Received: Added at production | Revised: Added at production | Accepted: Added at production

DOI: xxx/xxxx

ARTICLE TYPE

# A Model to Discipline Autonomy in Cyber-Physical Systems-of-Systems and its Application

Mohamad Gharib*[1] | Leandro Dias da Silva[2] | Andrea Ceccarelli[1]

[1]Department of Mathematics and Informatics, University of Florence, Florence, Italy
[2]Computing Institute, Universidade Federal de Alagoas, Alagoas, Brazil

Correspondence
*Mohamad Gharib, Email: mohamad.gharib@unifi.it

Present Address
Department of Mathematics and Informatics, University of Florence, Viale Morgagni 65, Florence, Italy

## Abstract

A Cyber-Physical System-of-Systems (CPSoS) can be defined as a System-of-Systems (SoS), composed of a number of operable and autonomous Constituent Systems (CSs) that are themselves Cyber-Physical Systems (CPSs). A main challenge in integrating CPSoS to function as a single integrated system is the autonomy of its components, which may result in conflicts due to the lack of coordination among its CPSs. In this paper, we advocate that in order to facilitate the integration of CPSs within the overall context of their CPSoS, we may need to adjust their level of autonomy in a way that enables them to coordinate their activities to avoid any conflict among one another. Reducing such conflicts surely contributes to the dependability of the CPSoS. In particular, we propose a novel model-based approach for modeling and analyzing the autonomy levels of CPSs based on their awareness concerning their operational environment as well as their capability to react in a timely and safe manner while performing their activity. The model is further described in a UML profile and applied to represent activities for autonomous driving scenarios. Using a driving simulator, we implement such models on a target vehicle and we show the resulting safety improvement, especially in terms of reduced collisions.

KEYWORDS:
Cyber-Physical Systems of Systems; Autonomy; Controllability; Conceptual Modeling; Simulation

## 1 | INTRODUCTION

A Systems of Systems (SoS) can be defined as an integration of a finite number of Constituent Systems (CSs) that are independent and operable, and which have been networked together to achieve a specific higher goal [1,2]. Therefore, ensuring that an SoS can function as a single integrated system to support a common mission is a main goal for the SoS community [1,3]. However, the integration of CSs is not a trivial task due to the special characteristics that distinguish SoSs from other types of systems, and especially the autonomy of its components is a major cause of concerns [2,3]. This problem becomes even harder to be tackled when CSs are Cyber-Physical Systems (CPSs), where a CPS is a system consisting of cyber components (e.g., computer systems), controlled components (e.g., physical objects) and possibly of social components (e.g., interacting humans) [3], and the integration of CPSs into an SoS is called Cyber-Physical System-of-Systems (CPSoS).

In particular, the autonomy of CPSs may result in a lack of coordination among one another while pursing their own goals and/or the overall goals of CPSoS. This may lead to unsafe situations, and potentially to disasters in the case of critical CPSoS [4]. For instance, a self-driving car that was in autonomous driving mode has hit and killed a woman that was walking outside of the crosswalk recently [5]. Therefore, coordination among

[0]**Abbreviations:** SoS (System of Systems), CPS (Cyber-Physical System), CPSoS (Cyber-Physical System of Systems), RMU (Road Marking Unit), AI (Artificial Intelligence).

CPSs is a must to avoid such situations. To this end, CPSs that operate in the same environment need to be *aware* of one another in order to coordinate their activities. More specifically, awareness is essential for any coordination since it is about understanding the activities, locations, and situations of other CPSs, which provides the required knowledge for each CPS to safely perform its activities [6,7]. In this context, awareness is surely a key aspect that should be considered while analyzing the autonomy levels of CPSs. Considering the previous example, if the woman and/or the self-driving car had been aware of one another such disaster could have been avoided.

