle, therefore the two possible actions the neighbor **F** may perform to the subject vehicle are:

$A_1^{\mathbf{F}}$: speed up to match the speed of the subject vehicle during the whole shared trajectory;

$A_2^{\mathbf{F}}$: change lane until the subject vehicle is in front;

$A_3^{\mathbf{F}}$: do nothing;

6.3.5.2 Subject Vehicle - Rear Neighbor

The rear neighbor does not show in any condition throughout the overtaking maneuver flowchart, and that is because the rear neighbor has no relation with the subject vehicle during the overtaking maneuver. The rear neighbor is responsible of keeping a distance higher than the security distance mentioned in Section 6.3.2.2 if it does not intend to overtake the subject vehicle. As long as the rear neighbor respects this distance, the subject vehicle may overtake its front neighbor. However, in the case where the rear vehicle intends to overtake the subject vehicle, it becomes the subject vehicle, and the subject vehicle becomes its front neighbor, which takes us back to the first relation (subject vehicle-front neighbor relation in Section 6.3.5.1).

6.3.5.3 Subject Vehicle - Top-Front Neighbor

The top-front neighbor shows in the first safety conditions. To allow the subject vehicle to perform the first lane change in the overtaking maneuver, the subject vehicle must have a TTC with its top-front neighbor which is bigger than the time necessary for the overtaking maneuver to be fully executed $T_{overtaking}$. Using the TTC formula provided in Section 6.3.4.1, we get the following relation (equation 6.3):

$$\frac{d_{sn}}{V_s - V_n} \geq T_{overtaking} \tag{6.3}$$

This equation gives us the minimum speed difference between the subject vehicle and its top-front neighbor required in order to meet the safety condition. At any given moment, for each distance d_{sn} between the subject vehicle and its top-front neighbor, the speed difference between the two vehicles must be higher or equal to a speed difference threshold. In other words, for any given distance, if the subject vehicle's speed is higher than that of its top-front by a certain threshold that depends on that given distance, the subject vehicle will collide with its top-front neighbor. So to avoid this, the subject vehicle's speed must be less or equal than that of its top-front neighbor. That being said, the possible actions the top-front neighbor might perform for the subject vehicle in order to meet the conditions and perform the overtaking are:

$A_1^{\mathbf{TF}}$**:** change lane until the subject vehicle is finished from its overtaking;

$A_2^{\mathbf{TF}}$**:** speed up until the speed difference between both vehicles is higher or equal to the speed difference threshold;

$A_3^{\mathbf{TF}}$**:** do nothing;

6.3.5.4 Subject Vehicle - Top Neighbor

The top neighbor is by definition the vehicle that occupies the left lane of the subject vehicle, and is inside the security distance discussed in Section 6.3.2.2. It shows in the first safety conditions set, and is required to not exist so that the subject vehicle may perform the overtaking maneuver. This requirement is made because without cooperation, the subject vehicle must wait until its top neighbor passes it, or it passes it. However, when we introduce cooperation, the subject vehicle is able to ask its top neighbor to either speed up and passes it, or slow down and let it pass. So the actions available to the top neighbor to perform are:

$A_1^{\mathbf{T}}$**:** change lane to allow the subject vehicle to overtake, then go back to lane;

$A_2^{\mathbf{T}}$**:** speed up so that its speed is higher than that of the subject vehicle's, until it becomes a top-front neighbor;

$A_3^{\mathbf{T}}$**:** slow down so that its speed is lower than that of the subject vehicle's, until it becomes a top-rear neighbor;

$A_3^{\mathbf{T}}$**:** do nothing;

6.3.5.5 Subject Vehicle - Top-Rear Neighbor

The relation between the subject vehicle and its top-rear neighbor is very similar to that with its top-front neighbor, in that the subject vehicle avoids collision with those neighbors. The difference is that the top-rear neighbor is behind the subject vehicle, which means that instead of having a higher speed than that of the subject vehicle, the top-rear neighbor must have a lower or equal speed. Therefore, the possible actions that may be performed by the top-rear neighbor in order to enable the overtaking maneuver of the subject vehicle are:

$A_1^{\mathbf{TR}}$**:** change lane until the subject vehicle is finished from its overtaking;

$A_2^{\mathbf{TR}}$**:** slow down until the speed difference between both vehicles is higher or equal to the speed difference threshold;

$A_3^{\mathbf{TR}}$**:** do nothing;

6.3.5.6 Subject Vehicle - Bottom-Front Neighbor

Bottom neighbors in general show in the second safety conditions set. Assuming that both lane change maneuvers performed are symmetric, meaning that they are literally the same, but they have opposite directions, we could apply the same analysis of top neighbors to bottom neighbors, since the analysis did not involve the direction of the lane change the subject vehicle intends to perform. Therefore, the possible actions the bottom-front neighbor may perform are the same as the one that the top-front neighbor may perform and they are:

$A_1^{\mathbf{BF}}$**:** change lane until the subject vehicle is finished from its overtaking;

$A_2^{\mathbf{BF}}$**:** speed up so that its speed until the speed difference between both vehicles is higher or equal to the speed difference threshold;

$A_3^{\mathbf{BF}}$**:** do nothing;

6.3.5.7 Subject Vehicle - Bottom Vehicle

Applying the analysis made to the “subject vehicle - top vehicle” relation in Section 6.3.5.5, the possible actions available to the bottom neighbor are:

$A_1^{\mathbf{B}}$**:** change lane to allow the subject vehicle to overtake, then go back to lane;

$A_2^{\mathbf{B}}$**:** speed up so that its speed is higher than that of the subject vehicle’s, until it becomes a bottom-front neighbor;

$A_3^{\mathbf{B}}$**:** slow down so that its speed is lower than that of the subject vehicle’s, until it becomes a bottom-rear neighbor;

$A_3^{\mathbf{B}}$**:** do nothing;

6.3.5.8 Subject Vehicle - Bottom-Rear Vehicle

The same analysis performed on the top-rear neighbor may be performed on the bottom-rear neighbor, since we assumed the symmetry of the lane change maneuvers. Therefore, the possible actions of the bottom-rear neighbor are:

$A_1^{\mathbf{BR}}$**:** change lane until the subject vehicle is finished from its overtaking;

$A_2^{\mathbf{BR}}$**:** slow down until the speed difference between both vehicles is higher or equal to the speed difference threshold;

$A_3^{\mathbf{BR}}$**:** do nothing;

6.3.6 Cooperation Constraints

So far we have listed all the possible actions that may be performed by neighbor vehicles, so that the subject vehicle may perform the desired overtaking maneuver. However, since the subject vehicle and its neighbors up to this point are all considered to be CMMAV vehicles, the actual actions they may perform to one another are constrained by the level of cooperation chosen inside each vehicle. So for example, if the top-front neighbor vehicle's owner had set the cooperation level to non-cooperative for a period of time, the only possible action it may perform for the subject vehicle is doing nothing. Building on this point, the first factor that affects the actions that a CMMAV vehicle might perform to another CMMAV vehicle is the cooperation level chosen by its owner/operator.

6.3.6.1 Cooperation Levels

In the CLMM, there are four cooperation levels available to choose from, denoted CL_i where i is the cooperation level, and $1 \leq i \leq 4$. CL_1 is the non-cooperative level, where the only action a CMMAV vehicle might perform is the "do nothing" action; CL_2 is the second level, which involved only speed actions (speeding up or slowing down) that require a speed change less than $10km/h$, that is, the speed difference between the actual vehicle's speed and the speed difference threshold is less than $10km/h$; CL_3 is a higher cooperation level which involves speed actions (same as the previous level), however, vehicles will accept to change their speeds up to $20km/h$ difference; and lastly CL_4 is the higher cooperation level, where CMMAV vehicles may change their speeds, and/or their lanes for other CMMAV vehicles. Table 6.1 represents each cooperation level alongside the actions involved in each level.

Table 6.1: Different cooperation levels and involved actions.

Cooperation Level:	**Involved Actions**
Level 1	no action
Level 2	speed actions up to $10km/h$ speed difference
Level 3	speed actions up to $20km/h$ speed difference
Level 4	speed actions up to $20km/h$ speed difference, and lane change

6.3.6.2 Chain Reactions

We have mentioned the traffic shock wave behavior in Section 5.10. The reason traffic shock wave happens is that a vehicle changed its speed, which forces the trailing vehicles to change their speed to avoid collision. This leads to chain reaction in the traffic flow, and eventually to the formation of a traffic shock wave. In our case of cooperation, suppose that the top-rear neighbor slows down to allow the subject vehicle to perform the overtaking maneuver, and that this top-rear neighbor has also a rear neighbor. This act of slowing down will force its rear neighbor to slow down to keep the security distance between the 2 vehicles, and if the conditions are right (i.e. the rear neighbor also has a rear neighbor, and so on) then the first action performed by the top-rear neighbor might lead to a traffic shock wave. Therefore, to avoid this, we limit the cooperation between CMMAV vehicles to vehicles that have a neighborhood degree of 0. This means that a CMMAV vehicle is not allowed to ask for cooperation from another CMMAV vehicle in order to perform an action for a third CMMAV vehicle, and is not allowed to cooperate in case this cooperation affects another vehicle.

6.4 CLMM architecture

To describe the architecture CLMM (Fig. 6.9), we are going to use the Robot Operating System (ROS)[1] notations (node, topic, etc.) because we believe that this is a very intuitive and clear way to describe the architecture of such systems. The standard unit of information is a message, which is a structure of data that contains relevant information. For example, a message could contain kinematics data related to the vehicle, or high level states such as the current action a vehicle is performing. Each type of messages is transmitted on what is called topics. Nodes are blocks of code that perform each a specific function. Nodes receive necessary data by subscribing to specific topics, and return the result of the function that they perform by publishing on specific topics as well. With these 3 notions in mind, message, topic, and node, we can describe the architecture of the CLMM, and the logic behind by separating the different functions it performs into nodes.
In summary, each node is responsible for a certain function, which takes messages as inputs by subscribing to topics, and returns messages by publishing them to specific topics as well.
The CLMM consists of 5 nodes, and 6 different message types, which are described in details in the following subsections.

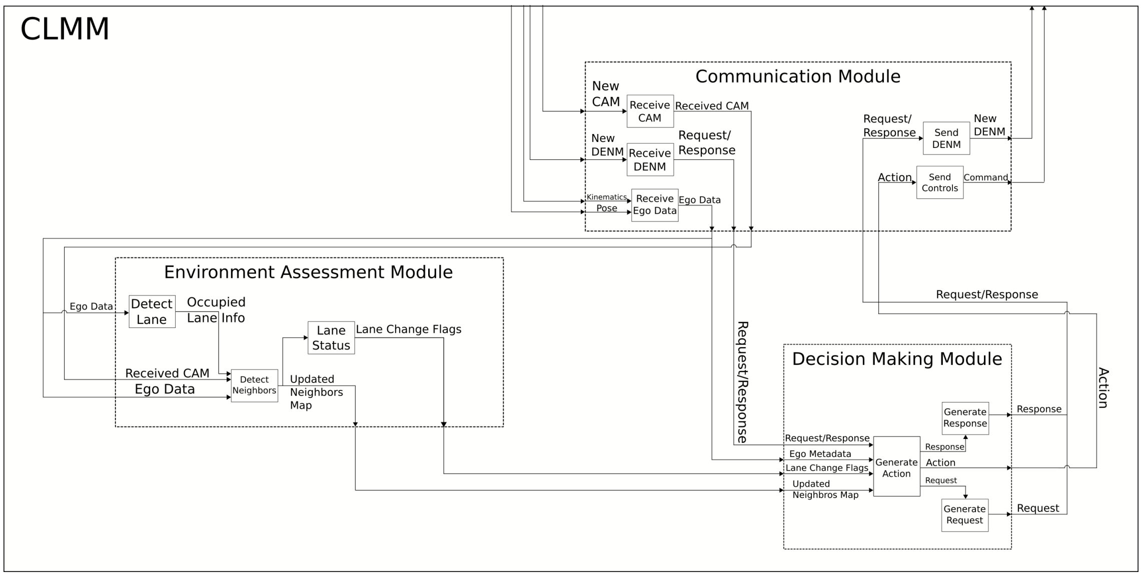


Figure 6.9: CLMM's architecture: information flow between different nodes.

[1]http://www.ros.org/

6.4.1 Nodes

In this section, we explain the different nodes in the CLMM package. For each node, we describe what it does, what it takes as inputs, and what it generates as outputs, and we give an algorithm that describes how this node do what it does.

6.4.1.1 Change Lane Status

This node verifies periodically the possibility for the subject vehicle to change its lane to the right or to the left. It uses the list of neighbors to compute the time-to-collision (TTC) with each neighbor on the adjacent lanes, it also checks the position of the current lane (far left or far right). It then returns for each possible lane change a Boolean flag, which is used later for verifying if a particular lane change is possible or not. Algorithm 2 explains the logic behind generating the `right_lane_status` flag, whereas the `left_lane_status` flag generation follows the same logic, but with neighbors to the left.

Inputs: neighbList, egoSpeed

Outputs: laneChangeStatus

Algorithm 2 Set `right_lane_status` flag

```
1: read egoMetaData
2: read neighbsList
3: if currentLane is far right lane then
4:     return false
5: else if no neighbors to the right then
6:     return true
7: else
8:     for each neighbor to the right do
9:         compute TTC
10:        if TTC ≤ ttcThreshold then
11:            return false
```

6.4.1.2 Lane Discovery

This node provides the CLMM with all necessary information about the occupied lane (see `egoLane` message definition in Section 6.4.2.2): identifier, maximum allowed speed, the lateral displacement from the middle of lane, two values that define its equation in the global frame of reference, and a Boolean that indicates whether this lane is the trajectory lane or not. The input and output messages for this node are:

Inputs: vehicleHighLevelState, egoCoord

Outputs: currentLaneInfo

6.4.1.3 Vehicle Meta Data

The objective of this node is to gather the different information about the subject vehicle state (see `vehicleMetaData` message definition in Section 6.4.2.4): its identifier, current pose, speed, steering wheel angle, the occupied lane information, its length and width, and its high level state (Section 6.4.2.3). This message is periodically broadcasted to all neighbors, and is used in different CLMM modules. The `toDistance` and `timeToCollision` elements of this message are set to -1 in the broadcasted message, and are calculated in the subject vehicle for each neighbor. This node's inputs and outputs are:

Inputs: currentLaneInfo, vehicleHighLevelState, egoCoord, kinematics, wheel angle

Outputs: vehicleMetaData

6.4.1.4 Neighbors Discovery

This node is responsible for assigning each neighbor a relative position with respect to the subject vehicle (Section 6.3.3). It transmits then a list of neighbors to be used in the different nodes of the CLMM. It also computes the TTC for each neighbor, as well as the relative distance. These two values are computed in the local reference frame of the subject vehicle, and consider only the x-axis distance. Inputs and outputs to and from this node are:

Inputs: egoMetaData, neighbMetaData

Outputs: neighbsList

The logic behind this node is explained in the following algorithm (algorithm 3):

Algorithm 3 Generating the neighbors list in the subject vehicle

```
1: read egoMetaData
2: read neighbsList
3: get new neighbor data
4: transform new_neighbor position to local reference
5: compute new_toDistance
6: compute new_TTC
7: detect relative lane of new_neighbor
8: find new_relative_position of new_neighbor in the neighbor map
9: if old_neighbor is in new_relative_position then
10:     compare new_toDistance to old_toDistance
11:     if |new_toDistance| > |old_toDistance| then
12:         replace old_neighbor in neighbsList at position old_relative_position with new_neighbor
13: return neighbsList
```

6.4.1.5 Decision Node

This node is the heart of CLMM. It gathers all the necessary information about the subject vehicle states, and all neighbors' states, and decides on whether or not a specific maneuver should be performed, and the time to perform it. In case cooperation is needed from one or several neighbors, this node generates the request, and handles the response. Moreover, when a neighbor requests cooperation from the subject vehicle, this node will handle this request, and generate the proper response. In addition to that, this node updates regularly the high level state of the vehicle (see `vehicleHighLevelState` message definition in Section 6.4.2.3).

Inputs: egoMetaData, neighbMetaData, laneChangeStatus

Outputs: vehicleHighLevelState, decision

This node works as follows (algorithm 4):

Algorithm 4 Decision Node gathers all necessary information in order to make a decision about cooperation or overtaking.

```
 1: read egoMetaData
 2: read neighbsList
 3: read laneChangeStatus
 4: if current_lane is trajectory_lane then
 5:     if front_neighbor exists and $V_{frontNeighbor} < V_{subjectVehicle}$ then
 6:         if safe_to_change_left then
 7:             Change lane to the left
 8:         else
 9:             change to distance following mode and follow front neighbor
10:             send requests to top_neighbors
11:             while response is not accepted do
12:                 stay in distance following mode
13: else
14:     if safe_to_change_right then
15:         Change lane to the right
16:     else
17:         send requests to top_neighbors
18:         while response is not accepted do
19:             stay in the current_lane
```

6.4.2 Messages

In this section, we describe each message used in the CLMM: the fields that constitute each message, what is the purpose of each field, and the topic name on which this message is transmitted. Every name that ends by ".msg" represents a message type.

6.4.2.1 laneChangeStatus.msg (Table 6.2)

Table 6.2: laneChangeStatus Message Definition.

Message Name:	laneChangeStatus	
Transmitted on:	ChangeLaneStatus	
Field Name:	**Field Type:**	**Field Description:**
`safeToChangeRight`	Boolean	True if a right lane change might be performed safely, False otherwise
`safeToChangeLeft`	Boolean	True if a left lane change might be performed safely, False otherwise

Table 6.3: egoLane Message Definition.

Message Name:	egoLane	
Transmitted on:	currentLaneInfo	
Field Name:	**Field Type:**	**Field Description:**
`laneId`	Integer	Unique lane Identifier
`MaxSpeed`	Double	Current occupied lane's maximum authorized speed
`isTrajectory`	Boolean	True if the current occupied lane is the trajectory lane of the subject vehicle, false otherwise
`lateralError`	Double	The distance between the vehicle and the middle of the occupied lane
`aValue`	Double	The slope of the lane's equation
`bValue`	Double	Second parameter of the lane's equation

6.4.2.2 egoLane.msg (Table 6.3)

6.4.2.3 vehicleHighLevelState.msg (Table 6.4)

6.4.2.4 vehicleMetaData.msg (Table 6.5)

6.4.2.5 neighList.msg (Table 6.6)

6.4.2.6 decision.msg (Table 6.7)

6.5 CLMM Validation

6.5.1 Validation by Formal Scenarios

Formal scenarios are scenarios that are used in the context of validation (Heymans and Dubois 1998) to explore the behavior of a system in certain conditions. In our case, we use those to verify the behavior of the CLMM in the most typical ways to make sure that its architecture and the logic behind it are sufficient, and will lead to the expected behavior, and that we are ready to proceed to other, more in depth validation techniques such as simulation and experimentation. Each scenario starts with a specific configuration, where the subject vehicle is surrounded by certain number of agents, and then uses the logic explained in the CLMM architecture section (Section 6.4.1) to predict the expected behavior of the involved vehicles. In these scenarios, we consider TTC threshold to be $TTC_{thresh} = 12s$.

6.5.1.1 Scenario 1

This scenario explores a simple overtaking case where the subject vehicle requires cooperation from only a single neighbor, which is the top-front neighbor. Figure 6.10 represents the initial configuration of this scenario. The subject vehicle (red vehicle) encounters a front neighbor (green vehicle), and detects a top-front neighbor (orange vehicle) through communication. The speeds of the subject vehicle, its front neighbor, and top-front neighbor respectively are: $V_1 = 70km/h$, $V_2 = 60km/h$, $V_3 = 60km/h$. Since the speed of the subject vehicle is higher than its front neighbor's which is also not performing any overtaking maneuver, then the initial condition applies and the subject vehicle intends to overtake its front neigh-

Table 6.4: Vehicle's high level state Message Definition.

Message Name:	vehicleHighLevelState	
Transmitted on:	HighState	
Field Name:	**Field Type:**	**Field Description:**
`isChangingLane`	Boolean	True if the subject vehicle is in the process of changing its lane, False otherwise
`isOvertaking`	Boolean	True if the subject vehicle is in the process of overtaking, i.e. during both lane changes and cruising. False otherwise
`SpeedFollowing`	Boolean	True if subject vehicle is cruising with no preceding neighbor, False when subject vehicle is following a neighbor
`DistanceFollowing`	Boolean	True if subject vehicle is keeping a fixed distance from a neighbor, False otherwise
`DepartureLane`	Integer	In case of lane changing, this variable contains the ID of the lane at which the subject vehicle started the maneuver
`ArrivalLane`	Integer	In case of lane changing, this variable contains the ID of the lane to which the subject vehicle is changing
`trajectoryLaneId`	Integer	The ID of the trajectory lane of the subject vehicle
`trajectorySpeed`	Double	The initial speed of the subject vehicle before performing any maneuver

bor. Since there are no top and top-rear neighbors, the first safety conditions are met if and only if the TTC between the subject vehicle and its front-top neighbor TTC_{S-TF} is higher than TTC_{thresh} when the subject vehicle is overtaking. But when overtaking, the difference between its speed and its front neighbor speed is at least $20km/h$, which is $V_1^{overtaking} = 80km/h$, and therefore, the speed difference between the subject vehicle and its top-front neighbor during overtaking would be $\Delta V_{S-TF} = 20km/h = 5.55m/s$, which gives us a $TTC_{s-TF} = 9s$ (equation 6.4).

$$\begin{aligned} TTC_{S-TF} &= \frac{d_2}{V_1 - V_3} \\ &= 50/5.55 = 9s \end{aligned} \tag{6.4}$$

Since $TTC_{S-TF} < TTC_{thresh}$, then the subject vehicle risks collision with its top-front neighbor if it performs the overtaking maneuver, and therefore cooperation is needed in order to meet the first safety conditions. Since cooperation level is set to 1, the top-front neighbor may only change its speed by up to $10km/h$. To reach TTC_{thresh}, the speed difference between the subject vehicle and its top-front neighbor must be at most $\Delta V^*_{S-TF} = 15km/h = 4.16m/s$, which gives $TTC^*_{S-TF} = 50/4.16 = 12s$. Therefore, the top-front neighbor must speed up to $V_3^* = 65km/h$

Table 6.5: Vehicle's Meta Data Message Definition.

Message Name:	vehicleMetaData	
Transmitted on:	EgoMetaData, NeighborsMetaData	
Field Name:	**Field Type:**	**Field Description:**
`VehicleID`	Integer	A unique identifier of the subject vehicle
`EgoCoord`	PoseStamped[2]	The subject vehicle's pose in the global frame of reference
`LaneInfo`	egoLane 6.3	All lane information of the occupied lane
`length`	Double	The length of the subject vehicle
`width`	Double	The width of the subject vehicle
`vehicleSpeed`	Double	The current speed in m/s
`wheelAngle`	Double	The current wheel angle in rad
`highLevelState`	vehicleHigh-LevelState 6.4	The current high level states of the subject vehicle
`toDistance`	Double	The inter distance between the subject vehicle and a neighbor vehicles, this parameter is set inside the subject vehicle for each neighbor in the neighbors list
`timeToCollision`	Double	the TTC between the subject vehicle and another specific neighbor, this parameter is set inside the subject vehicle for each neighbor in the neighbors list

so that the subject vehicle may perform the first part of the overtaking safely. Since cooperation level is set to 1, and $\Delta V_3^* = 5km/h$, the top-front neighbor will cooperate and speed up to $65km/h$ until the subject vehicle finishes the overtaking maneuver.

6.5.1.2 Scenario 2

In this scenario we will work on the configuration of Fig. 6.11. Just as with the first scenario, the subject vehicle encounters a front neighbor which is moving at a lower speed, and therefore intends to overtake it. Besides the front neighbor, the subject vehicle also detects a top-rear neighbor throughout communication. The TTC between the subject vehicle and its top-rear neighbor is $TTC_{S-TR} = 60/5.55 = 10.9s < TTC_{thresh}$, therefore the subject vehicle may not overtake its front neighbor unless its top-rear neighbor cooperates. The speed at which the top-rear neighbor must be moving could be computed from $TTC^*_{S-TR} = 12s$, which yields $\Delta V^*_{S-TR} = 5m/s = 18km/h$. Finally, assuming cooperation level is set to 2, the top-rear neighbor must slow down to $V^*_{TR} = 82km$, which is less than the $20km/h$ speed limit difference in cooperation level 2. However, in order to slow down to this speed, the rear neighbor of the top-rear neighbor of the subject vehicle will have to slow down as well, otherwise the 2 vehicles will collide. Since vehicles cannot cooperate if that cooperation affects other vehicles, the top-rear neighbor cannot cooperate with

Table 6.6: Neighbors List Message Definition.

Message Name:	neighbList	
Transmitted on:	NeighborsList	
Field Name:	**Field Type:**	**Field Description:**
`neighbMetaData`	vehicleMeta-Data	An array that contains the data of all neighbor vehicles

Table 6.7: Decision Message Definition.

Message Name:	decision	
Transmitted on:	Decision	
Field Name:	**Field Type:**	**Field Description:**
`speedRef`	Double	The reference speed sent by the CLMM to the subject vehicle's speed control unit
`LaneRef`	Integer	The target lane that the vehicle must change to, sent by the CLMM to the vehicle's trajectory control unit

the subject vehicle. In this case, the subject vehicle will be forced to slow down to $60km/h$ and follow its front neighbor until it is possible to overtake.

6.5.2 Validation by Simulation

The second method we used to validate the CLMM is validation by simulations. Since we assume that each CS is an independent system (operationally and managerially), we have decided to use a simulation software that allows us to integrate the logic in each CS into what is called agents, and perform agent-based simulation. Furthermore, we consider visualizing the behavior of the system as more important than just performing numerical simulations and analyzing the outputs, since numerical simulations are most useful when we know what we are searching for in a simulation. However, when there is a potential emergent behavior that might rise in the system, visualizing the simulation provides us with a more general understanding to the effect of the system on the environment. With these two points in mind, we have decided to use Anylogic[3], a simulation software that offers the possibilities to perform graphical simulations, and that is adapted for agent-based simulations. The first step in building the simulation is building the environment. As we have discussed in Section 6.3, the environment that interests us is 2-lanes or more highways with all lanes having the same direction. Therefore, our environment consists of a 3-lanes single direction highway. The second step is to develop the agents. Each CMMAV vehicle is considered as an independent agent, therefore vehicles have a trajectory lane, and a trajectory speed. The logic behind each decision an agent makes is explained earlier in Section 6.3.4, and presented in Fig. 6.8. The state chart that contains the logic agents use to perform a lane changing maneuver, in the Anylogic software, is shown in Fig. 6.13. Agents establish communication when they identify each others as neighbors. In Fig. 6.12, the last vehicle from the right of the figure only considers the second vehicle from the right of the figure (both are

[3]www.anylogic.com

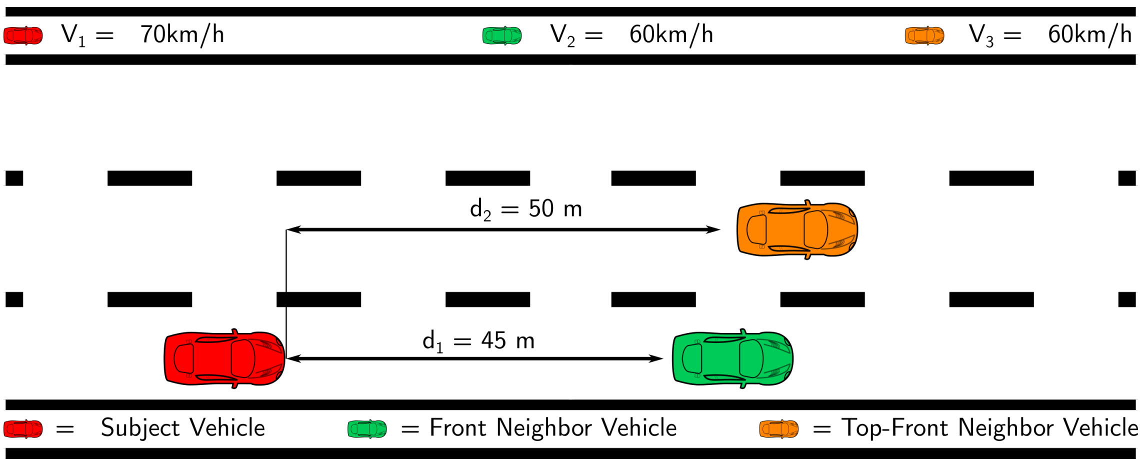


Figure 6.10: Scenario 1: front neighbor and top-front neighbor.

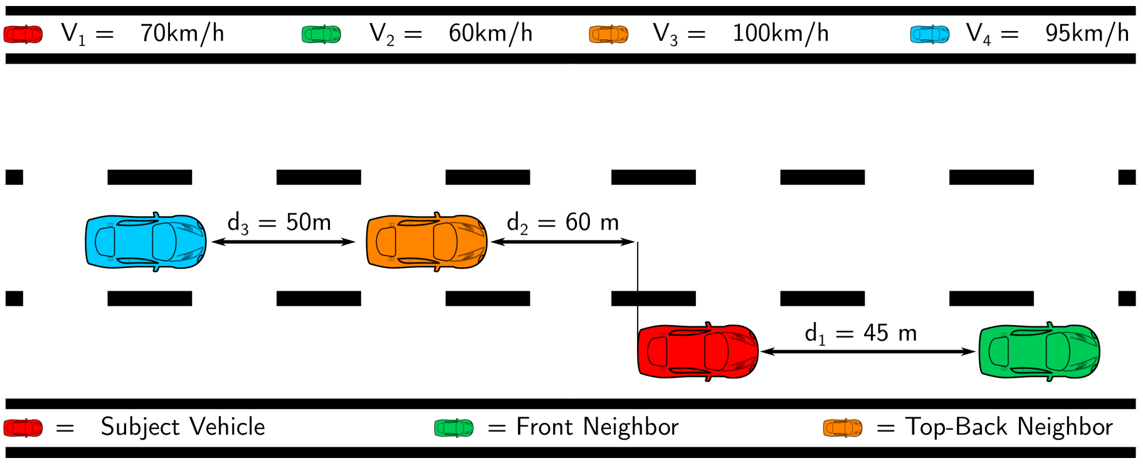


Figure 6.11: Scenario 2: front neighbor and 2 top-rear neighbors.

in the middle lane) as front neighbor (depicted by the black connector), and does not consider the first vehicle from the right of the figure (also in the middle lane) to be a front neighbor since they have a neighborhood degree of 2, and the vehicle in the bottom lane only considers the second vehicle from the right of the figure to be its top-front neighbor, depicted by the red connector.

At a random time between $2 - 5$ seconds, an agent is generated on either the bot-

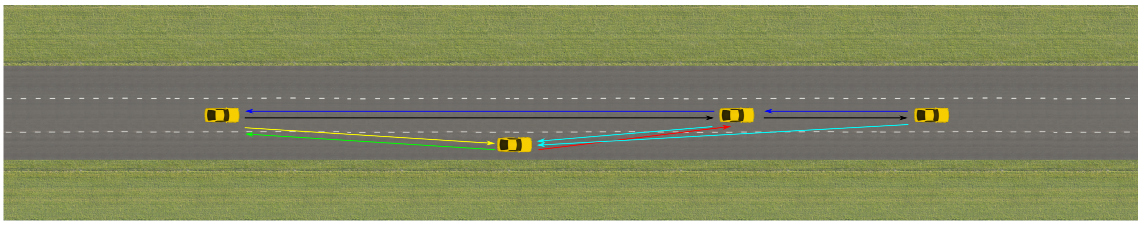


Figure 6.12: Inter-agent links show which agents are neighbors.

tom or the middle lane, with a random speed between the lower and higher speeds allowed on that specific lane, on the left of Fig. 6.12, and moves to the right of the Figure on that lane using that speed, until it reaches the end of the lane where it is destroyed. During this time, whenever an agent encounters a front neighbor, and

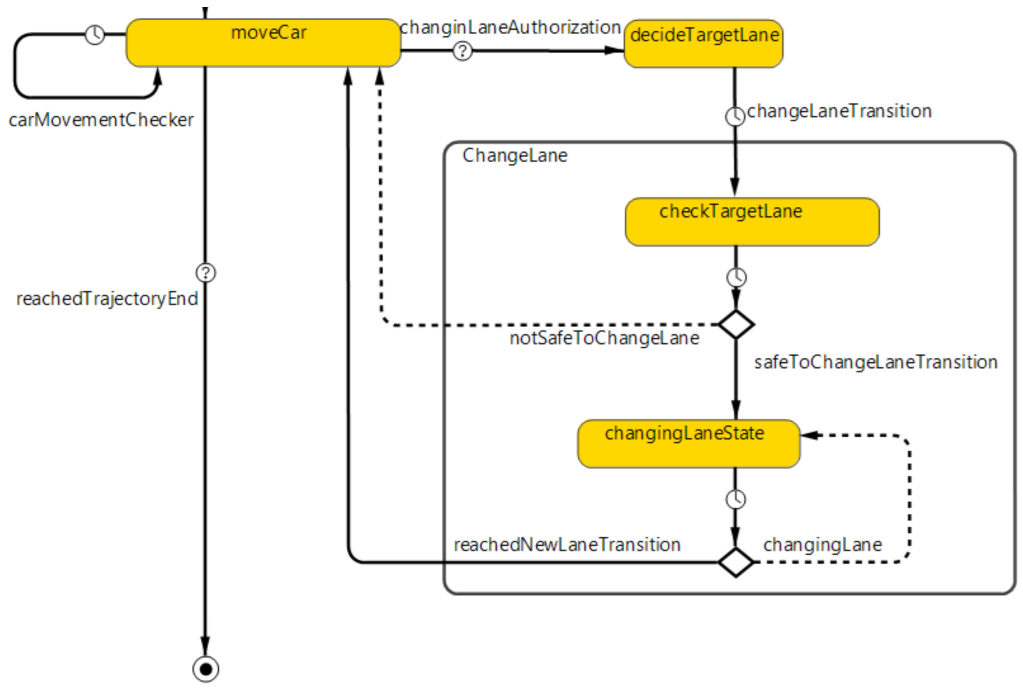


Figure 6.13: The state chart that describes the logic given to agents in Anylogic.

the situation allows it, it overtakes its front neighbor in order to maximize the time it spends on its trajectory lane moving at its trajectory speed. Because otherwise, it must slow down to match its front neighbor's speed and keep that speed until that neighbor changes its lane or speeds.
The first objective of the simulation was to study the global behavior of the system by observing any collision or congestion that might occur. The second objective was to ensure that all security distances are respected, and that no maneuver led to any violation to that distance. Multiple simulations were performed, and as a sample, in one case where 142 vehicles were generated, 32 overtaking maneuvers took place, 3 of which were double overtaking maneuvers. During these maneuvers, no collisions, congestion, or violation of the security gap were observed. Since the CLMM only apply high level control over CS during the maneuvers, and since we do not consider the way vehicles perform those maneuvers, we assume that the CLMM met the first objective.

6.5.2.1 Simulation Drawbacks

In the simulations performed, all agents perform lane changing maneuvers in the same way, because we introduced the same logic to all agents, including the way they perform lane changing. Even though this logic is independent from other agents, that is, the way agents change lane does not take into consideration neighbors, we are not sure about the effect this has on the system. That is why in future simulations, we will use multiple different ways that agents could use to change their lanes, to compare the behavior of the system under that condition, with its behavior when all agents perform the maneuvers in the same manner.

6.5.3 Validation by Experimentation

The third step of validation is the validation by experimentation. In this validation step, we have implemented the CLMM on three autonomous vehicles of the Heudiasyc Laboratory: Carmen (Fig. 6.14), and tow Renault Zoé vehicles of the platform APACHE (Fig. 6.15). The two Zoé vehicles are equipped with autonomous driving, and the three vehicles are equipped with communication functionalities. For further details about the architecture of each Zoé vehicle, readers are invited to see (Xu et al. 2018).

Vehicles are labeled "Vehicle i", where $i = 1, 2, 3$, where the subject vehicle is

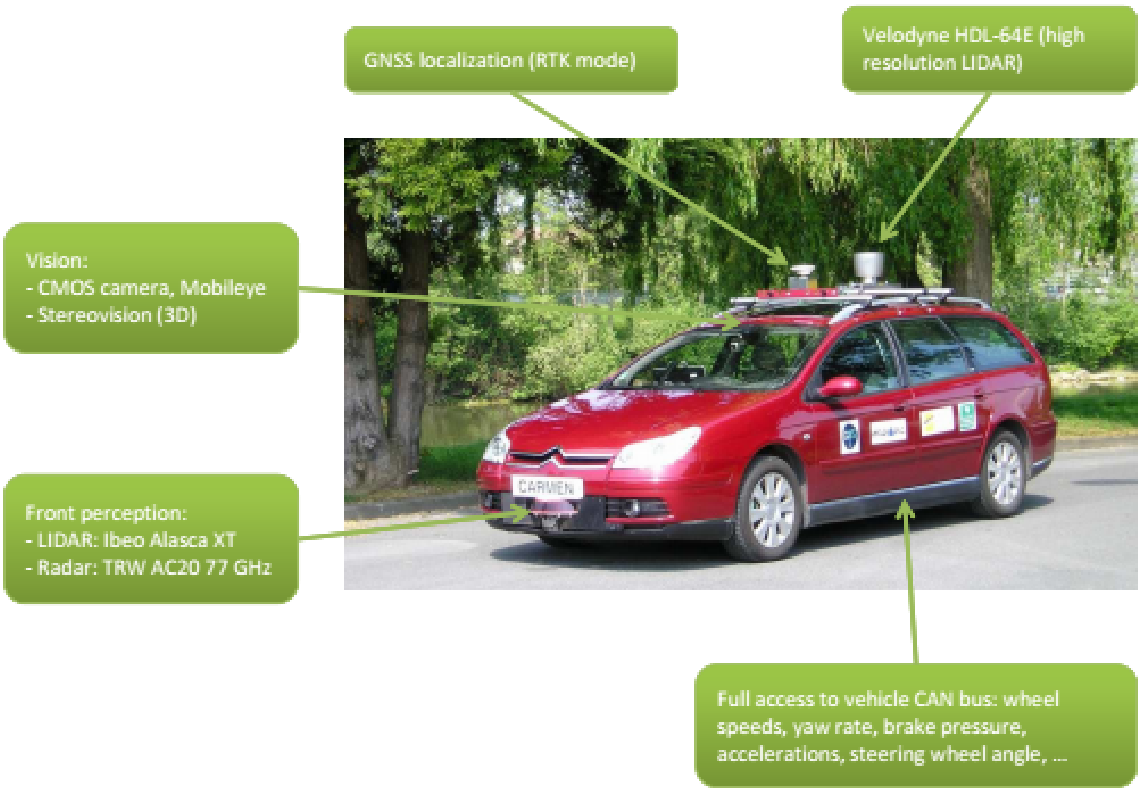

GNSS localization (RTK mode)
Velodyne HDL-64E (high resolution LIDAR)
Vision:
- CMOS camera, Mobileye
- Stereovision (3D)
Front perception:
- LIDAR: Ibeo Alasca XT
- Radar: TRW AC20 77 GHz
Full access to vehicle CAN bus: wheel speeds, yaw rate, brake pressure, accelerations, steering wheel angle, ...
CARMEN

Figure 6.14: Citröen 5c: Carmen.