However, a CPS may not have the self-capability for acquiring the required knowledge about its operation environment; therefore, it may depend on other CPSs for such knowledge. Thus, we differentiate between two cases: when the CPS has the self-capability to be aware of its operational environment (e.g., aware by self), and when it needs to depend on other CPS for such awareness (e.g., aware by dependency). Consider for example a system that supports drivers to avoid overtake-related accidents on undivided rural roads, where Road Marking Units (RMUs) cooperate to assist the driver. The driver has the self-capability to be aware of its operational environment (road situation) if no obstacles limiting his visual capabilities (e.g., heavy fog, a sharp curve). In the case of heavy fog or a sharp curve, a driver may need to depend on RMUs for such knowledge since he does not have the self-capability for acquiring it. As the CPS may be dependent for such knowledge, we cannot say that the CPS is fully autonomous for performing its activity, thus, its autonomy should be adjusted (e.g., limited, restricted).

On the other hand, having the required awareness about the operational environment may not be enough to guarantee that a CPS can safely perform its activities. Considering the previous example, in cases of an imminent crash due to an overtake, the driver may not have the capability to take the right decision in a timely manner. Therefore, the *controllability* of the CPS (driver) over the performance of the activity should be also considered while analyzing its autonomy level. *Controllability* has been considered in several safety standards (e.g., Road vehicles - Functional safety (ISO 26262) [8]), and it is used to measures the ability to avoid a specified *harm*/*damage* through a timely reaction.

To this end, we advocate that in order to facilitate the integration of CPSs within the overall context of their CPSoS, we may need to adjust the autonomy level of some CPSs in a way that enables them to safely perform their own activities without endangering any other CPS that is operating in the same environment. This requires addressing the following three challenges: 1- Define the type of awareness that a CPS has (e.g., aware by self or by dependency) concerning its operational environment; 2- Define the level of controllability that a CPS has concerning the activity it aims to perform; and 3- Define the level of autonomy that each CPS should have while performing an activity, based on its type of awareness as well as its level of controllability.

To tackle these challenges, we have previously proposed a model-based approach for modeling and analyzing the autonomy levels of CPSs [4]. The approach offers a conceptual model and a UML profile for modeling the autonomy levels of CPSs, and it also proposes constraints expressed in Object Constraint Language (OCL) [9] for the verification of such models. In this paper, we initially build on [4] but we bring significant updates and additional contributions. The most notable novel contributions are that i) we refine the conceptual model with an augmented categorization of autonomy concepts, ii) we consequently adjust the UML profile and we re-design its application to a use case on autonomous driving, and ii) we implement such use case in a simulator, namely the Carla simulator [44]. The code we introduce in the simulator realizes the relations and concepts as defined in the application of the UML profile. In fact, exploiting the simulator, we are able to represent a car whose control system implements the relations and criteria of the UML profile to decide and adjust its autonomy level, depending on its awareness of the environment as well as its level of controllability of its activities. We demonstrate that a timely reduction of the autonomy, in determined situations, improves the overall safety, i.e., the number of collisions with other vehicles is decreased when the autonomy of vehicles is adjusted based on the proposed criteria.

The rest of this paper is organized as follows. Section 2 describes a motivating example we use to illustrate our work. We propose a conceptual model for modeling and analyzing the autonomy level of CPSs in Section 3, and we present and discuss our model-based approach in Section 4. Section 5 exercises the envisioned approach through simulations. Related work is presented in Section 6, and we conclude the paper and discuss future work in Section 7.

## 2 | GUIDING EXAMPLE: COOPERATIVE DRIVING ASSISTANCE SYSTEM

To contextualize our reasoning and guide the rest of the paper, we introduce a running example concerning an autonomous driving system. We consider as CPSoS a town where cars (or vehicles, in general) and pedestrians are acting, each for their own goal. We define the goal for a car as reaching a destination point $B$ within the city.

A car, consisting of the mechanics of the car, the control system within the car and the driver, can be described as a CPS operated by a human, and it is active within the town. The car interacts with other CPSs and humans to reach its specified goal: the overall CPSoS consists of a plurality of cars and pedestrians cruising in an area. In other words, each individual car is an autonomous CPS that tries to achieve its given objective without guidance from other systems [46].

Vehicles in a town and their interaction have been often used as a representative example of CPSoS. Consider for example, the coordination of cars on a busy highway to realize a smooth flow of traffic and their exchange of information to achieve such a common goal. In addition to the

direct communication by explicit signals among the drivers of the cars (e.g., the blinker or horn), information flow between cars is based on the observation of the movement of the vehicles on the road (caused by the actions of other drivers)[47].