"Vehicle 2" (red color), the front neighbor is "Vehicle 3" (blue color), and the top neighbor is "Vehicle 1" (green color). Figure 6.16 shows the graphical interface that is used to monitor the states of the subject vehicle (left side of the interface), as well as the states of the different neighbors when they are detected (right side of the interface). The initial positions of the 3 vehicles are shown in Fig. 6.17 as colored circles, whereas lanes are shown in black lines, annotated with their respective lane number, as well as in Fig. 6.18, which shows an approximation of the initial configuration. Two cases were explored in these tests: a case where "Vehicle 1" is always in the top position (Section 6.5.3.1), and another case where it is in the top-rear position (Section 6.5.3.2).

6.5.3.1 Case 1

In this case, the subject vehicle was forced to slow down and follow its front neighbor since it was not allowed to change its lane due to the presence of the top neighbor. Figure 6.19 shows the positions of the 3 vehicles during this test, where we can observe that the subject vehicle ("Vehicle 2") is always behind its front neighbor, and no lane changing or overtaking maneuver were performed. Figure 6.20 shows the speeds of the 3 vehicles, as we can see, the speed of the subject vehicle decreased around sample 130 in order to avoid collision with its front neighbor since it was

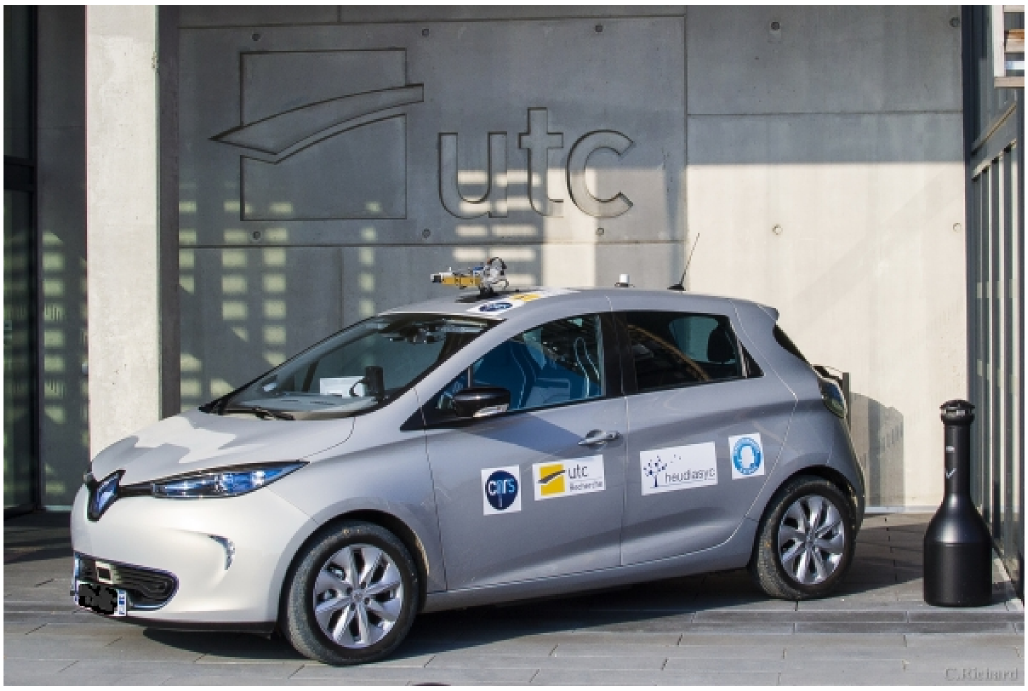


Figure 6.15: Renault Zoé: Apache.

not allowed to overtake it.

6.5.3.2 Case 2

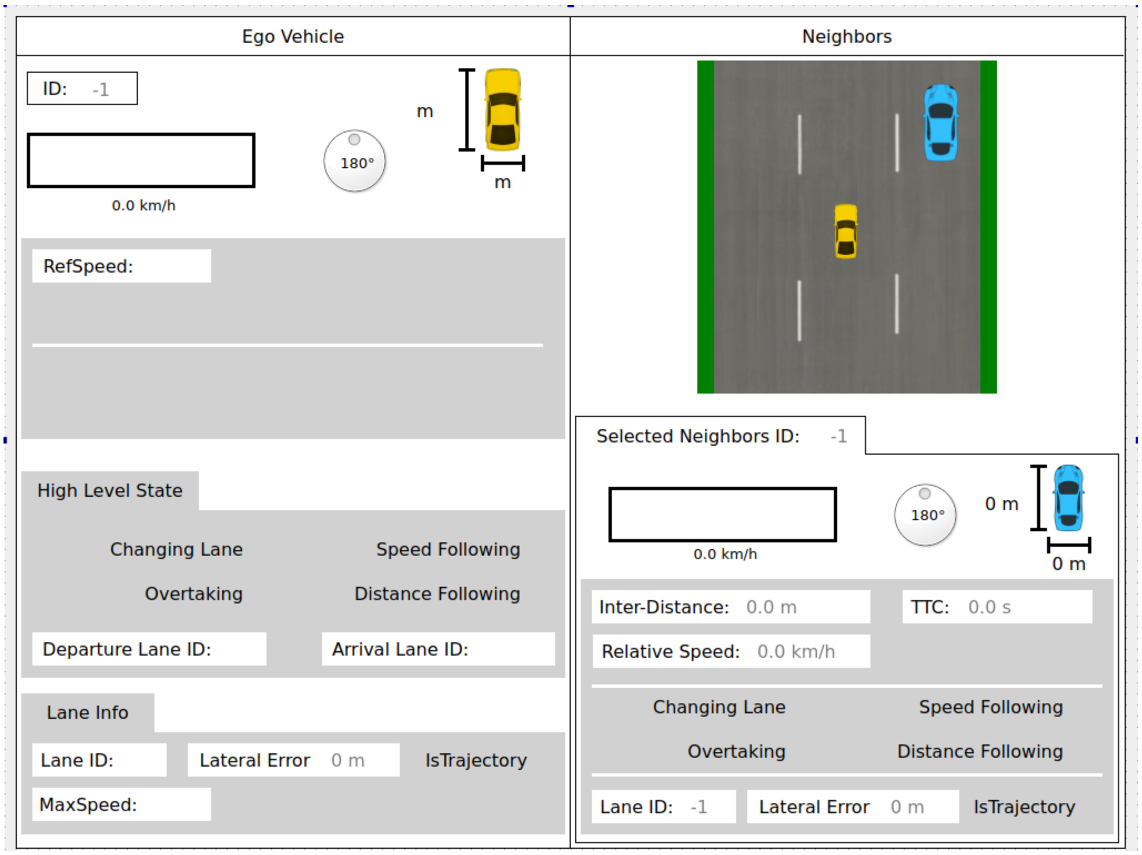
In this case, unlike the previous case, "Vehicle 1" was in the top-rear position with respect to the subject vehicle. As we can see in Fig. 6.21, the subject vehicle overtook its front neighbor while its top-rear neighbor was behind it. In Fig. 6.22, we can see that the subject vehicle's speed increased at around sample 100, where it started to perform the overtaking maneuver.

6.5.3.3 Experimentation Drawbacks

Even though the overall behavior during these tests was close to the desired behavior, however, more real-life validations must be performed. Due to the limited space for performing the experiments, it is hard to see the effect of parameters such as the time-to-collision threshold, and how it affects the behavior of the system when operating in the intended environment (highways). Furthermore, the experiments did not involve requests from and to different vehicles, but rather it involved communication of information between the different vehicles, and in consequence, the decision to overtake or not (based upon the situation). Further experiments must be performed in order to observe the behavior of the CLMM in an environment similar to the intended, and to study the effect of various parameters, and explore more complex scenarios.

6.6 CLMM Challenges

Similar to any SoS, the CLMM faces different challenges because of the interconnected dynamic of its environment. The current architecture and design supports an environment which consists only of autonomous vehicles, because the ultimate objective is to create a SoS that can strive in that specific environment when its ready. However, the transition from the current environment (human-driven vehicles) to


Ego Vehicle

ID: -1

0.0 km/h

180°

m

m

RefSpeed:

High Level State

Changing Lane

Speed Following

Overtaking

Distance Following

Departure Lane ID:

Arrival Lane ID:

Lane Info

Lane ID:

Lateral Error 0 m

IsTrajectory

MaxSpeed:

Neighbors

Selected Neighbors ID: -1

0.0 km/h

180°

0 m

0 m

Inter-Distance: 0.0 m

TTC: 0.0 s

Relative Speed: 0.0 km/h

Changing Lane

Speed Following

Overtaking

Distance Following

Lane ID: -1

Lateral Error 0 m

IsTrajectory

Figure 6.16: Graphical Interface for Monitoring Vehicles' states.

the future environment (fully autonomous vehicles) is gradual, and it is expected that at some point, autonomous vehicles might share the roads with human-driven vehicles. Furthermore, in the future environment, we assume that not all vehicles will be equipped with the CLMM, and that there will be other applications that achieve the same objectives, so sometimes the subject vehicle might have conflict with a neighbor vehicle that is autonomous, but is equipped with another system that is incompatible with the CLMM. To account for these challenges, several solutions might be proposed. In the first case, one solution is to refine the CLMM so that in human-driven communicating vehicles it behaves as a driving assistance system, where the drivers receive the results of the negotiation done by the CLMM in both vehicles, so that they can perform the action that allows the neighbor to perform the overtaking. Whereas in the second case, the solution would be on a higher level, where the maintainers of the incompatible applications refine their applications so that vehicles equipped with those applications are able to cooperate.
Another challenge that faces the CLMM is the V2V communication technology it self. While V2V communication is a good solution for vehicles discovery and information sharing, it still faces several problems, one of which is the reliability of this solution. When the number of the vehicles increases in an area, the wireless network they use to exchange messages might get overwhelmed, which leads to packet losses. In the CLMM, we suppose that the communication is reliable, and that is to test and validate the idea behind the CLMM itself. The problems communication might be investigated later to study the consequences on the system.
When we introduce incentive into the equation, one of the technical challenges that

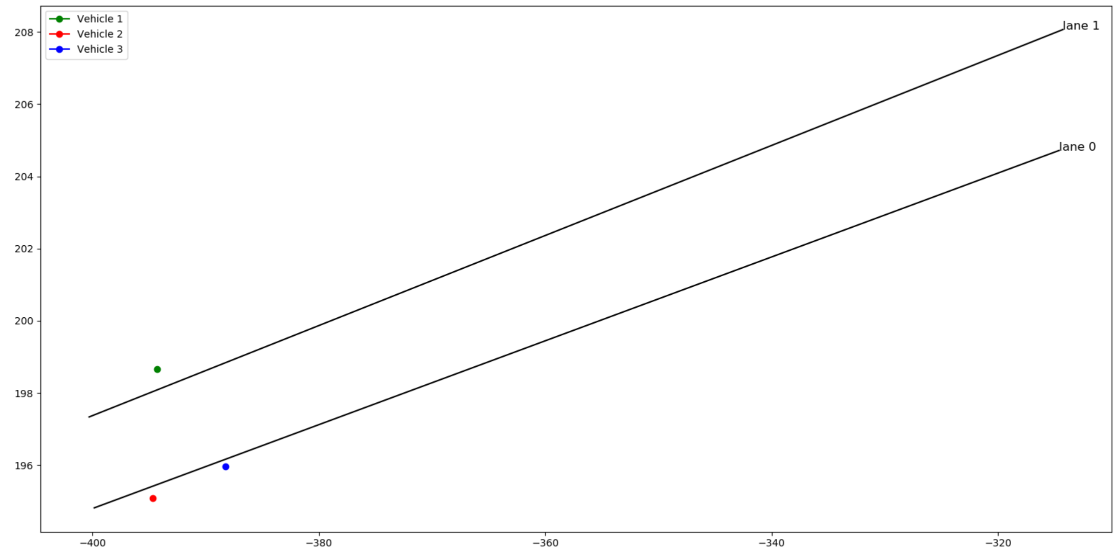


Figure 6.17: Vehicles' initial positions with respect to lanes.

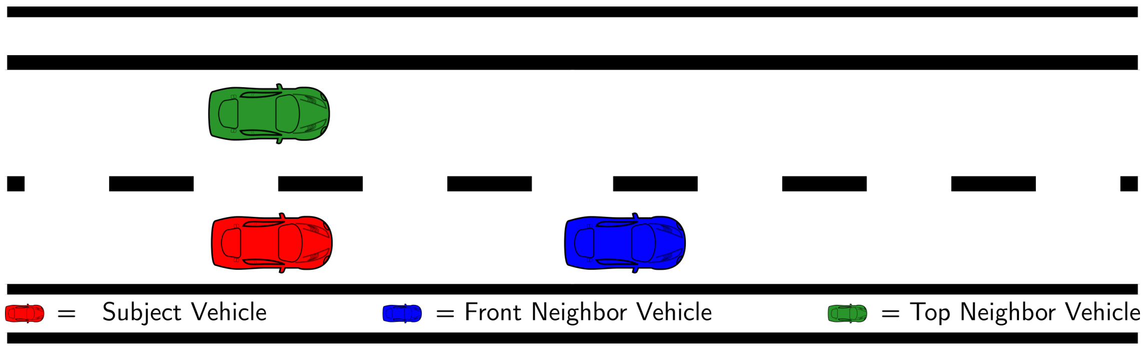


Figure 6.18: Vehicles' initial positions approximation.

are expected is how to register the data related to the cooperation between CMMAV vehicles. Data such as which vehicle requested cooperation, which vehicle cooperated with that request, on which cooperation level the cooperation happened, etc. These data are important because the principle behind incentive is to provide "rewards" to vehicles for their cooperation, and therefore they will be used to identify vehicles that cooperate and to reward them based upon the chosen incentive mechanism. Our solution to this challenge would be to store each request and response a CMMAV vehicle has over a period of time (let us say daily for example), at the end of that period of time, all the stored information are then uploaded to cloud-based application. Since cooperation in the CLMM involves 2 vehicles, then we have 2 sources of information that could be cross examined to verify the integrity of the data. After verifying each cooperation log, the cloud-based application generate rewards for each vehicle based upon a specific algorithm.
Connecting the vehicles to the internet to generate rewards, and the wireless communication technology used in vehicles lead to a potential risk related to cyber security. A malicious software altering the messages exchanged between vehicles, or between vehicles and the cloud-based application, might lead to undesired consequences. Similar to communication challenges, this challenge is inherited from the technology used itself. There are numerous possible solutions to this challenge,

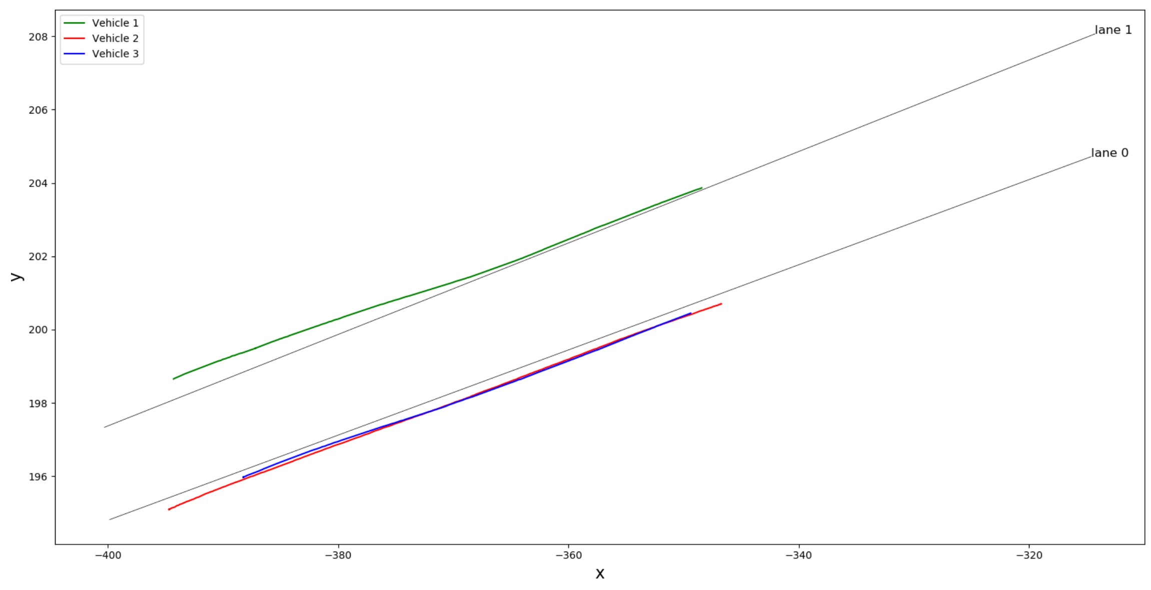


Figure 6.19: Vehicles' positions during the first test.

but they depend mainly on the adopted architecture in autonomous vehicles in the future. The first architecture that we could think of is the one that says that the vehicle's main system itself is responsible for protecting the different messages it sends and receives, whereas the second architecture is that any application installed on the vehicle is responsible for protecting the messages, and that the vehicles responsibility is only sending and receiving messages, and then routing them to the different applications. The first architecture does not entail any specific requirements for the CLMM, whereas the second architecture means that we need to add the cyber security requirement to the CMMAV, and therefore to integrate that into the CLMM.