We define a scenario of a car moving from the starting point $A$ to the destination point $B$ in the city. We assume the possible support for driving assistance from the infrastructure. In particular, we base our driving example on the works[13,14], and consequently the main components of our CPSoS are identified as:

- *Cars* (with drivers). Each car is moving from a starting point to a destination point.
- *Road Marking Units* (RMUs) that are integrated into the road infrastructure, and they collect and disseminate information that assists drivers to avoid accidents to the best possible extent. In our example, we assume that RMUs, when available, are distributing positions and directions of the cars on selected lanes.
- *Pedestrians*, which contributes to the entourage of the CPSoS. Pedestrians are not CPSs and do not operate them, but still they are an integral part of the resulting CPSoS.

We assume the existence of undivided roads, and of bad weather conditions that may lead to e.g., reduced visibility. This creates a challenging scenario for safe driving, as several studies linked bad weather and ambient lighting conditions that influence the drivers' visibility to the occurrence of accidents[10,11,12], especially if matched with the presence of undivided roads[12].

The CPSs can communicate in this CPSoS exploiting two different types of communication channels:

- *Direct channels*, where information can be exchanged (sent and received) relying on dedicated channels e.g., wired or wireless communication means. Two direct channels are identified: i) RMUs-to-RMUs, or Infrastructure-to-Infrastructure (I2I): allows RMUs to communicate with one another, and ii) RMUs-to-Vehicle, or Infrastructure-to-Vehicle (I2V): allows RMUs to communicate with vehicles.
- *Indirect channels*, where information can be acquired by sensing/observing the real world (domain)[15]. Two indirect channels are identified: i) Vehicle-to-Vehicle (V2V): allows vehicles/drivers to collect (acquire) information about other vehicles e.g., location, direction. (this is done for example through sensors as cameras, radar, as well as directly by the driver); and ii) Vehicle-to-RMUs (V2I): allows RMUs to collect information about close-by vehicles.

Figure 1 shows the system in terms of its main components allowing also to discuss their direct and indirect communication channels. A driver in such setting is subject to different scenarios. He might be driving alone, with limited risks of collisions with other participants of the CPSoS (Figure

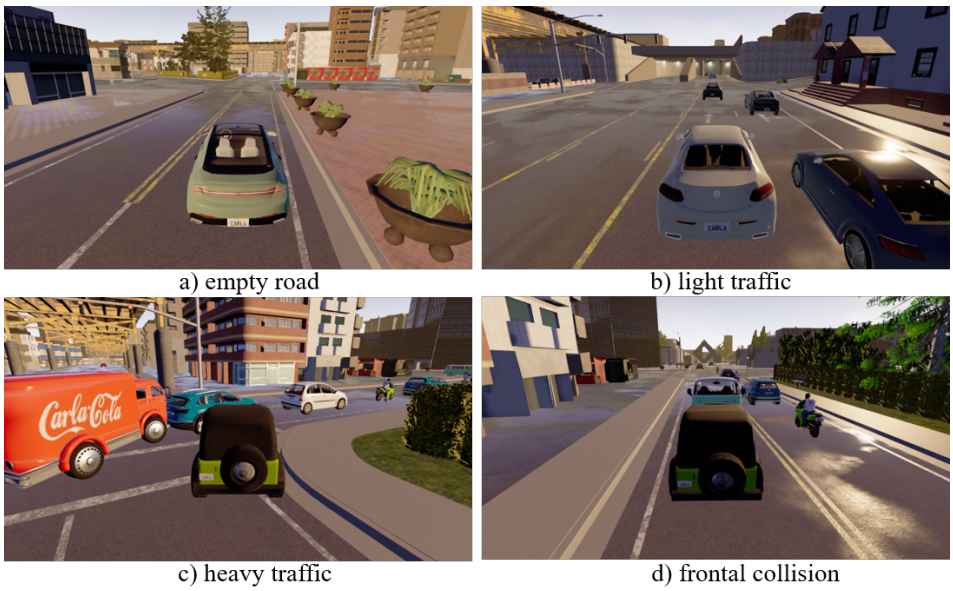



**FIGURE 1** Scenarios for assisted driving: a) empty road: direct and indirect interactions with other systems and risks of collisions are low or absent; b) and c) with traffic, direct or indirect interactions with other system are present and there are credible risks of collisions; d) a frontal collision (direct interaction between the two systems involved).