6.7 Conclusion

The CLMM is the first application that uses the CMMAV recommendation in order to enable cooperation during overtaking maneuvers on highways. When the subject vehicle detects a front neighbor that is slower than itself, the CLMM handles the decision making inside the subject vehicle in order to evaluate the situation and send the appropriate requests to its neighbor vehicles. The CLMM uses wireless communications to detect the different neighbors in the vicinity of the subject vehicle, constructs a local map that shows the relative positions of the neighbors with respect to the subject vehicle, and then evaluates the possibility of safely performing an overtaking. Otherwise, it sends the appropriate requests to the neighbors in order to facilitate the maneuver. If no cooperation is possible, the subject vehicle is then forced to follow its front neighbor at the same speed until either a neighbor accepts to cooperate, or the situation allows an overtaking without cooperation. Since it uses the CMMAV recommendations, there are some constraints on the way the maneuvers are executed, for example, in principle, autonomous vehicles may overtake to the left, however the actual traffic law does not allow that and therefore the subject vehicle may only overtake to the right. We used 3 methods to perform the initial validations: formal scenarios, simulation, and experimentation on autonomous vehicles. The first validation method, formal scenarios, was used

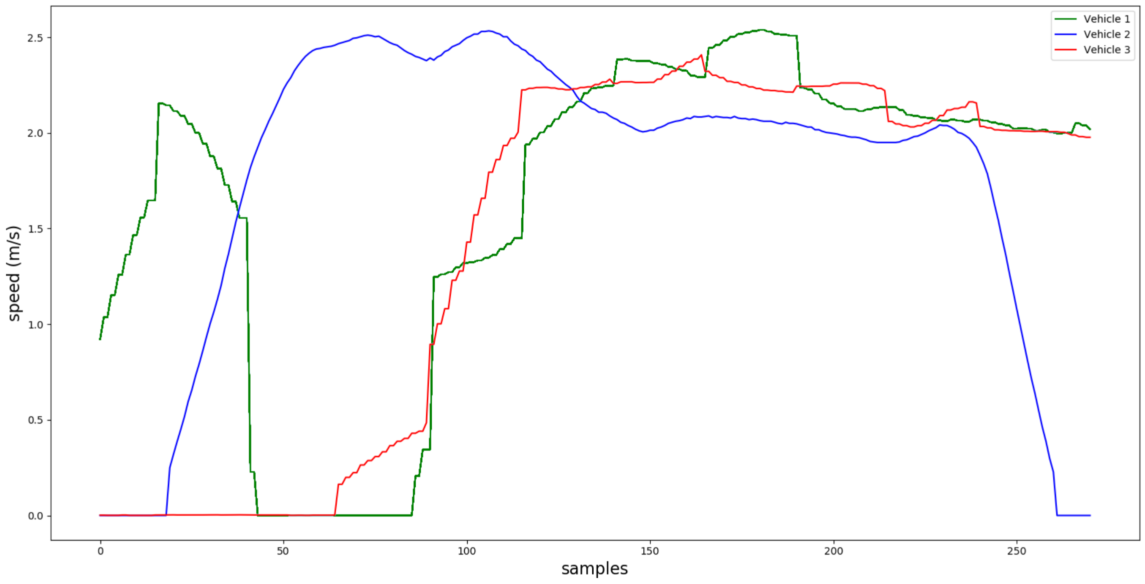


Figure 6.20: Vehicles' speeds during the first test.

in order to validate initially that the way vehicles perform the maneuvers does not affect the performance, and that by using relative speeds and distances we are able to achieve our objectives. The simulations performed showed us that the CLMM works as described in the formal scenarios, and that there are no undesirable effects (congestion, collisions) that resulted from its use. Finally, the experimentation on autonomous vehicles of the Heudiasyc laboratory further assured that the idea behind the CLMM is valid. Some shortcomings of the experimentation are to be solved in future work, since the distances and speeds at which they were performed are way less than the intended operational environment.

The CLMM is a SoS, and is affected by its environment and the technologies involved in its operation just as any SoS is affected by its environment. The CLMM depends heavily on wireless communication in order to acquire the necessary information to perform, and therefore it is exposed to the same vulnerabilities wireless communication has. Therefore, the wide spread and adoption of such SoS (SoS that use collective functionalities) will not be achieved until those technologies are reliable.

To benefit from the simulation tool that we consider very useful, the case where CMMAV vehicles share the roads with non CMMAV vehicles will be studied using simulation to study the behavior of the SoS in such case. Because it is important to identify if there is a CMMAV vehicle to non CMMAV vehicle ratio threshold at which the CMMAV produces undesirable behaviors in the presence of non CMMAV vehicles, and to study the behavior on different ratios of CMMAV vehicles to non CMMAV vehicles sharing the roads.

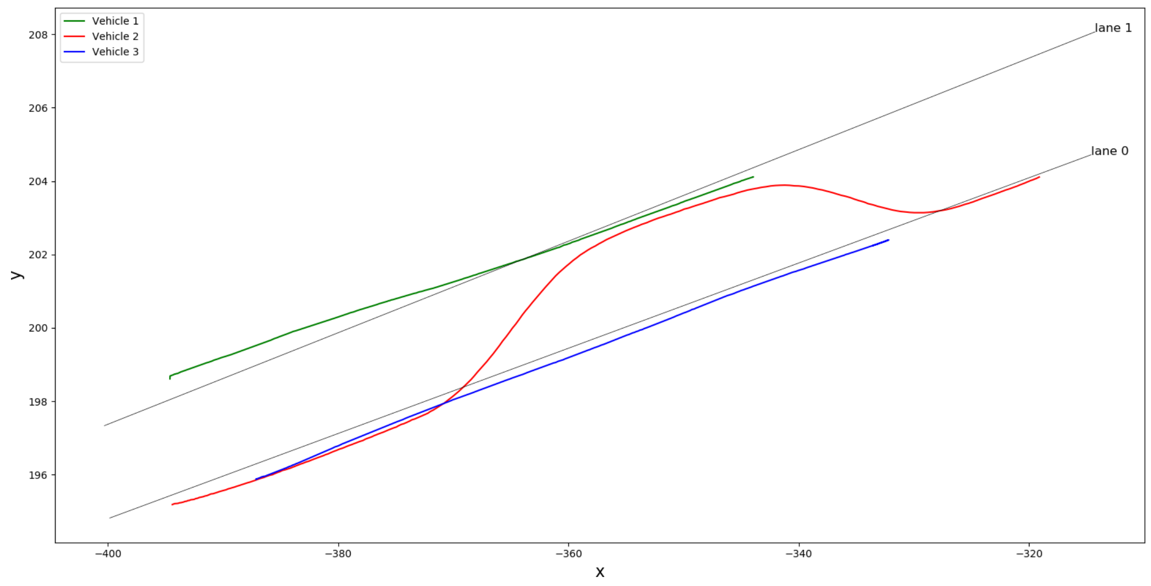


Figure 6.21: Vehicles' positions during the second test.

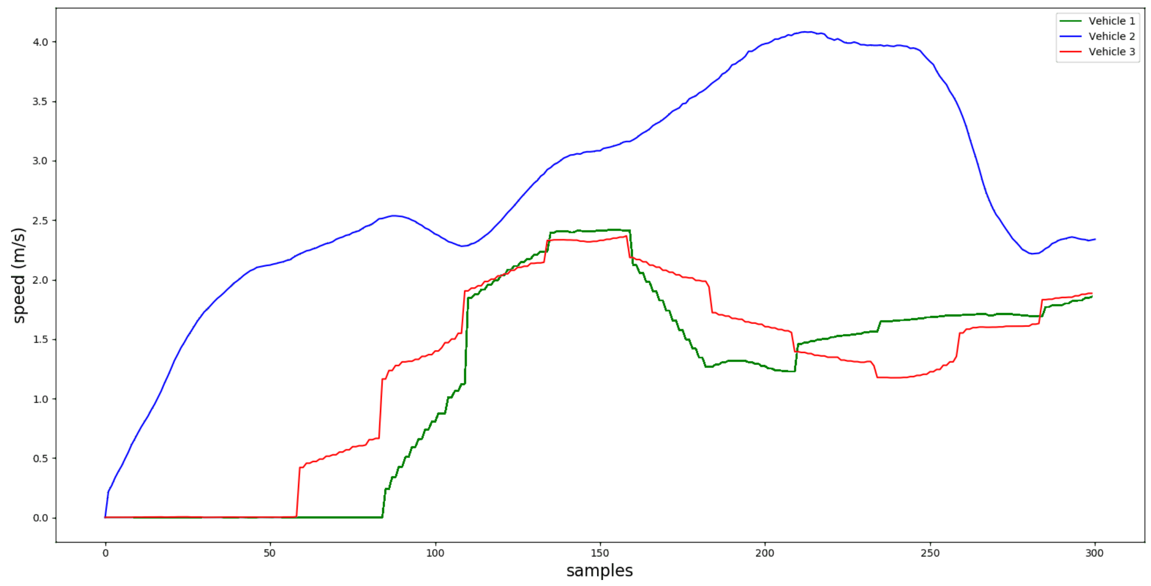


Figure 6.22: Vehicles' speeds during the second test.

Chapter 7

Conclusion

SoS are large complex systems, that result from the interactions between different independent systems, namely CS, and they exist in the majority of domains of applications. SoS emerge naturally due to the increasing interactions between independent systems, and the service-based business models adopted by various systems, or they are built and designed to serve a specific purpose. SoS are complex systems, and thus they inhibit emergent behaviors just as any complex system does. Several taxonomies might be used to group different SoS, depending on what aspects of the SoS are we interested in: management taxonomies group SoS based on their management style, whereas anticipation taxonomies describe the degree of anticipation of engineering activities in the SoS during the operation, and contracting taxonomies refer to the way SoS acquire their CS, and how CS contribute to the SoS.
SoS are dynamic systems in the sense that CS might join or leave the SoS during operation time, and they are closely interacting with their environment, which leads to a number of challenges that face SoS that do not face traditional systems, and therefore, traditional SE is not enough to address these challenges. That is why developing a SoSE discipline is important to engineer and build better and more resilient SoS.
In the context of systems control, control may be applied using a centralized or a non-centralized paradigm. In centralized paradigms a single controller is responsible for acquiring the states of the overall system and for sending inputs to the different actuators, whereas in non-centralized paradigms, whether be it hierarchical, distributed or decentralized, the task of control is distributed over multiple controllers. In hierarchical paradigm, multiple levels exist with authority going from bottom to top, and no communication is allowed horizontally on the same level. Distributed paradigm lacks the vertical distribution of tasks, and it uses inter-part communication to achieve the desired behavior. Decentralized paradigm is similar to distributed paradigm in that it has no vertical levels of authority, and similar to the hierarchical paradigm, no communication occurs between parts.
There are similarities between types of SoS and traditional complex systems on the organizational level. Directed and acknowledged SoS often have a certain hierarchy between CS, and on the same level we might find communication between CS. Collaborative SoS are so much similar in organization to a graph of nodes, where nodes share information and each node is responsible for making its own decisions. Virtual SoS, on the other hand, are similar to a decentralized paradigm, where CS are not aware of the presence of other CS in the same SoS. However, unlike control

in traditional systems, control in SoS is the process of defining the different relations between CS, and what actions should CS do in order to achieve a desired behavior on the SoS level.
To do so, we are going to need a tool that can capture the complexity and dynamic nature of SoS. Architecture frameworks, most notably multi-views frameworks, enables us to represent SoS in a way that gives us the ability to understand SoS, and to provide the rules for CS. Furthermore, they can be integrated into tools that may be used to perform analysis on the performance and behavior of the SoS during design time as well as during the run time, in order to aid in the decision-making process concerning different aspects of the SoS.

To achieve reliable and commercial autonomous driving, the development of collective capabilities is an important step. For this purpose, and using SoS principles, and a contribution to ITS, the CMMAV, a framework conceived to guide the development of such capabilities in different transportation systems was developed. The main objective of the CMMAV is to provide recommendations and logical architectures for any cooperative application that targets autonomous vehicles. To account for the dynamic environment of CMMAV, and the diversity of stakeholders and CS, CMMAV uses agile development process based on use-cases that define the stakeholders and CS, and a SoS approach to map the requirements of different stakeholders into a set of capabilities and validation requirements that may be used by any user who wishes to develop CMMAV based capabilities in their systems. To represent the system, a seven-views framework was derived from SoS-ACRE framework, and consists of: use-case view, organizational view, sources view, requirements view, capabilities view, validation view, and traceability view. Each view contains a high-level representation that uses a hierarchical tree to represent the different relations within this view, and a detailed view via forms that describes each element in this view, and is used to generate the relations that exist between different views in the traceability view.
The first use case that was adopted to build the current draft of the CMMAV is also considered to validate it, is the "overtaking on highways" use-case. The CLMM is the first application that uses CMMAV recommendations in order to enable cooperation during overtaking maneuvers on highways. When the subject vehicle detects a front neighbor that is slower than itself, the CLMM handles the decision-making inside the subject vehicle in order to evaluate the situation and send the appropriate requests to its neighbor vehicles. It uses wireless communication to detect the different neighbors in the vicinity of the subject vehicle, constructs a local map that shows the relative positions of the neighbors with respect to the subject vehicle, and then evaluates the possibility of safely performing an overtaking, otherwise, it sends the appropriate requests to the neighbors in order to facilitate the maneuver. If no cooperation is possible, the subject vehicle is then forced to follow its front neighbor at the same speed until either a neighbor accepts to cooperate, or the situation allows an overtaking without cooperation.
Since it uses CMMAV recommendations, there are some constraints on the way the maneuvers are executed, for example, in principle, autonomous vehicles may overtake to the right. However, the actual traffic laws does not allow that and therefore, the subject vehicle may only overtake to the left. We used three methods to perform the initial validations: formal scenarios, simulation, and experimentation

on autonomous vehicles. Formal scenarios were used in order to validate initially that the way vehicles perform the maneuvers does not affect the performance, and that by using relative speeds and distances we are able to achieve our objectives. The simulations performed showed us that the CLMM works as described in the formal scenarios, and that there are no undesirable effects (congestion, collisions) that resulted from its use. Finally, the experimentation on autonomous vehicles of the Heudiasyc laboratory further assured that the idea behind the CLMM is valid. Some shortcomings of the experimentation are to be solved in future work, since the distances and speeds at which they were performed are way less than the intended operational environment.
The CLMM is a SoS, and is affected by its environment and the technologies involved in its operation just as any SoS is affected by its environment. The CLMM depends heavily on wireless communication in order to acquire the necessary information to perform, and therefore it is exposed to the same vulnerabilities wireless communication has. Therefore, the wide spread and adoption of such SoS (SoS that use collective capabilities) will not be achieved until those technologies are secure.

7.1 Perspectives

The work presented in this thesis opens the door to numerous potential applications and studies in the context of autonomous vehicles and SoS architecture. Developing collective capabilities in autonomous vehicles is very important towards achieving reliable autonomous driving, since autonomous vehicles are part of a wider SoS that contains many stakeholders and CS, with different types of interrelationships. By modeling the environment of autonomous vehicles, we get to know the different systems and stakeholders that are related to autonomous vehicles. By analyzing the results, we can find new cooperation opportunities between autonomous vehicles and other systems in their environment (e.g. infrastructure, service providers). Furthermore, this allows us to come up with better incentive mechanisms to favor cooperation between the different players.

The CMMAV as a framework, was developed with the idea to enable better communication between the different stakeholders in order to elicit requirements in an efficient and unambiguous way. It is also developed in a way that favors the evolution of the SoS, and enables constant refinement. However, to get the most from it, it is best to be developed as a software, in order to benefit from traceability, and to enable analysis. Another aspect of the CMMAV may also be the subject of further studies, which is the process that was used to build it, which we think is suitable for many SoS. For example, consider an entity which develops an autonomous vehicle. This process involves the effort of various teams, with different disciplines, focuses, and objectives. By using the process used to develop the CMMAV, we can develop a framework that guides the different teams in their missions, while at the same time preserving their independence.

As for the application of the CMMAV, they are not limited to the cases treated by the CLMM, and there are numerous applications that may be developed using the CMMAV recommendation. Shared parking, where autonomous vehicles communicate with infrastructure in order to find parking spots are another example of applications that involve cooperation between autonomous vehicles and other systems in their environment. Another application is the cooperative traffic signals, where light signals communicate with autonomous vehicles in order to make decisions about timings.

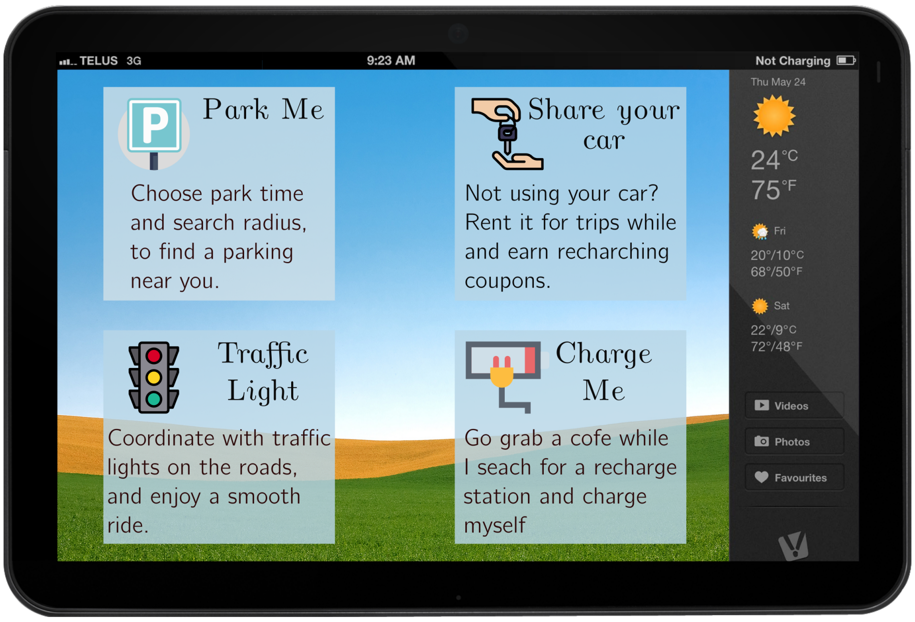
The CLMM represents a proof of concept, and has so much room for improvement. By considering acceleration during maneuvers, we increase the accuracy of the maneuvers, and in consequence we improve the system. Furthermore, if we increase the neighborhood degree to more than 0, the potential for cooperation increases, but at the same time, so does the risk of having undesired behaviors such as congestion and traffic shockwave. These are potential future works that will improve the CLMM.

Finally, if the CMMAV is developed further, a challenge similar to the Grand Cooperative Driving Challenge (GCDC) may be organized, where organizations that work on the development of autonomous vehicles develop CMMAV-based applications, and compete in different scenarios that involve cooperation. This event offers several benefits: it helps validate the idea behind collective capabilities on a large scale, opens the eyes on the importance of such capabilities, which hopefully increases the interest in this aspect of autonomous driving development.

7.2 Final thought

Autonomous driving is still in the development stage, and we are still far from having fully autonomous vehicles commercially available. It is still unclear what type of environment will emerge when we reach that stage. The driver-vehicle interaction may take several forms: autonomous vehicles might become similar to what smart phones are today, in the sense that driver might be able to install applications that control the behavior of the vehicle to some extent, or they might not change in that sense, and manufacturers will always be responsible for setting the behavior of their vehicles. If we were to imagine each scenario, the second scenario will be pretty much similar to the current state of the automobile, but with more advanced technologies in vehicles, and more communicating infrastructure. So let us imagine what it would be like in the first scenario.
First of all, there would be a dedicated network of V2V and V2X communication, that enables vehicles and infrastructures to communicate based upon needs, rather than proximity (the current state of communication). Then, there would be teams developing applications that target specific types of needs, which could be installed on vehicles, infrastructure, smart phones, etc. Examples of such applications are shown in Fig. 7.1. The owner of an autonomous vehicle that wants to go to a meeting can simply open the “Park me” application, chooses the park time and duration, and leaves the car. This application searches the dedicated network for available parking places, and based on proximity and availability of parking spots, an automated reservation system chooses a parking spot, and the vehicle drives itself to park at that location. In case the owner wishes to extend the parking duration, he can simply do so using the application on his mobile phone, and the reservation system either extends the parking time, or searches for another parking spot available elsewhere.
There exist a lot of applications similar to the “Park Me” application that could

be thought of, which might be very useful. Depending on how the development of autonomous driving and the environment around it proceed, the previous scenario might occur naturally. However, instead of waiting for the future to unfold, and adapting accordingly, it would be good if we anticipate it and guide its development.