1a). Alternatively, He might be in presence of other participants; this requires a higher level of attention and potentially attention warnings. The risk of collision is credible, depending on the intentions of the driver, e.g., he should not risk lane change in Figure 1b, he should reduce speed in Figure 1c). Also in some cases, collisions are unavoidable (Figure 1d). In these last three scenarios, information from RMUs may be used to support the driver and providing driving assistance, i.e., ultimately resulting in a control of the driver (and the vehicle) actions and a reduction of its autonomy.

Assisted driving in a town is a good example, where the autonomy of CPSs might need to be adjusted based on their capability of performing a safe operation, i.e., based on their type of awareness as well as their level of controllability.

## 3 | A CONCEPTUAL MODEL FOR MODELING AND ANALYZING THE AUTONOMY LEVELS FOR CYBER-PHYSICAL SYSTEMS-OF-SYSTEMS

A conceptual model should include a set of fundamental constructs that represent the main *concepts* of the domain along with the *relationships* among them. Therefore, the main purpose of our conceptual model is to capture (model and analyze) the autonomy levels for CPSoS. In what follows, first we present the general concepts of the model, followed by the concepts, relations and attributes specialized for analyzing the autonomy levels.

The meta-model of the proposed conceptual model is depicted in Figure 2. In such Figure, we can identify a *CPSoS* that *integrates CPSs*. For example, the traffic system in a city can be seen as a *CPSoS* that *integrates* several *CPSs* such as RMUs, vehicles, pedestrians. Moreover, a *CPS* can *perform activities* while pursing its own objectives or the objectives of the overall *CPSoS*. An *activity* is *performed* in an operational environment, which we call *Sphere of Action (SoA)* that is a part of the domain, where activities can be performed [15]. A *SoA* can be *described* by *information*. For example, a driver may *perform* a turn (*activity*) at a crossing (*SoA*).

In what follows, we present concepts, relations and attributes for analyzing the autonomy levels of *CPSs*. Several researchers have argued that autonomy is not an all-or-nothing affair [16,17,18]. For instance, the notion of adjustable autonomy have been discussed in [18], and adaptable autonomy has been presented in [16]. Moreover, several works suggest analyzing autonomy level based on several aspects of the system such as the level of its intelligence [19], its motivations and behavior [20], or as in [18] based on four dimensions, namely: responsibility, commitment, authority and independence. Although no general agreement has been reached on which of these aspects should be considered while analyzing autonomy, there is general agreement that autonomy is surely influenced by the awareness of the system concerning its environment [21,22,17].

In this context, we define the relation *awareness of* between a *CPS* and the *SoA*, where it exists and operates, and it is characterized by a type attribute. Following [21], the type attribute can be: 1- *aware by self*, a CPS has the self-capability to be aware of its *SoA*, e.g., a CPS is socially independent, or 2- *aware by dependency*, a CPS needs to depend on other CPS to be aware of its *SoA*, e.g., a CPS is socially dependent. Note that a *CPS* must be aware of its *SoA* to operate in it, that is why we only consider the type of its awareness not whether it is aware or not of its *SoA*.

To analyze *aware by self*, we define the *acquire* relation between a *CPS* and *information* that *describes* a *SoA*. In other words, a *CPS* is *aware by self* of its *SoA*, if it is capable of *acquiring information* that *describes* the *SoA* by itself. For instance, a driver is said to be *aware by self* of the road situation, if he has the self-capability for acquiring information describing the road situation.

To analyze *aware by dependency*, we define *information provision* concept that captures the *provide*/*receive* relations between two *CPSs* concerning a specific *information*. In this case, a *CPS* is *aware by dependency* of its *SoA* since *information* that *describes* the *SoA* has been provided to it by another *CPS*. For example, a driver is *aware by dependency* of the road situation, when he depends on a RMU to provide him with such information.