Park Me

Choose park time and search radius, to find a parking near you.

Share your car

Not using your car? Rent it for trips while and earn recharching coupons.

Traffic Light

Coordinate with traffic lights on the roads, and enjoy a smooth ride.

Charge Me

Go grab a cofe while I seach for a recharge station and charge myself

Figure 7.1: Graphical interface in an autonomous vehicle.

Bibliography

Acheson, Paulette, Cihan Dagli, and Nil Kilicay-Ergin (2013). “Model based systems engineering for system of systems using agent-based modeling”. In: *Procedia Computer Science* 16, pp. 11–19.

Ackoff, Russell L (1971). “Towards a system of systems concepts”. In: *Management science* 17.11, pp. 661–671.

Alur, Rajeev (2015). *Principles of cyber-physical systems.* MIT Press.

Andary, James F and Andrew P Sage (2010). “The role of service oriented architectures in systems engineering”. In: *Information Knowledge Systems Management* 9.1, pp. 47–74.

Arnautovic, E., D. Svetinovic, and A. Diabat (July 2012). “Business interactions modeling for systems of systems engineering: Smart grid example”. In: *7th International Conference on System of Systems Engineering (SoSE)*, pp. 107–112.

Assaad, Mohamad Ali (2015). *Control of Systems of Systems: Case of several robots in interaction.* [Master Thesis]. Universié de Technologie de Compiègne.

Åström, Karl Johan, Tore Hägglund, and Karl J Astrom (2006). *Advanced PID control.* ISA-The Instrumentation, Systems, and Automation Society Research Triangle Park, NC.

Azar, Ahmad Taher and Quanmin Zhu (2015). *Advances and applications in sliding mode control systems.* Springer.

Ballagny, Cyril, Nabil Hameurlain, and Franck Barbier (2009). “Mocas: A state-based component model for self-adaptation”. In: *Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO).* Pp. 206–215.

Banks, Jerry et al. (2004). *Discrete-event system simulation.* Prentice Hall.

Belkadi, Farouk, Eric Bonjour, and Maryvonne Dulmet (2004). “Proposition of a Situation Model in View to Improve collaborative design”. In: INCOM.

Boardman, J. and B. Sauser (Apr. 2006). “System of Systems - the meaning of of”. In: *IEEE/SMC International Conference on System of Systems Engineering.*

Boardman, John and Brian Sauser (2008). *Systems thinking: Coping with 21st century problems.* CRC Press.

Boccara, Nino (2010). *Modeling complex systems.* Springer Science & Business Media.

Boulding, Kenneth E (Apr. 1956). “General systems theory—the skeleton of science”. In: *Management science* 2.3, pp. 197–208.

Bryson, Arthur Earl (2018). *Applied optimal control: optimization, estimation and control.* Routledge.

Butterfield, M. L., J. S. Pearlman, and S. C. Vickroy (Sept. 2008). “A System-of-Systems Engineering GEOSS: Architectural Approach”. In: *IEEE Systems Journal* 2.3, pp. 321–332.

Cai, Huaning and Andrew EB Lim (2011). “Decentralized control of a multi-agent stochastic dynamic resource allocation problem”. In: *50th IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pp. 6400–6406.

Cantot, Pascal and Dominique Luzeaux (2013). *Simulation and Modeling of Systems of Systems*. John Wiley & Sons.

Carlock, Paul G. and Robert E. Fenton (2001). “System of Systems (SoS) enterprise systems engineering for information-intensive organizations”. In: *Systems Engineering* 4.4, pp. 242–261.

Christofides, Panagiotis D et al. (2013). “Distributed model predictive control: A tutorial review and future research directions”. In: *Computers & Chemical Engineering* 51, pp. 21–41.

Cole, Reggie (2008). “SoS architecture”. In: *Systems of systems engineering: principles and applications* 37.

Collins, Bernie, Dr. Steve Doskey, and Dr. James Moreland (May 2016). *Relative Comparison of the Rate of Convergence of Collaborative Systems of Systems: A Quantified Case Study*. Presented to the Systems of Systems Engineering Collaborators Information Exchange, George Washington University.

Dahmann, J. S. and K. J. Baldwin (Apr. 2008). “Understanding the Current State of US Defense Systems of Systems and the Implications for Systems Engineering”. In: *2008 2nd Annual IEEE Systems Conference*, pp. 1–7.

Daniel, K. et al. (Mar. 2009). “AirShield: A system-of-systems MUAV remote sensing architecture for disaster response”. In: *3rd Annual IEEE Systems Conference*, pp. 196–200.

DeLaurentis, Daniel (2005). “Understanding transportation as a system-of-systems design problem”. In: *43rd AIAA Aerospace Sciences Meeting and Exhibit*, p. 123.

DeLaurentis, Daniel and Robert Callaway (2004). “A system-of-systems perspective for public policy decisions”. In: *Review of Policy research* 21.6, pp. 829–837.

Dickerson, Charles and Dimitri N Mavris (2016). *Architecture and principles of systems engineering*. CRC Press.

DiMario, M. J., J. T. Boardman, and B. J. Sauser (Sept. 2009). “System of Systems Collaborative Formation”. In: *IEEE Systems Journal* 3.3, pp. 360–368.

Dimarogonas, Dimos V, Emilio Frazzoli, and Karl H Johansson (2012). “Distributed event-triggered control for multi-agent systems”. In: *IEEE Transactions on Automatic Control* 57.5, pp. 1291–1297.

Dimarogonas, Dimos V, Michael M Zavlanos, et al. (2003). “Decentralized motion control of multiple holonomic agents under input constraints”. In: *42nd IEEE Conference on Decision and Control Proceedings.* 4, pp. 3390–3395.

Díaz, Nelson L et al. (2017). “Centralized control architecture for coordination of distributed renewable generation and energy storage in islanded ac microgrids”. In: *IEEE Transactions on Power Electronics* 32.7, pp. 5202–5213.

Durbha, S et al. (2006). “Semantics-enabled knowledge management for global earth observation system of systems”. In: *IEEE International Conference on Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 25–28.

Eisner, Howard, John Marciniak, and Ray McMillan (1991). “Computer-aided system of systems (S2) engineering”. In: *IEEE International Conference on Systems, Man, and Cybernetics, 1991: Decision Aiding for Complex Systems, Conference Proceedings*, pp. 531–537.

Elshenawy, Mohamed, Baher Abdulhai, and Mohamed El-Darieby (2018). “Towards a service-oriented cyber–physical systems of systems for smart city mobility applications”. In: *Future Generation Computer Systems* 79, pp. 575–587.

Englund, Cristofer et al. (2016). “The grand cooperative driving challenge 2016: boosting the introduction of cooperative automated vehicles”. In: *IEEE Wireless Communications* 23.4, pp. 146–152.

Estefan, Jeff A et al. (2007). “Survey of model-based systems engineering (MBSE) methodologies”. In: *Incose MBSE Focus Group* 25.8, pp. 1–12.

Fang, Zhemei, Navindran Davendralingam, and Daniel DeLaurentis (2018). “Multi-stakeholder Dynamic Optimization for Acknowledged System-of-Systems Architecture Selection”. In: *IEEE Systems Journal* 99, pp. 1–12.

Farcas, Claudiu et al. (2010). “Addressing the integration challenge for avionics and automotive systems—From components to rich services”. In: *Proceedings of the IEEE* 98.4, pp. 562–583.

Farroha, Deborah L and Bassam S Farroha (2011). “Agile development for system of systems: Cyber security integration into information repositories architecture”. In: *IEEE International on Systems Conference (SysCon)*, pp. 182–188.

Figueiredo, L. et al. (2001). “Towards the development of intelligent transportation systems”. In: *2001 IEEE Intelligent Transportation Systems. Proceedings*, pp. 1206–1211.

Findeisen, Wladyslaw et al. (1980). *Control and coordination in hierarchical systems.* John Wiley & Sons.

Furrer, Frank J. (2017). *Architecture for Cyber-Physical Systems-of-Systems.* `http://st.inf.tu-dresden.de/files/teaching/ws17/ring/TUD_Ringvorlesung-WS1718_20180108-V11.pdf`. [Online]. Presented at WS Conference, Technical University of Dredsen.

Gao, Hui et al. (2015). “A Survey of Incentive Mechanisms for Participatory Sensing.” In: *IEEE Communications Surveys and Tutorials* 17.2, pp. 918–943.

Garnier, Jean-Luc (2018). *Investigating in SoS Taxonomies to improve Systems Engineering.* `http://sosengineering.org/2018/wp-content/uploads/2014/07/Jean-Luc-Garnier-180620-IEEE-SoSE-SoS-Taxonomies-V0.2.pdf`. [Online]. Presented at the 14th IEEE Systems of Systems Engineering Conference, Paris, France.

Gorod, Alex, Ryan Gove, et al. (2007). “System of systems management: A network management approach”. In: *IEEE International Conference on System of Systems Engineering (SoSE)*, pp. 1–5.

Gorod, Alex, Brian Sauser, and John Boardman (Dec. 2008). *System-of-Systems Engineering Management: A Review of Modern History and a Path Forward.*

Gorod, Alex, Brian E White, et al. (2014). *Case studies in system of systems, enterprise systems, and complex systems engineering.* CRC Press.

Grigoroudis, E., V. S. Kouikoglou, and Y. A. Phillis (June 2012). “A System-of-Systems approach for improving healthcare systems”. In: *World Automation Congress*, pp. 1–6.

Guériau, Maxime et al. (2016). “How to assess the benefits of connected vehicles? A simulation framework for the design of cooperative traffic management strategies”. In: *Transportation research part C: emerging technologies* 67, pp. 266–279.

Handley, Holly AH and Robert J Smillie (2008). “Architecture framework human view: The NATO approach”. In: *Systems Engineering* 11.2, pp. 156–164.

Haren, Van (2011). *TOGAF Version 9.1.* Van Haren Publishing.

Hashem, Ibrahim Abaker Targio et al. (2015). “The rise of “big data” on cloud computing: Review and open research issues”. In: *Information systems* 47, pp. 98–115.

Hayward, J (1971). *Near misses as a measure of safety at urban intersections.* Pennsylvania Transportation and Traffic Safety Center.

Henshaw, Michael et al. (2013). “Systems of systems engineering: a research imperative”. In: *International Conference on System Science and Engineering (ICSSE)*, pp. 389–394.

Heymans, Patrick and Eric Dubois (Mar. 1998). “Scenario-Based Techniques for Supporting the Elaboration and the Validation of Formal Requirements”. In: *Requirements Engineering* 3.3, pp. 202–218.

Hidas, Peter (2002). “Modelling lane changing and merging in microscopic traffic simulation”. In: *Transportation Research Part C: Emerging Technologies* 10.5-6, pp. 351–371.

Holland, JH (1995). “Hidden order: how adaptation builds complexity”. In: *Reading, MA: Perseus.*

Holt, Jon, Simon A Perry, and Mike Brownsword (2011). *Model-based Requirements Engineering.* The Institution of Engineering and Technology.

Holt, Jon, Simon Perry, et al. (2015). “A Model-Based Approach for Requirements Engineering for Systems of Systems.” In: *IEEE Systems Journal* 9.1, pp. 252–262.

Hou, Zhicheng and Isabelle Fantoni (2015). “Distributed leader-follower formation control for multiple quadrotors with weighted topology”. In: *10th System of Systems Engineering Conference (SoSE)*, pp. 256–261.

INCOSE (2015). *INCOSE Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities, 4th Edition.* Hoboken, NJ, USA: Wiley.

ISO, May (2011). *Systems and software engineering–architecture description.* Tech. rep. ISO/IEC/IEEE 42010.

ITS, ETSI (2008). *Standards on the move.* [Online]. Presented at the 7th European Congress on ITS, Geneva.

Jackson, Michael C and Paul Keys (1984). “Towards a system of systems methodologies”. In: *Journal of the operational research society* 35.6, pp. 473–486.

Jacob, François (1974). *The logic of living systems: a history of heredity.* Lane.

Jamshidi, Mo (1997). *Large-scale systems: Modeling, control and fuzzy logic.* Prentice Hall.

— (2008). *Systems of systems engineering: principles and applications.* CRC press.

Jamshidi, Mohammad (1996). *Large-scale systems: modeling, control, and fuzzy logic.* Prentice-Hall, Inc.

Jaramillo, Juan José and R Srikant (2010). “A game theory based reputation mechanism to incentivize cooperation in wireless ad hoc networks”. In: *Ad Hoc Networks* 8.4, pp. 416–429.

Jazar, Reza N (2017). *Vehicle dynamics: theory and application.* Springer.

Kariwala, Vinay (2007). “Fundamental limitation on achievable decentralized performance”. In: *Automatica* 43.10, pp. 1849–1854.

Kaslow, David et al. (2015). “Developing a cubesat model-based system engineering (mbse) reference model-interim status”. In: *IEEE Aerospace Conference*, pp. 1–16.

Kewley Jr, Robert H and Andreas Tolk (2009). “A systems engineering process for development of federated simulations”. In: *Proceedings of the 2009 Spring Simulation Multiconference*, p. 68.

Kilian, Christopher T (2006). *Modern control technology: components and systems*. Delmar/Thomson Learning.

Kotov, Vadim (1997). *Systems of systems as communicating structures*. Vol. 119. Hewlett Packard Laboratories.

Laureshyn, Aliaksei, Åse Svensson, and Christer Hydén (2010). “Evaluation of traffic safety, based on micro-level behavioural data: Theoretical framework and first implementation”. In: *Accident Analysis & Prevention* 42.6, pp. 1637–1646.

Lee, Jay, Behrad Bagheri, and Hung-An Kao (2015). “A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems”. In: *Manufacturing Letters* 3, pp. 18–23.

Lewis, Grace A et al. (2009). “Requirements engineering for systems of systems”. In: *3rd Annual IEEE Systems Conference*, pp. 247–252.

Liu, S. (June 2011). “Employing System of Systems Engineering in China’s Emergency Management”. In: *IEEE Systems Journal* 5.2, pp. 298–308.

Lu, Xiaonan et al. (2014). “Hierarchical control of parallel AC-DC converter interfaces for hybrid microgrids”. In: *IEEE Transactions on Smart Grid* 5.2, pp. 683–692.

Luo, Yugong et al. (2016). “A dynamic automated lane change maneuver based on vehicle-to-vehicle communication”. In: *Transportation Research Part C: Emerging Technologies* 62, pp. 87–102.

Luzeaux, Dominique and Jean-René Ruault (2010). *Systems of systems: concepts, illustrations, standards and methods*. John Wiley & Sons.

Luzeaux, Dominique, Jean-Rene Ruault, and Jean-Luc Wippler (2013). *Large-scale Complex System and Systems of Systems*. John Wiley & Sons.

Macki, Jack and Aaron Strauss (2012). *Introduction to optimal control theory*. Springer Science & Business Media.

Mahmud, Nasif and A Zahedi (2016). “Review of control strategies for voltage regulation of the smart distribution network with high penetration of renewable distributed generation”. In: *Renewable and Sustainable Energy Reviews* 64, pp. 582–595.

Maier, Mark W. (1996). “Architecting Principles for Systems-of-Systems”. In: *INCOSE International Symposium* 6.1, pp. 565–573.

— (1998). “Architecting principles for systems-of-systems”. In: *Systems Engineering: The Journal of the International Council on Systems Engineering* 1.4, pp. 267–284.

— (Oct. 2005). “Research Challenges for Systems-of-Systems”. In: *IEEE International Conference on Systems, Man and Cybernetics* 4, pp. 3149–3154.

— (2006). “Disentangling modeling, architectures, and architecture descriptions”. In: *INSIGHT* 8.2, pp. 24–25.

Manthorpe, William HJ (1996). “The emerging joint system of systems: A systems engineering challenge and opportunity for APL”. In: *Johns Hopkins APL Technical Digest* 17.3, p. 305.

Marvasti, A. K. et al. (May 2014). “Optimal Operation of Active Distribution Grids: A System of Systems Framework”. In: *IEEE Transactions on Smart Grid* 5.3, pp. 1228–1237.

Mauss, Fredrick et al. (2015). “System of Systems Approaches for Mobile Source Transit Security”. In: *INCOSE International Symposium* 25.1, pp. 1278–1289.

McGehee, Daniel V, Elizabeth N Mazzae, and GH Scott Baldwin (2000). “Driver reaction time in crash avoidance research: validation of a driving simulator study on a test track”. In: *Proceedings of the human factors and ergonomics society annual meeting* 44.20, pp. 3–320.

Mehta, Ruta and Vijay V Vazirani (2018). “An incentive compatible, efficient market for air traffic flow management”. In: *Theoretical Computer Science*.

Mesarovic, Mihajlo D, D Macko, and Yasuhiko Takahara (1970). “Theory of multi-level hierarchical systems”. In: *New York, Academic*.

Mostafavi, A. et al. (June 2011). “Exploring the Dimensions of Systems of Innovation Analysis: A System of Systems Framework”. In: *IEEE Systems Journal* 5.2, pp. 256–265.

Murgovski, Nikolce and Jonas Sjöberg (2015). “Predictive cruise control with autonomous overtaking”. In: *IEEE 54th Annual Conference on Decision and Control (CDC)*, pp. 644–649.

NASA (2016). *Systems Engineering Handbook - Rev 2*. URL: `https://www.nasa.gov/sites/default/files/atoms/files/nasa_systems_engineering_handbook_0.pdf`.

Nguyen, Ngoc Anh et al. (2017). “Autonomous overtaking using stochastic model predictive control”. In: *11th Asian Control Conference (ASCC)*, pp. 1005–1010.

Nie, Jianqiang et al. (2016). “Decentralized cooperative lane-changing decision-making for connected autonomous vehicles”. In: *IEEE Access* 4, pp. 9413–9420.

Officer, DoD Deputy Chief Information (2010). *United states department of defense architecture framework (DoDAF)*. Technical Report Version 2.02.

Olfati-Saber, Reza, J Alex Fax, and Richard M Murray (2007). “Consensus and cooperation in networked multi-agent systems”. In: *Proceedings of the IEEE* 95.1, pp. 215–233.

Ota, Daniel and Michael Gerz (2011). *Benefits and challenges of architecture frameworks*. Tech. rep. Fraunhofer-FKIE Wachtberg (Germany).

Otto, A. et al. (June 2016). “A Quantified System-of-Systems Modeling Framework for Robust National Infrastructure Planning”. In: *IEEE Systems Journal* 10.2, pp. 385–396.