On the other hand, to capture *controllability* of a CPS over the performance of its activity, we rely on the *controllability* relation between a *CPS* and the *activity* it aims to perform. Such relation is characterized by one attribute, namely *controllability level*, that can be:

**(1) Controllable**, when the *CPS* can detect and avoid any obstacle that might prevent it from safely performing its activity in a timely manner;

**(2) Might be controllable**, when the *CPS* might not be able to detect and avoid all obstacles that might prevent it from safely performing its activity in a timely manner;

**(3) Uncontrollable**, when the *CPS* cannot detect and/or avoid all obstacles that might prevent it from safely performing its activity in a timely manner.

For example, making a turn at a crossing during daylight where no obstacles are limiting the driver visibility is *controllable* by the driver. Instead, making a turn at a crossing with/without the support of RMUs when the driver visibility is limited *might be controllable* since the driver might not be able to detect and avoid all obstacles that might prevent him from safely performing this activity on time. However, cases such as an imminent crash due to making turn at a crossing is usually *uncontrollable* by a driver, since he does not have the capability to avoid the crash on time.

For capturing the autonomy level of a *CPS*, we extend the *perform* relation between the *CPS* and *Activity* concepts with the *autonomy level* attribute that can be: i) *Full autonomy*, ii) *Partial autonomy*, iii) *Limited autonomy* and iv) *No autonomy*. The autonomy level of a *CPS* can be analyzed based on: i) its type of *awareness* (e.g., *aware by self or dependency*) concerning its *SoA*; and ii) its *controllability level* concerning the *activity* it aims to *perform* e.g., *controllable*, *might be controllable* or *uncontrollable*. In details, autonomy levels are defined as follows:

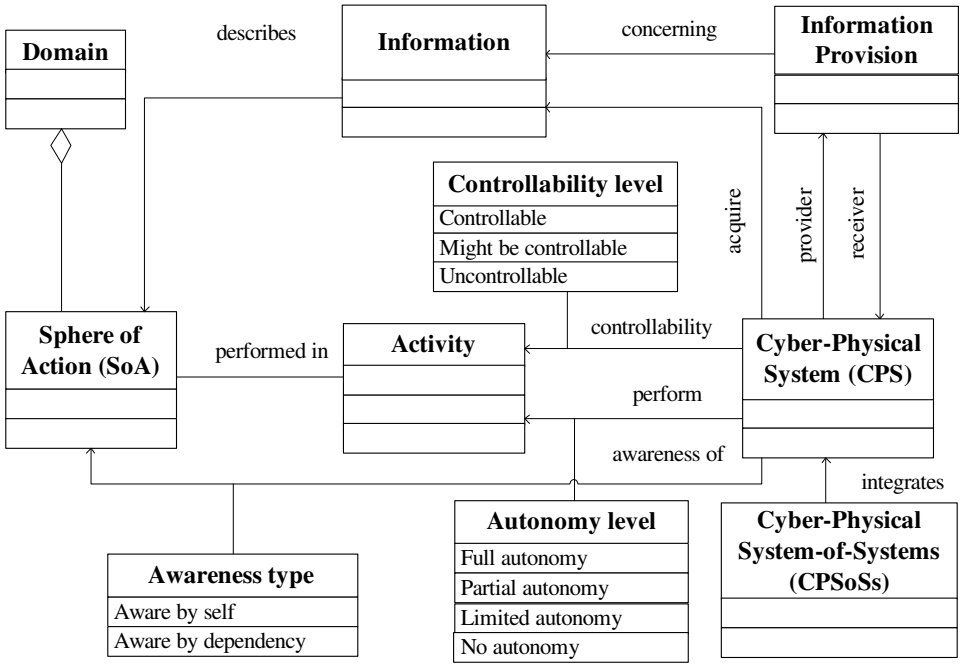



**FIGURE 2** The meta-model of the proposed conceptual model

**(1) Full autonomy**: the *CPS* can have *full autonomy* for performing a specific *activity* in a specific *SoA*, if it is *aware by self* of the *SoA*, and the *activity* is *controllable* with respect to its capability. For example, a driver can have *full autonomy* for performing an overtake, if he is *aware by self* of the road situation (e.g., clear visual conditions), and he is also capable of detecting and avoiding any obstacle that might prevent it for safely performing the overtake in a timely manner.