Ouyang, Min (2014). “Review on modeling and simulation of interdependent critical infrastructure systems”. In: *Reliability engineering & System safety* 121, pp. 43–60.

Owens, William A (1996). *The emerging US system-of-systems*. Tech. rep. National Defense Univ. Washington DC. Inst. For National Strategic Studies.

Palizban, Omid, Kimmo Kauhaniemi, and Josep M Guerrero (2014). “Microgrids in active network management—Part I: Hierarchical control, energy storage, virtual power plants, and market participation”. In: *Renewable and Sustainable Energy Reviews* 36, pp. 428–439.

Partners, MODAF (2005). *British ministry of defense architecture framework (modaf): Technical handbook*. Technical Report Version 1.0.

Pei, Richard S (2000). “System of Systems Integration (SoSI)-A” SMART” Way of Acquiring Army C412WS Systems”. In: *Summer Computer Simulation Conference*, pp. 574–579.

Petrov, Plamen and Fawzi Nashashibi (2014). “Modeling and Nonlinear Adaptive Control for Autonomous Vehicle Overtaking.” In: *IEEE Trans. Intelligent Transportation Systems* 15.4, pp. 1643–1656.

Planning, U.S. JPDO Joint and Development Office (2007). *Concept of operations for the next generation air transportation system.* `http://www.dtic.mil/dtic/tr/fulltext/u2/a535795.pdf`. [Online]. Joint Planning and Development Office, Washington, DC.

Rainey, Larry B and Andreas Tolk (2015). *Modeling and simulation support for system of systems engineering applications.* John Wiley & Sons.

Ray, Anjan Kumar et al. (2009). “Decentralized motion coordination for a formation of rovers”. In: *IEEE Systems Journal* 3.3, pp. 369–381.

Ren, Wei (2008). “Synchronization of coupled harmonic oscillators with local interaction”. In: *Automatica* 44.12, pp. 3195–3200.

Ren, Wei, Randal W Beard, and Ella M Atkins (2007). “Information consensus in multivehicle cooperative control”. In: *IEEE Control Systems* 27.2, pp. 71–82.

Rezaei, E Mohammad Hadi and Parviz Jabehdar-Maralani (2012). “Two-level hierarchical optimal control based on interaction principle for large scale systems”. In: *20th Iranian Conference on Electrical Engineering (ICEE)*, pp. 828–833.

Ruiz-Romero, Salvador et al. (2014). “Integration of distributed generation in the power distribution network: The need for smart grid control systems, communication and equipment for a smart city—Use cases”. In: *Renewable and sustainable energy reviews* 38, pp. 223–234.

Sage, Andrew P (2011). *System of systems engineering: innovations for the 21st century.* Vol. 58. John Wiley & Sons.

Sanduka, Imad and Roman Obermaisser (2014). “Model-based development of systems-of-systems with real-time requirements”. In: *12th IEEE International Conference on Industrial Informatics (INDIN)*, pp. 188–194.

Satzoda, Ravi Kumar and Mohan M Trivedi (2014). “Overtaking & receding vehicle detection for driver assistance and naturalistic driving studies”. In: *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 697–702.

Scattolini, Riccardo (2009). “Architectures for distributed and hierarchical model predictive control–a review”. In: *Journal of process control* 19.5, pp. 723–731.

Shams, Fereidoon et al. (2008). “A service driven development process (SDDP) model for ultra large scale systems”. In: *2nd international workshop on Ultra-large-scale software-intensive systems*, pp. 37–40.

Shenhar, Aaron J. (1994). “2.5.1 A New Systems Engineering Taxonomy”. In: *INCOSE International Symposium* 5.1, pp. 723–732.

Shtessel, Yuri et al. (2014). *Sliding Mode Control and Observation.* Springer, pp. 105–141.

Siljak, Dragoslav D (2011). *Decentralized control of complex systems.* Courier Corporation.

Šiljak, Dragoslav D and AI Zečević (2005). “Control of large-scale systems: Beyond decentralized feedback”. In: *Annual Reviews in Control* 29.2, pp. 169–179.

Su, BB et al. (2007). “A game theory model of urban public traffic networks”. In: *Physica A: Statistical Mechanics and its Applications* 379.1, pp. 291–297.

Swan, Melanie (2015). *Blockchain: Blueprint for a new economy.* O’Reilly Media, Inc.

Turban, Efraim, Ramesh Sharda, and Dursun Delen (2010). *Decision Support and Business Intelligence Systems*. 9th. Upper Saddle River, NJ, USA: Prentice Hall Press.

U.S. DoD, The United States Department of Defense (2008). *Systems Engineering Guide for Systems of Systems*. `https://www.acq.osd.mil/se/docs/se-guide-for-sos.pdf`. [Online].

Vander Werf, Joel et al. (2002). "Effects of adaptive cruise control systems on highway traffic flow capacity". In: *Transportation Research Record: Journal of the Transportation Research Board* 1800, pp. 78–84.

Vandermeulen, Isaac, Martin Guay, and P James McLellan (2018). "Distributed control of high-altitude balloon formation by extremum-seeking control". In: *IEEE Transactions on Control Systems Technology* 26.3, pp. 857–873.

Vargas, Alix (2016). "Decision-Making System and Operational Risk Framework for Hierarchical Production Planning". In: *Journal of Control Engineering and Applied Informatics* 18.3, pp. 72–81.

Vermillion, Chris et al. (2014). "Model-based plant design and hierarchical control of a prototype lighter-than-air wind energy system, with experimental flight test results". In: *IEEE Transactions on Control Systems Technology* 22.2, pp. 531–542.

Wagenhals, Lee W and Alexander H Levis (2009). "Service oriented architectures, the DoD architecture framework 1.5, and executable architectures". In: *Systems Engineering* 12.4, pp. 312–343.

Wang, Renzhong and Cihan H Dagli (2011). "Executable system architecting using systems modeling language in conjunction with colored Petri nets in a model-driven systems development process". In: *Systems Engineering* 14.4, pp. 383–409.

Weik, Norman, Nora Niebel, and Nils Nießen (2016). "Capacity analysis of railway lines in Germany – A rigorous discussion of the queueing based approach". In: *Journal of Rail Transport Planning & Management* 6.2, pp. 99–115.

Whittington, P. and H. Dogan (June 2016). "SmartDisability: A smart system of systems approach to disability". In: *11th System of Systems Engineering Conference (SoSE)*, pp. 1–6.

Wu, Yen-Chun Jim and Pi-Ju Lee (2007). "The use of patent analysis in assessing ITS innovations: US, Europe and Japan". In: *Transportation Research Part A: Policy and Practice* 41.6, pp. 568–586.

Xin, Huanhai et al. (2016). "A Decentralized Hierarchical Control Structure and Self-Optimizing Control Strategy for FP Type DGs in Islanded Microgrids." In: *IEEE Trans. Smart Grid* 7.1, pp. 3–5.

Xu, Philippe et al. (2018). "System Architecture of a Driverless Electric Car in the Grand Cooperative Driving Challenge". In: *IEEE Intelligent Transportation Systems Magazine* 10.1, pp. 47–59.

Yuan, Wang and Zhijun Li (2017). "Navigation and collision avoidance for nonholonomic robots using quadrupole potential function". In: *2nd International Conference on Advanced Robotics and Mechatronics (ICARM)*, pp. 47–52.

Zhang, Ying et al. (2012). "A Service-Oriented Method for System-of-Systems Requirements Analysis and Architecture Design." In: *Journal of Software* 7.2, pp. 358–365.

Zheng, Zuduo (2014). “Recent developments and research needs in modeling lane changing”. In: *Transportation research part B: methodological* 60, pp. 16–32.

Zhou, Zhongbao et al. (2017). “Stochastic network DEA models for two-stage systems under the centralized control organization mechanism”. In: *Computers & Industrial Engineering* 110, pp. 404–412.

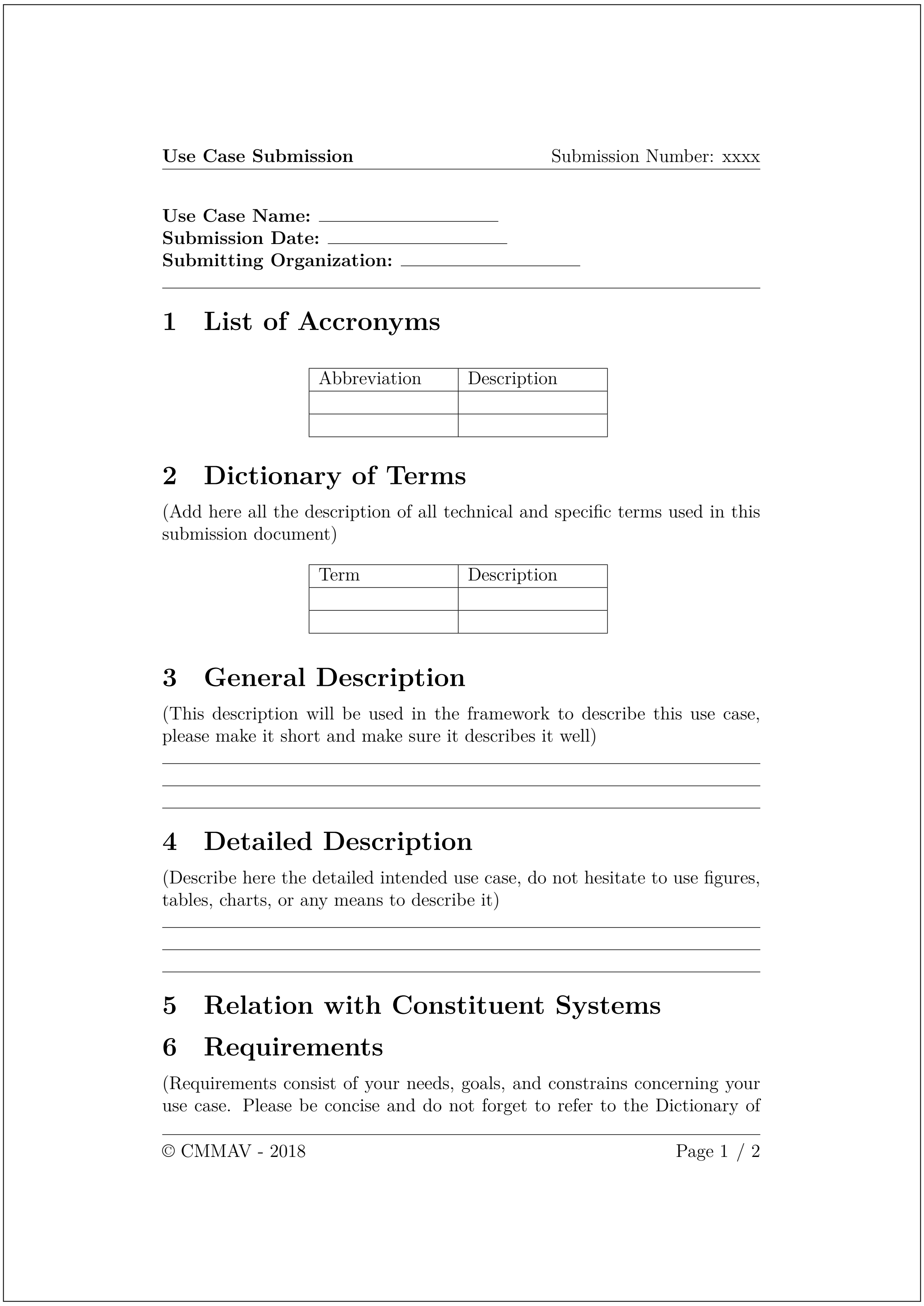
Appendix A

Appendix

A.1 Overtaking Use Case submission document

A.1.1 Template

This is a template of the use case submission document. Any user who has a use case that is not treated in the CMMAV or that has special requirements that are not met by it may use this document to submit their specific use case to CMMAV maintainers.



Use Case Submission Submission Number: xxxx

Use Case Name: ____________________
Submission Date: ____________________
Submitting Organization: ____________________

1 List of Accronyms

Abbreviation	Description

2 Dictionary of Terms

(Add here all the description of all technical and specific terms used in this submission document)

Term	Description

3 General Description

(This description will be used in the framework to describe this use case, please make it short and make sure it describes it well)

4 Detailed Description

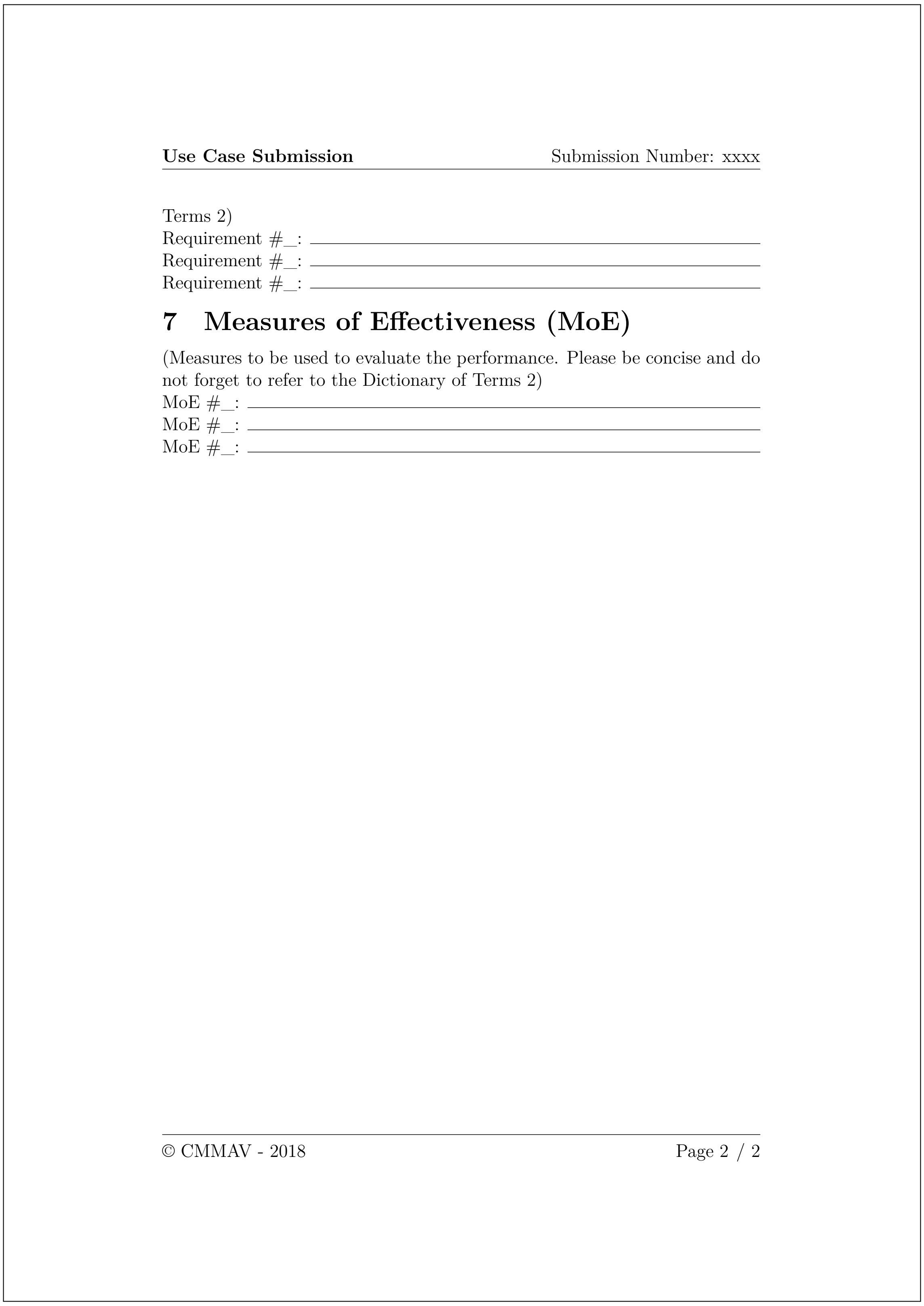
(Describe here the detailed intended use case, do not hesitate to use figures, tables, charts, or any means to describe it)

5 Relation with Constituent Systems

6 Requirements

(Requirements consist of your needs, goals, and constrains concerning your use case. Please be concise and do not forget to refer to the Dictionary of

© CMMAV - 2018 Page 1 / 2



Use Case Submission Submission Number: xxxx

Terms 2)
Requirement #_: ________________________________
Requirement #_: ________________________________
Requirement #_: ________________________________

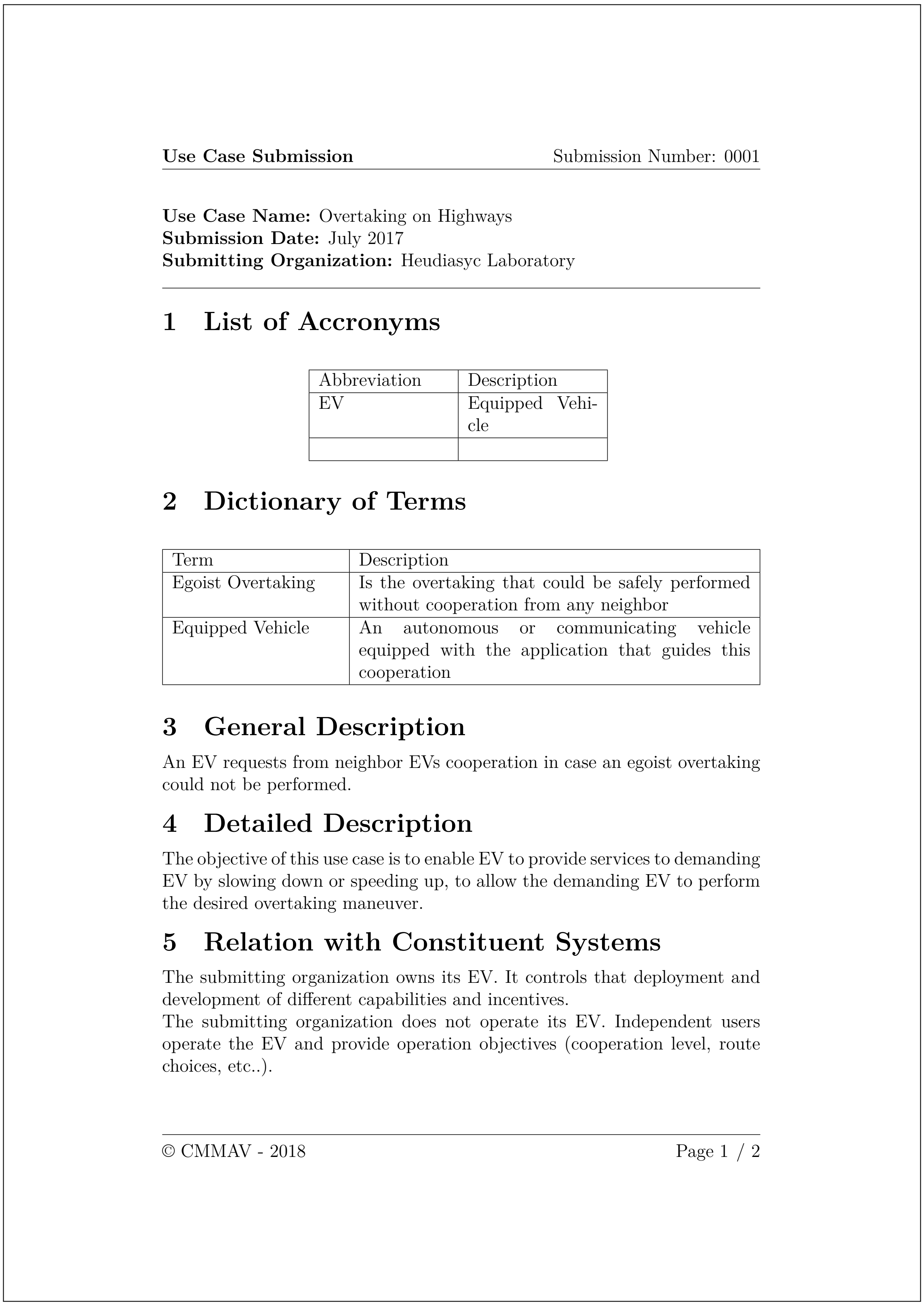
7 Measures of Effectiveness (MoE)

(Measures to be used to evaluate the performance. Please be concise and do not forget to refer to the Dictionary of Terms 2)
MoE #_: ________________________________
MoE #_: ________________________________
MoE #_: ________________________________

© CMMAV - 2018 Page 2 / 2

A.1.2 Overtaking on Highways use case submission

This is a the overtaking use case document submitted by Heudiasyc laboratory to the CMMAV.