**(2) Partial autonomy**: the *CPS* should have *partial autonomy* for performing a specific *activity* in a specific *SoA*, if it is *aware by dependency* of its *SoA*, and the *activity* is *controllable* with respect to its capabilities. For example, when the driver is *aware by dependency*, the safe performance of turning at a crossing depends on information provided by other *CPS*, e.g., RMUs. Therefore, the driver cannot have *full autonomy* for performing the turning. *Partial autonomy* imply that other *CPSs* need to *monitor* and *assist* the *CPS* to safely perform its *activity*. In particular, although the overtake is *controllable* with respect to the driver capabilities, it cannot be performed safely unless RMUs *monitor* the driver along with other vehicles using the same *SoA* (e.g., sense their positions, directions, speed, etc.), and then *assist* the driver with information that is considered essential for safely performing the overtake.

**(3) Limited autonomy**: the *CPS* should have *limited autonomy* for performing a specific *activity* in a specific *SoA*, if it is *aware by self* or *aware by dependency* of its *SoA*, and the *activity might be controllable* with respect to its capabilities. For example, a driver should have a *limited autonomy* for turning at a crossing, if he is *aware by self/dependency* of the road situation, but he might not be able to detect and avoid all obstacles that might prevent him from safely turning at a crossing. In particular, *Limited autonomy* is considered to cover situations when drivers might have poor judgment concerning the road situation that is why they might not be able to detect and avoid all obstacles on time.

**(4) No autonomy**: the *CPS* should have *No autonomy* for performing a specific *activity* in a specific *SoA*, if the *CPS activity* is *uncontrollable* with respect to its capabilities. For example, a driver should have *No autonomy* for turning at a crossing, when the turning is *uncontrollable* by him regardless if he is *aware by self/ dependency*, i.e., the driver might be prevented from turning. In such a situation, the driver is not believed to be able to safely turning. That is why, he should have no autonomy for such activity, i.e., its autonomy should be adjusted to *No autonomy*. *No autonomy* implies that other *CPSs* may interrupt and control[1] the activity to be performed. For instance, several Advanced Driver Assistance Systems (ADAS) have been developed to increase the driver's safety by interrupting and controlling the activity to be performed [24,25].

Figure 3 summarizes the identification of the autonomy level based on controllability and awareness. Figure 4 describes the state transitions envisioned for the autonomy model, as they emerge from the discussion above.

[1] *Control* can be defined as the capability to direct or influence the behavior of others [23], and it is constructed based on different kinds of power (e.g., legitimate, reward, expert, etc.), which is out of the scope of this paper

| Awareness type | Controllability level | | |
|---|---|---|---|
| | Controllable | Might be controllable | Uncontrollable |
| Aware by self | Full Autonomy | Lim. Autonomy | No Autonomy |
| Aware by dependency | Par. Autonomy | Lim. Autonomy | No Autonomy |

**FIGURE 3** Determining autonomy level based on awareness type and controllability level

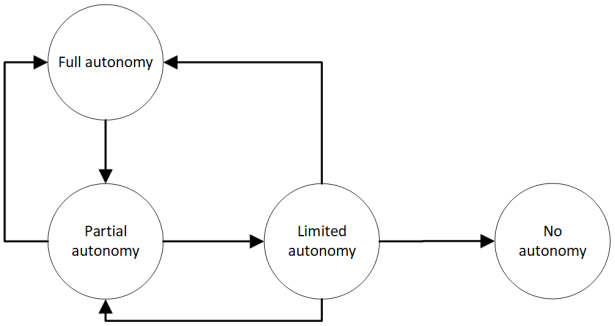



**FIGURE 4** State machine describing the autonomy levels according to the conceptual model proposed. Once no-autonomy is reached, an event from the system is required to re-establish acceptable driving autonomy. In our running example, this may be a maneuver or an acknowledgment from the driver.

## 4 | A MODEL-BASED APPROACH FOR MODELING AND ANALYZING THE AUTONOMY LEVELS FOR CYBER-PHYSICAL SYSTEMS-OF-SYSTEMS

In what follows, we describe the modeling environment in our approach. First, we present the methodological process. Second, we describe the UML profile that allows for modeling the autonomy levels. Then, we briefly discuss the reasoning support that can be used to verify such models.

### 4.1 | Methodology

The process underlying our approach is shown in Figure 5. It aims to assist software engineers while modeling the autonomy levels for *CPSs* in the overall context of *CPSoS*, and is composed of two main phases:

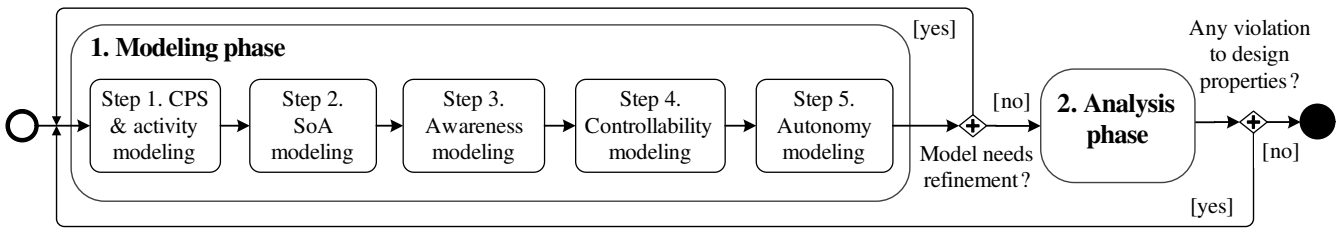



**FIGURE 5** A process for modeling and analyzing the autonomy levels for CPSoS

(1) **Modeling phase** aims to model the autonomy levels for *CPSs* in the overall context of *CPSoS*, and it is composed of the following five steps:

**1 CPS & activity modeling**, in which we define and model the *CPSoS* in terms of its main *CPSs* along with the *activities* they aims to perform.

**2 SoA modeling**, in which we define and model the various *SoAs*, where each of the *activities* are performed, and we use *performedIn* relation to describe the relation between them.

**3 Awareness modeling**, in which we define and model the *awareness type* of *CPSs* concerning the *SoAs*, where they aim to perform their *activities*. This can be done by first modeling the relations between *SoAs* and *information* describing them (e.g., *describe*). Then, the relations between *CPSs* and *information* describing such *SoAs* (e.g., *acquire*). Finally, if a *CPS* is not able to *acquire information* that *describes* a *SoA* in which it *performs* an *activity*, we model the *information provision* between it and the *CPS* that is able to *acquire* such *information*.

**4 Controllability modeling**, in which we define and model the *controllability level* for each *CPS* concerning each *activity* they aim to perform.

5 **Autonomy modeling**, in which we define and model the *autonomy level* for each *CPS* concerning each *activity* it aims to perform based on its type of *awareness* and its *controllability level*.

**(2) Analysis phase** aims to verify the correctness and consistency of the model depending on a set of properties of the design.

## 4.2 | Modeling Phase

We followed the guidelines proposed in[26] to formulate our UML profile in terms of its stereotypes and tagged values based on the previously defined conceptual model. In what follows, we present our UML profile[2] that is shown in Figure 6. Then, we describe how it can be applied to model the cooperative driver overtaking assistance system.