Use Case Submission Submission Number: 0001

Use Case Name: Overtaking on Highways
Submission Date: July 2017
Submitting Organization: Heudiasyc Laboratory

1 List of Accronyms

Abbreviation	Description
EV	Equipped Vehicle

2 Dictionary of Terms

Term	Description
Egoist Overtaking	Is the overtaking that could be safely performed without cooperation from any neighbor
Equipped Vehicle	An autonomous or communicating vehicle equipped with the application that guides this cooperation

3 General Description

An EV requests from neighbor EVs cooperation in case an egoist overtaking could not be performed.

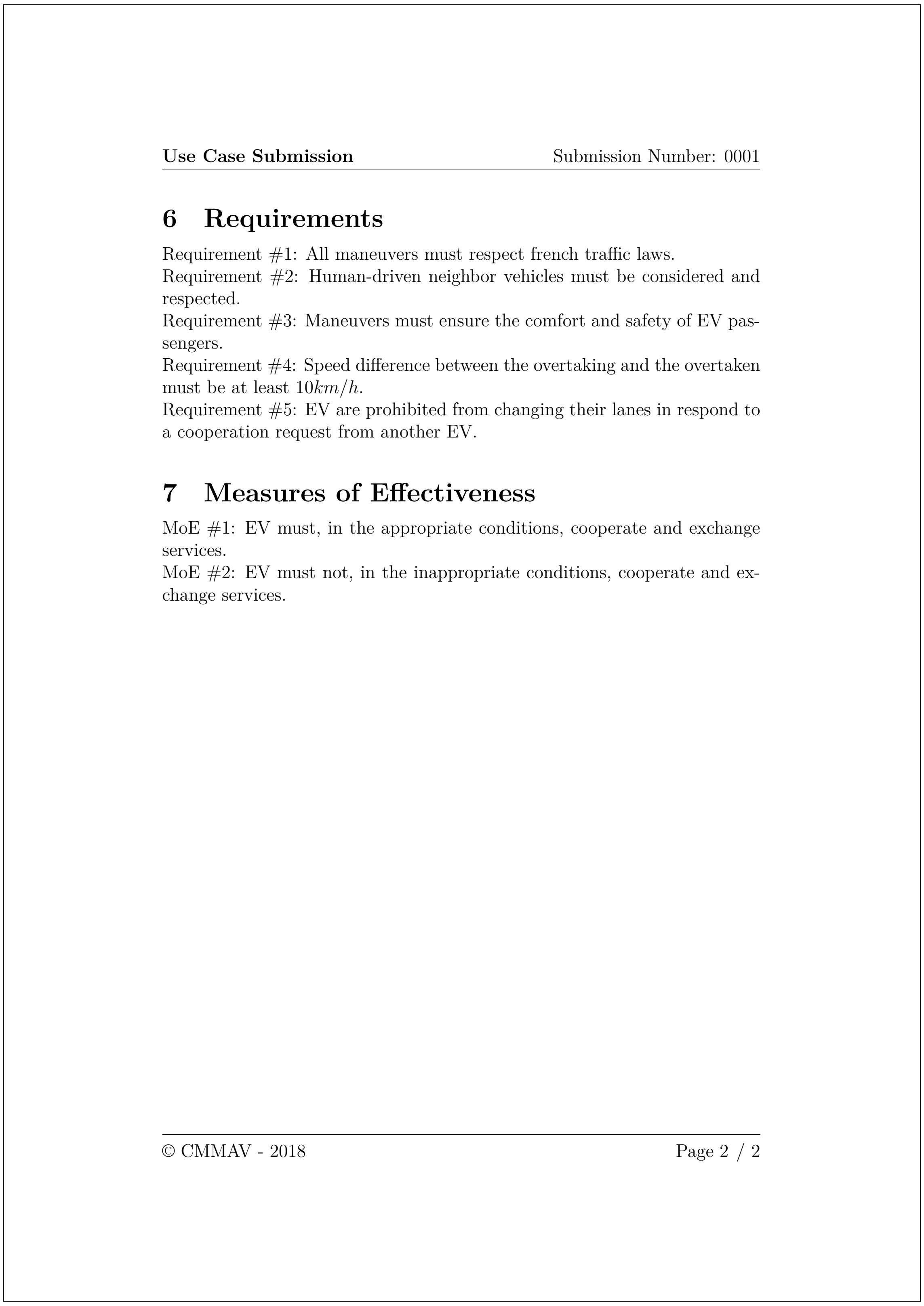
4 Detailed Description

The objective of this use case is to enable EV to provide services to demanding EV by slowing down or speeding up, to allow the demanding EV to perform the desired overtaking maneuver.

5 Relation with Constituent Systems

The submitting organization owns its EV. It controls that deployment and development of different capabilities and incentives.
The submitting organization does not operate its EV. Independent users operate the EV and provide operation objectives (cooperation level, route choices, etc..).

© CMMAV - 2018 Page 1 / 2



6 Requirements

Requirement #1: All maneuvers must respect french traffic laws.
Requirement #2: Human-driven neighbor vehicles must be considered and respected.
Requirement #3: Maneuvers must ensure the comfort and safety of EV passengers.
Requirement #4: Speed difference between the overtaking and the overtaken must be at least 10km/h.
Requirement #5: EV are prohibited from changing their lanes in respond to a cooperation request from another EV.

7 Measures of Effectiveness

MoE #1: EV must, in the appropriate conditions, cooperate and exchange services.
MoE #2: EV must not, in the inappropriate conditions, cooperate and exchange services.

A.2 CMMAV Framework Views

A.2.1 Organizational View (Fig. A.1)

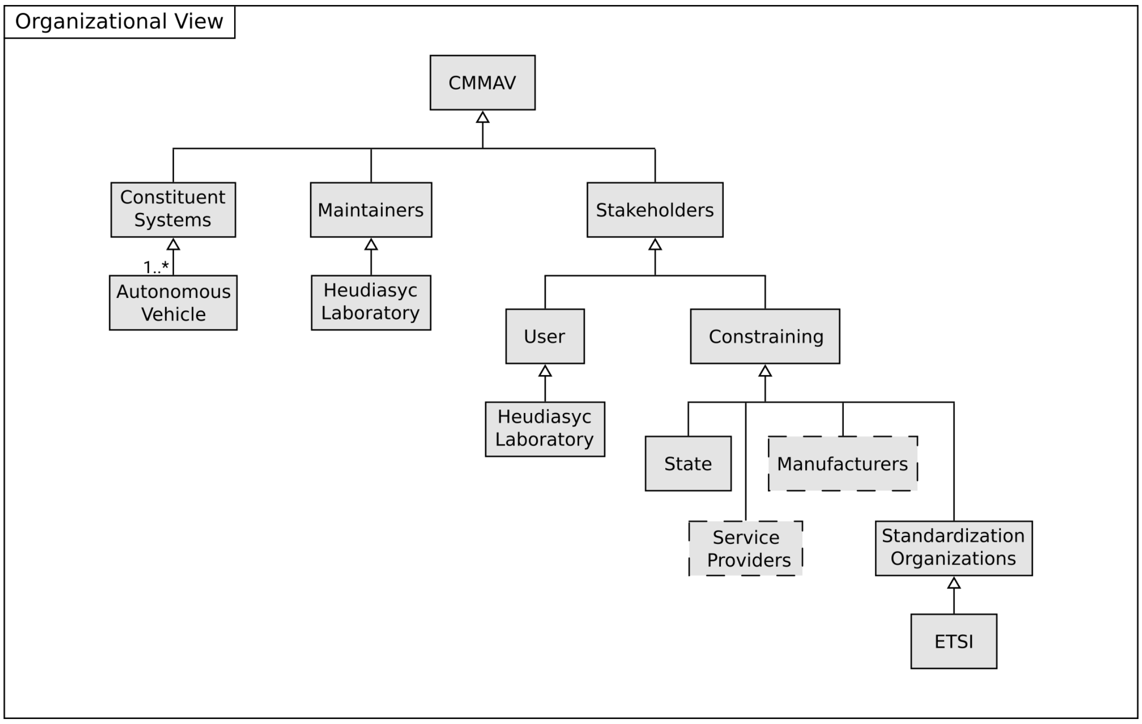


Figure A.1: Organizational View: Maintainers, Stakeholders, and Constituents.

A.2.2 Sources View (Fig. A.2)

A.2.3 Requirements View (Fig. A.3)

A.2.4 Capabilities View (Fig. A.4)

A.2.5 Validation View (Fig. A.5)

A.3 CMMAV Framework - Detailed Forms

A.3.1 Organizational View Forms (Fig. A.6)

A.3.2 Sources View Forms (Fig. A.7)

A.3.3 Requirements View Forms (Fig. A.8, A.9)

A.3.4 Capabilities View Forms (Fig. A.10)

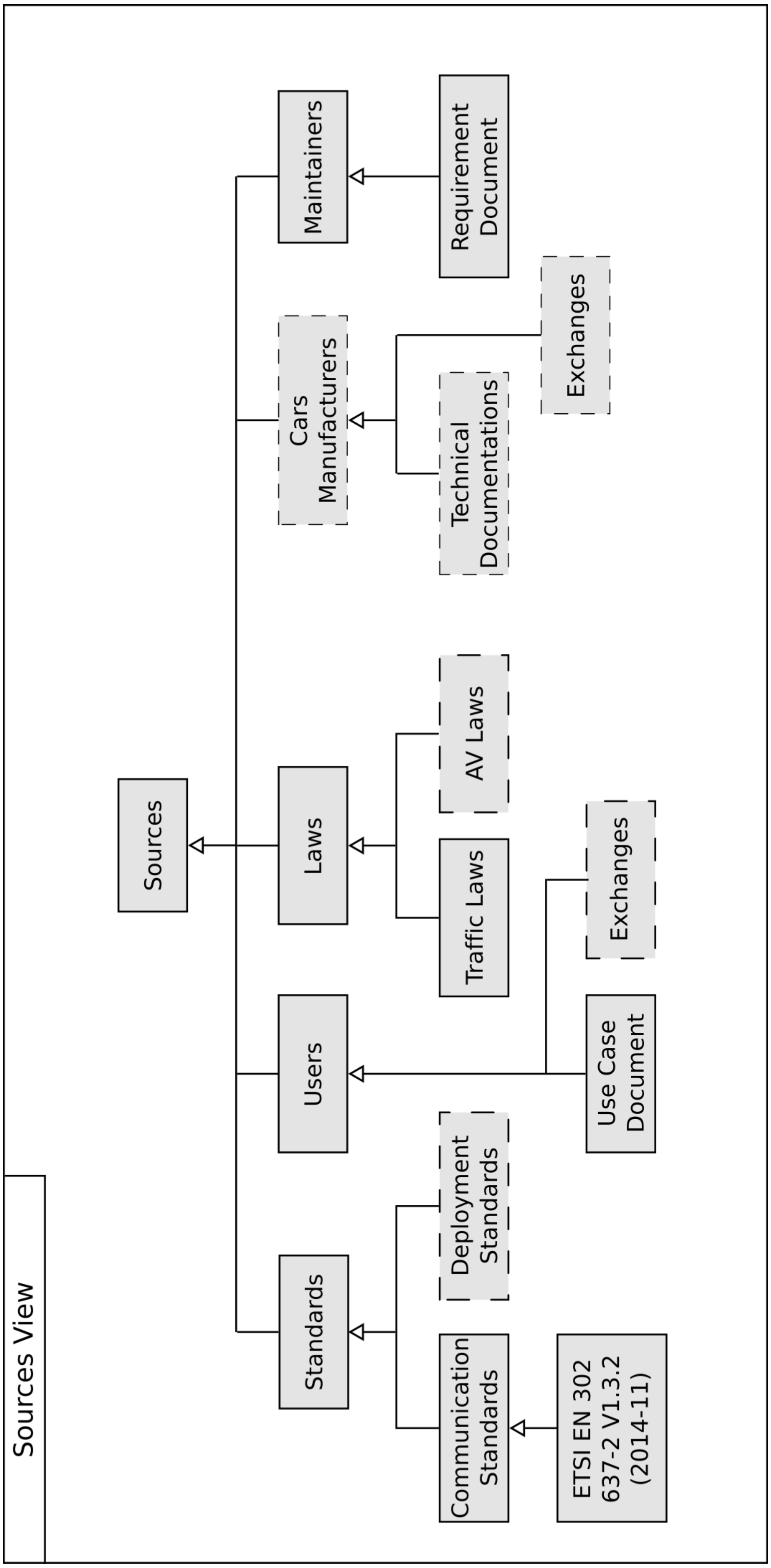


Figure A.2: Sources View: Every stakeholder provides sources for its requirements.

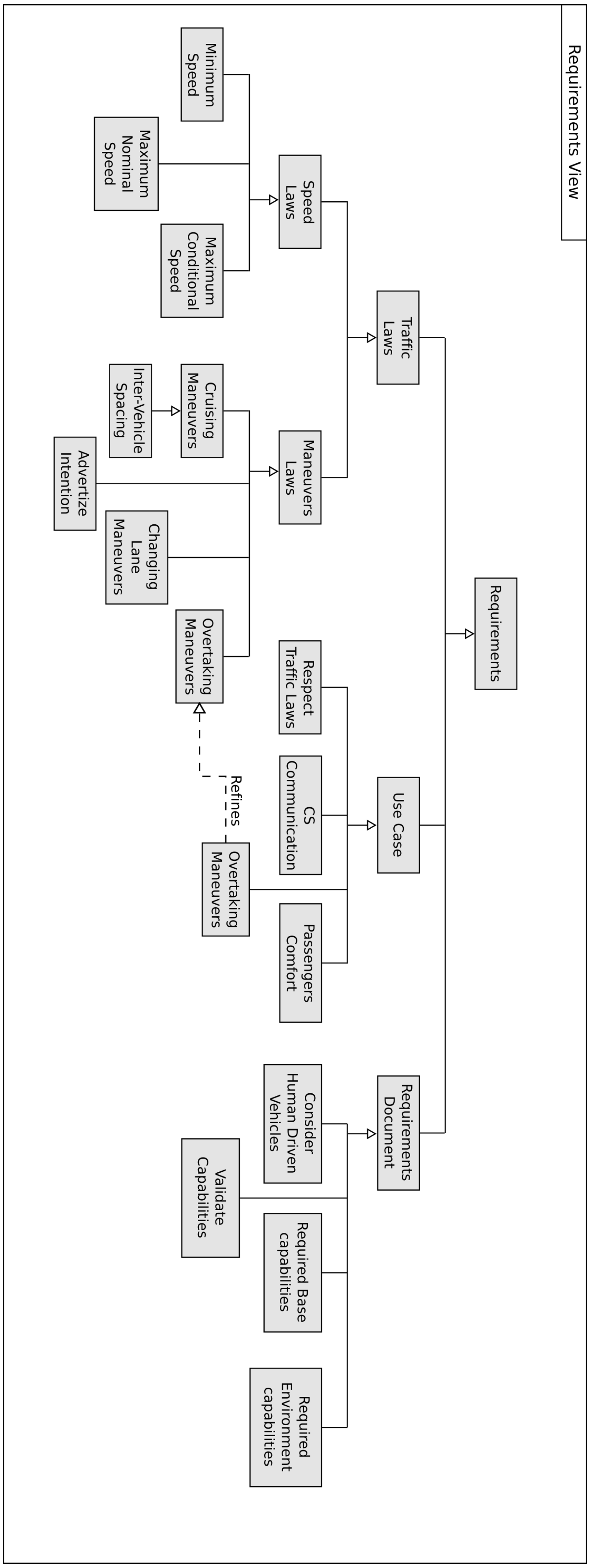


Figure A.3: Requirements View: different requirements from different sources.

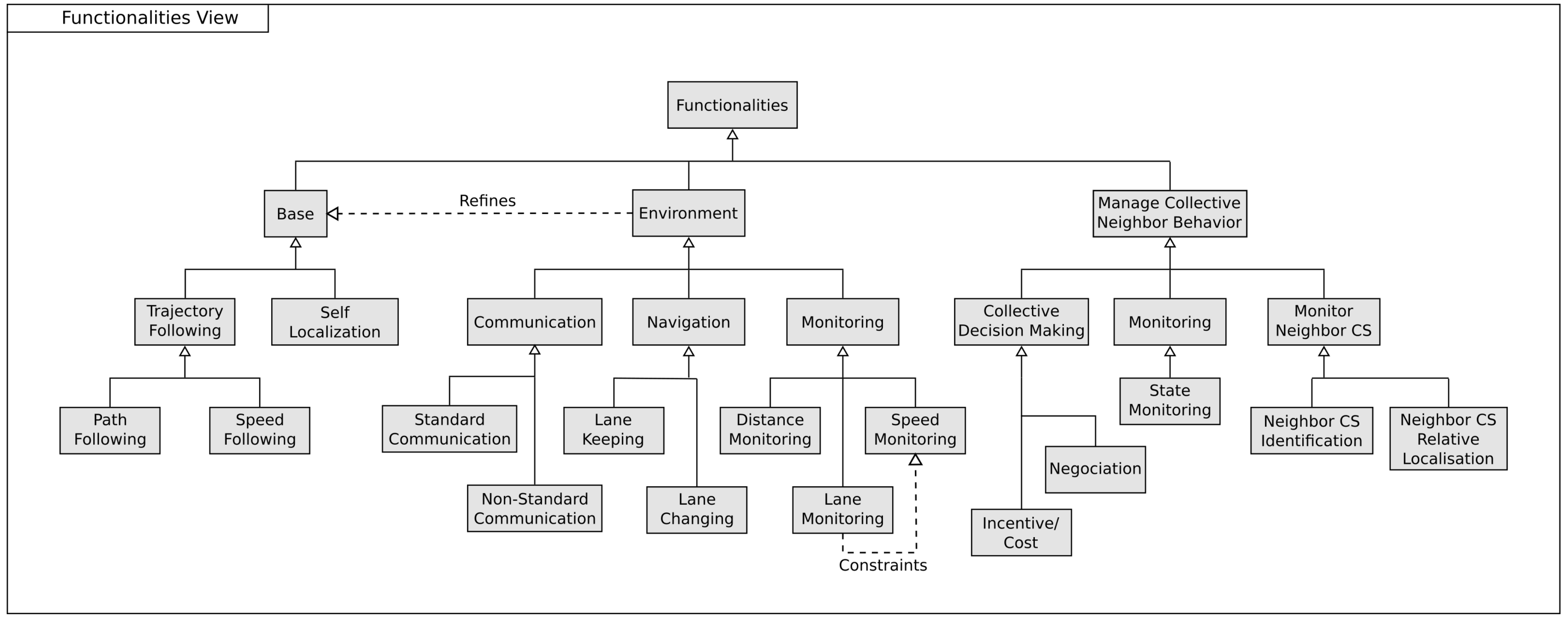


Figure A.4: Capabilities View: base, environment, and collective capabilities.

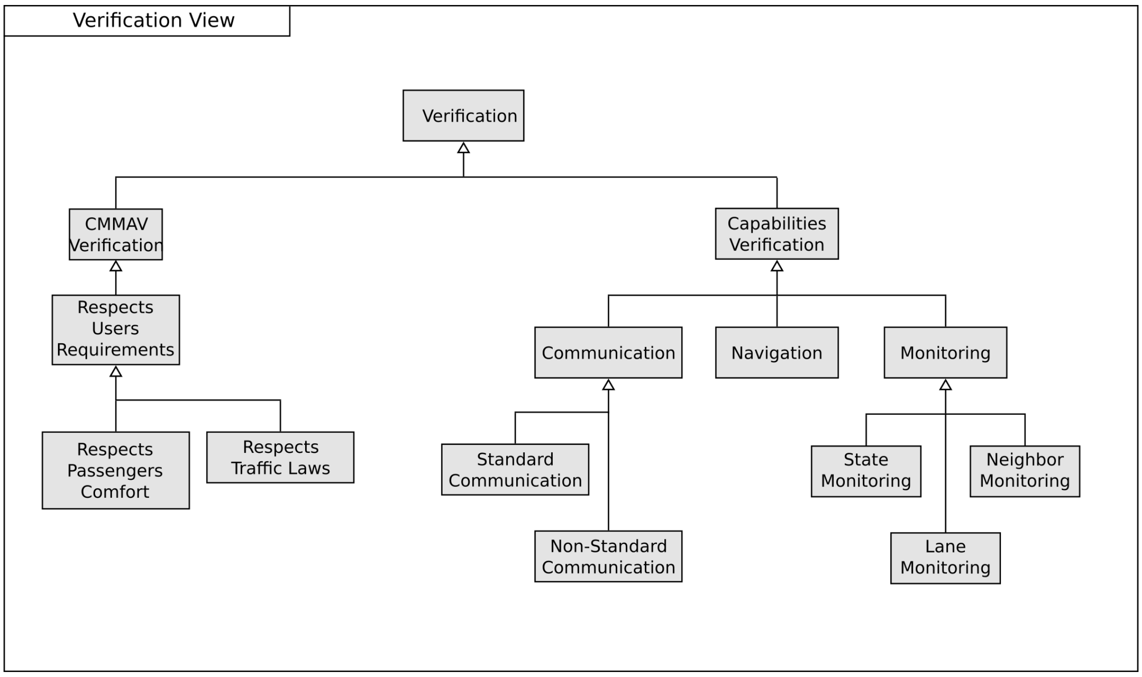


Figure A.5: Validation View

Actor	UID: A001
Name:	Heudiasyc Laboratory
Type:	Maintaining Organization
Related Use Cases:	UC001

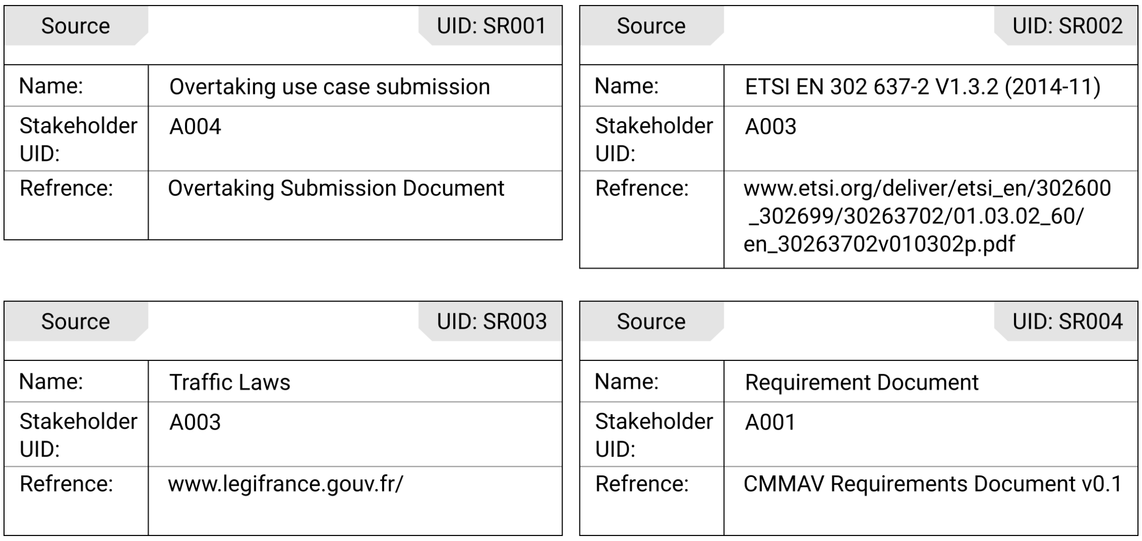
Actor	UID: A002
Name:	State
Type:	Constraining Stakeholder
Related Use Cases:	UC001

Actor	UID: A003
Name:	ETSI
Type:	Constraining Stakeholder
Related Use Cases:	UC001

Actor	UID: A004
Name:	Heudiasyc Laboratory
Type:	User Stakeholder
Related Use Cases:	UC001

Actor	UID: A005
Name:	Autonomous Vehicles
Type:	Constituent System
Related Use Cases:	UC001

Figure A.6: Organizational Forms



Source	UID: SR001
Name:	Overtaking use case submission
Stakeholder UID:	A004
Refrence:	Overtaking Submission Document

Source	UID: SR002
Name:	ETSI EN 302 637-2 V1.3.2 (2014-11)
Stakeholder UID:	A003
Refrence:	www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.03.02_60/en_30263702v010302p.pdf

Source	UID: SR003
Name:	Traffic Laws
Stakeholder UID:	A003
Refrence:	www.legifrance.gouv.fr/

Source	UID: SR004
Name:	Requirement Document
Stakeholder UID:	A001
Refrence:	CMMAV Requirements Document v0.1

Figure A.7: Requirements Sources Forms

Requirement	UID: RQ001
Name:	Minimum Speed
Source UID:	SR003
Authority:	Mandatory
Description:	[...] On the motorway, when the traffic is fluid and the weather conditions allow sufficient visibility and grip, drivers using the left-most lane can not travel at a speed below 80km/h. (Article R413-19)

Requirement	UID: RQ002
Name:	Maximum Nominal Speed
Source UID:	SR003
Authority:	Mandatory
Description:	Outside built-up areas, the speed of vehicles is limited to: 1. 130km/h on Highways; 2. 110km/h on dual carriageway roads separated by a median strip; 3. 90km/h on other roads. (Article R413-2.1)

Requirement	UID: RQ005
Name:	Advertize Intention
Source UID:	SR003
Authority:	Mandatory
Description:	Any driver who is about to make a change in the direction of his vehicle or to slow down its pace must inform other users of its intention, in particular when it is going to go to the left, cross the road, or when, after a stop or parking, he wants to resume his place in the flow of traffic. (Article 412-10)

Requirement	UID: RQ003
Name:	Maximum Conditional Speed
Source UID:	SR003
Authority:	Mandatory
Description:	In case of visibility less than 50 meters, the maximum speeds are lowered to 50km/h on all road and motorway networks. (Article R413-4)

Requirement	UID: RQ004
Name:	Inter-Vehicle Spacing
Source UID:	SR003
Authority:	Mandatory
Description:	When two vehicles are following each other, the driver of the second vehicle must maintain a sufficient safety distance [...] It corresponds to the distance traveled by the vehicle for a period of at least two seconds. (Article 412-19)

Requirement	UID: RQ006
Name:	Overtaken Vehicle
Source UID:	SR003
Authority:	Mandatory
Description:	When they are about to be overtaken, drivers must immediately tighten to their right without accelerating pace. (Article 414-10)

Requirement	UID: RQ007
Name:	Required Base Capabilities
Source UID:	SR004
Authority:	Mandatory
Description:	CS msut have the following base capabilities: 1- Trajectory Following which consists of the abilitiy to follow a predefined path, with a predefined speed profile 2- Self localize, by having its accurate position and heading with a certain confidence.

Requirement	UID: RQ008
Name:	Required Environemnt Capabilities
Source UID:	SR004
Authority:	Mandatory
Description:	CS must have the following environment capabilities: 1- V2V communication 2- lane keeping and lane changing 3- lane monitoring, speed monitoring, and distance monitoring.

Requirement	UID: RQ009
Name:	Validate Capabilities
Source UID:	SR004
Authority:	Mandatory
Description:	Users must validate each capability by using the specific validation element for each capability

Requirement	UID: RQ010
Name:	Post Overtaking
Source UID:	SR003
Authority:	Mandatory
Description:	The overtaking is done on the left. (Article R414-6.I)

Requirement	UID: RQ011
Name:	Post Overtaking
Source UID:	SR003
Authority:	Mandatory
Description:	As an exception to REQ0010, all drivers must overtake on the right: A vehicle whose driver has signaled that he was preparing to change direction to the left. (Article R414-6.II-1)

Requirement	UID: RQ012
Name:	Overtaking Condition
Source UID:	SR003
Authority:	Mandatory
Description:	One may only overtake a vehicle if the three child requirements RQ013, RQ014, and RQ015 are met. (Article R414-6.II)

Figure A.8: Requirements Forms - part 1

Requirement	UID: RQ013
Name:	Overtaking Condition 1.1
Source UID:	SR003
Authority:	Mandatory
Description:	They are not themselves about to be overtaken. (Article R414-4.II-3)

Requirement	UID: RQ014
Name:	Overtaking Condition 1.2
Source UID:	SR003
Authority:	Mandatory
Description:	The relative speed of the two vehicles will allow the overtaking to take place in a sufficiently short time. (Article R414-4.II-2)

Requirement	UID: RQ015
Name:	Overtaking Condition 1.3
Source UID:	SR003
Authority:	Mandatory
Description:	It has the possibility to resume its place in the normal flow of traffic without hindering it.. (Article R414-4.II-1)

Requirement	UID: RQ016
Name:	Overtaking Condition 1.3
Source UID:	SR003
Authority:	Mandatory
Description:	It has the possibility to resume its place in the normal flow of traffic without hindering it.. (Article R414-4.II-1)

Requirement	UID: RQ017
Name:	Passengers Comfort
Source UID:	SR004
Authority:	Mandatory
Description:	All performed maneuvers shall assure passengers comfort. (RQ018 and RQ019)

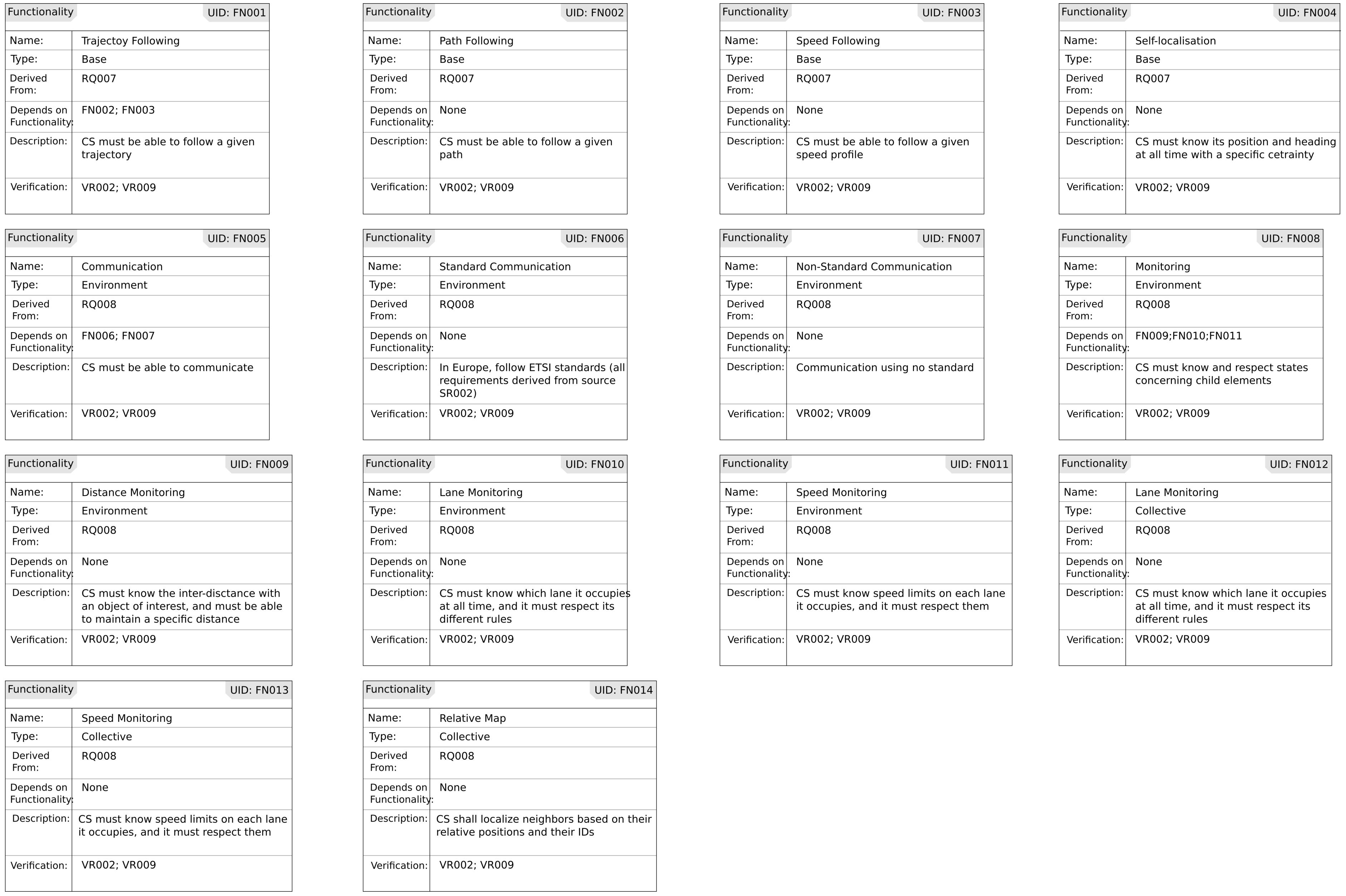
Requirement	UID: RQ018
Name:	Passengers Comfort 1.1
Source UID:	SR004
Authority:	Mandatory
Description:	Acceleration throughout any maneuver must respect the comfort norms ($-2m/s^2 \leq a \leq 2m/s^2$).

Requirement	UID: RQ019
Name:	Passengers Comfort 1.2
Source UID:	SR004
Authority:	Mandatory
Description:	Any change in steering angle must be bounded by min-max.

Requirement	UID: RQ020
Name:	Reach Destination Quickly
Source UID:	SR004
Authority:	Mandatory
Description:	CMMAV will try to minimize the time spent on the highway by using the maximum.allowed speed on lane when possible

Requirement	UID: RQ021
Name:	Passengers Comfort
Source UID:	SR004
Authority:	Mandatory
Description:	All performed maneuvers shall assure passengers comfort. (RQ018 and RQ019)

Figure A.9: Requirements Forms - part 2



Functionality	UID: FN001
Name:	Trajectoy Following
Type:	Base
Derived From:	RQ007
Depends on Functionality:	FN002; FN003
Description:	CS must be able to follow a given trajectory
Verification:	VR002; VR009

Functionality	UID: FN002
Name:	Path Following
Type:	Base
Derived From:	RQ007
Depends on Functionality:	None
Description:	CS must be able to follow a given path
Verification:	VR002; VR009

Functionality	UID: FN003
Name:	Speed Following
Type:	Base
Derived From:	RQ007
Depends on Functionality:	None
Description:	CS must be able to follow a given speed profile
Verification:	VR002; VR009

Functionality	UID: FN004
Name:	Self-localisation
Type:	Base
Derived From:	RQ007
Depends on Functionality:	None
Description:	CS must know its position and heading at all time with a specific cetrainty
Verification:	VR002; VR009

Functionality	UID: FN005
Name:	Communication
Type:	Environment
Derived From:	RQ008
Depends on Functionality:	FN006; FN007
Description:	CS must be able to communicate
Verification:	VR002; VR009

Functionality	UID: FN006
Name:	Standard Communication
Type:	Environment
Derived From:	RQ008
Depends on Functionality:	None
Description:	In Europe, follow ETSI standards (all requirements derived from source SR002)
Verification:	VR002; VR009

Functionality	UID: FN007
Name:	Non-Standard Communication
Type:	Environment
Derived From:	RQ008
Depends on Functionality:	None
Description:	Communication using no standard
Verification:	VR002; VR009

Functionality	UID: FN008
Name:	Monitoring
Type:	Environment
Derived From:	RQ008
Depends on Functionality:	FN009;FN010;FN011
Description:	CS must know and respect states concerning child elements
Verification:	VR002; VR009

Functionality	UID: FN009
Name:	Distance Monitoring
Type:	Environment
Derived From:	RQ008
Depends on Functionality:	None
Description:	CS must know the inter-disctance with an object of interest, and must be able to maintain a specific distance
Verification:	VR002; VR009

Functionality	UID: FN010
Name:	Lane Monitoring
Type:	Environment
Derived From:	RQ008
Depends on Functionality:	None
Description:	CS must know which lane it occupies at all time, and it must respect its different rules
Verification:	VR002; VR009

Functionality	UID: FN011
Name:	Speed Monitoring
Type:	Environment
Derived From:	RQ008
Depends on Functionality:	None
Description:	CS must know speed limits on each lane it occupies, and it must respect them
Verification:	VR002; VR009

Functionality	UID: FN012
Name:	Lane Monitoring
Type:	Collective
Derived From:	RQ008
Depends on Functionality:	None
Description:	CS must know which lane it occupies at all time, and it must respect its different rules
Verification:	VR002; VR009

Functionality	UID: FN013
Name:	Speed Monitoring
Type:	Collective
Derived From:	RQ008
Depends on Functionality:	None
Description:	CS must know speed limits on each lane it occupies, and it must respect them
Verification:	VR002; VR009

Functionality	UID: FN014
Name:	Relative Map
Type:	Collective
Derived From:	RQ008
Depends on Functionality:	None
Description:	CS shall localize neighbors based on their relative positions and their IDs
Verification:	VR002; VR009

Figure A.10: Capabilities Forms - part 1