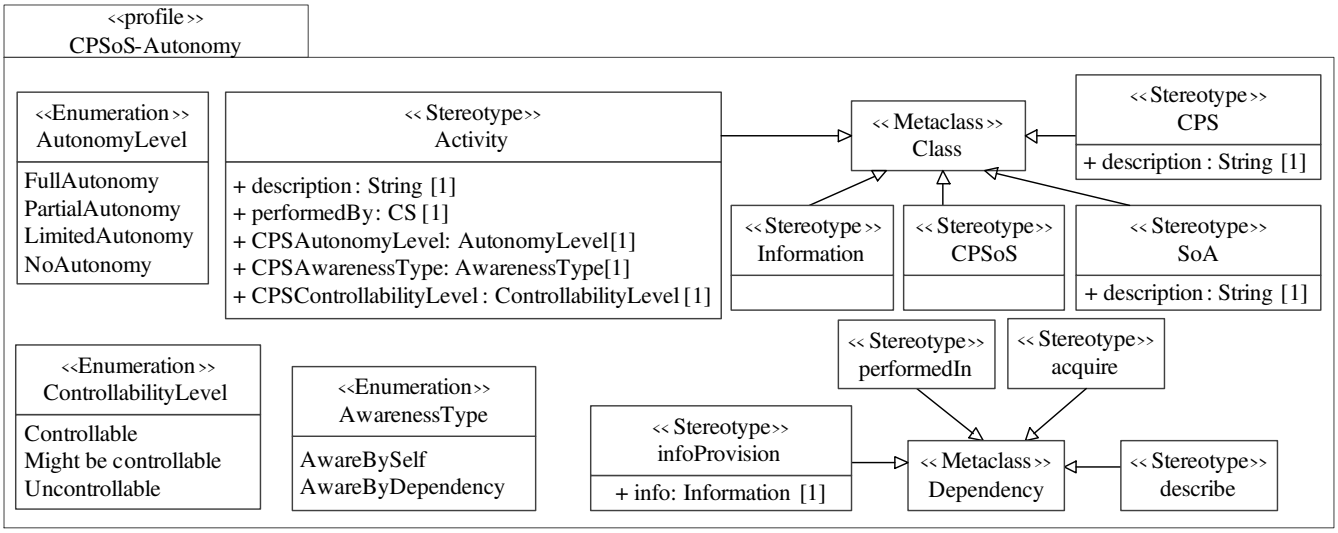



**FIGURE 6** UML Profile for modeling the autonomy levels for CPSoS

Several stereotypes have been specialized from the «Class» Metaclass to capture the main modeling concepts. For example, The «CPSoS» stereotype has been developed based on the *CPSoS* concept, and it is an *abstract* stereotype, i.e., it cannot be used to instantiate *instances*. While the «CPS» and «SoA» stereotypes have been developed based on the *CPS* and *SoA* concepts respectively, and each of them has a *description* property to describe its main characteristics. Moreover, the «Information» stereotype has been developed based on the *Information* concept.

The «Activity» stereotype has been developed based on the *Activity* concept, and it has several properties: 1- *description* that is used to describe the main characteristics of the activity; 2- *performedBy* that is used to identify the *CPS*, which is responsible for performing the activity; 3- *CPSAutonomyLevel* that describes the level of autonomy the performing *CPS* has; 4- *CPSAwarenessType* that describes the type of awareness the performing *CPS* has; 5- *CPSControllabilityLevel* that describes the level of controllability the performing *CPS* has.

The last three properties can be described depending on the following three «Enumerations» respectively: 1- «AutonomyLevel» that can be *FullAutonomy*, *PartialAutonomy*, *LimitedAutonomy* or *NoAutonomy*; 2- «AwarenessType» that can be *AwareBySelf* or *AwareByDependency*; and 3- «ControllabilityLevel» that can be *Controllable*, *MightbeControllable* or *Uncontrollable*. Note that the *autonomy level* of a *CPS* for performing a specific *activity* in a specific *SoA* is determined based on its *awareness type* as well as *controllability level* in accordance with Figure 3.

On the other hand, several stereotypes have been specialized from the «Dependency» Metaclass to capture the relations among the previously mentioned stereotypes. «acquire» stereotype captures dependencies starting from «CPS» and pointing towards «Information». The «describe» stereotype captures dependencies starting from «Information» and pointing towards «SoA». The «performedIn» stereotype captures dependencies starting from «Activity» and pointing towards «SoA». Finally, «infoProvision» stereotype captures dependencies starting from a «CPS» and pointing towards another «CPS», and it has one property that indicates «Information» to be provided.

In what follows, we describe how the UML Profile can be applied to model a realistic scenario of the cooperative driver overtaking assistance system that is represented in Figure 7. Following our methodology, we start by identifying all *CPSs* along with their *activities*. We identify a *driver*

[2]The profile has been developed depending on Eclipse-Papyrus (https://www.eclipse.org/papyrus/), and it is available at https://bit.ly/2QHzDQu

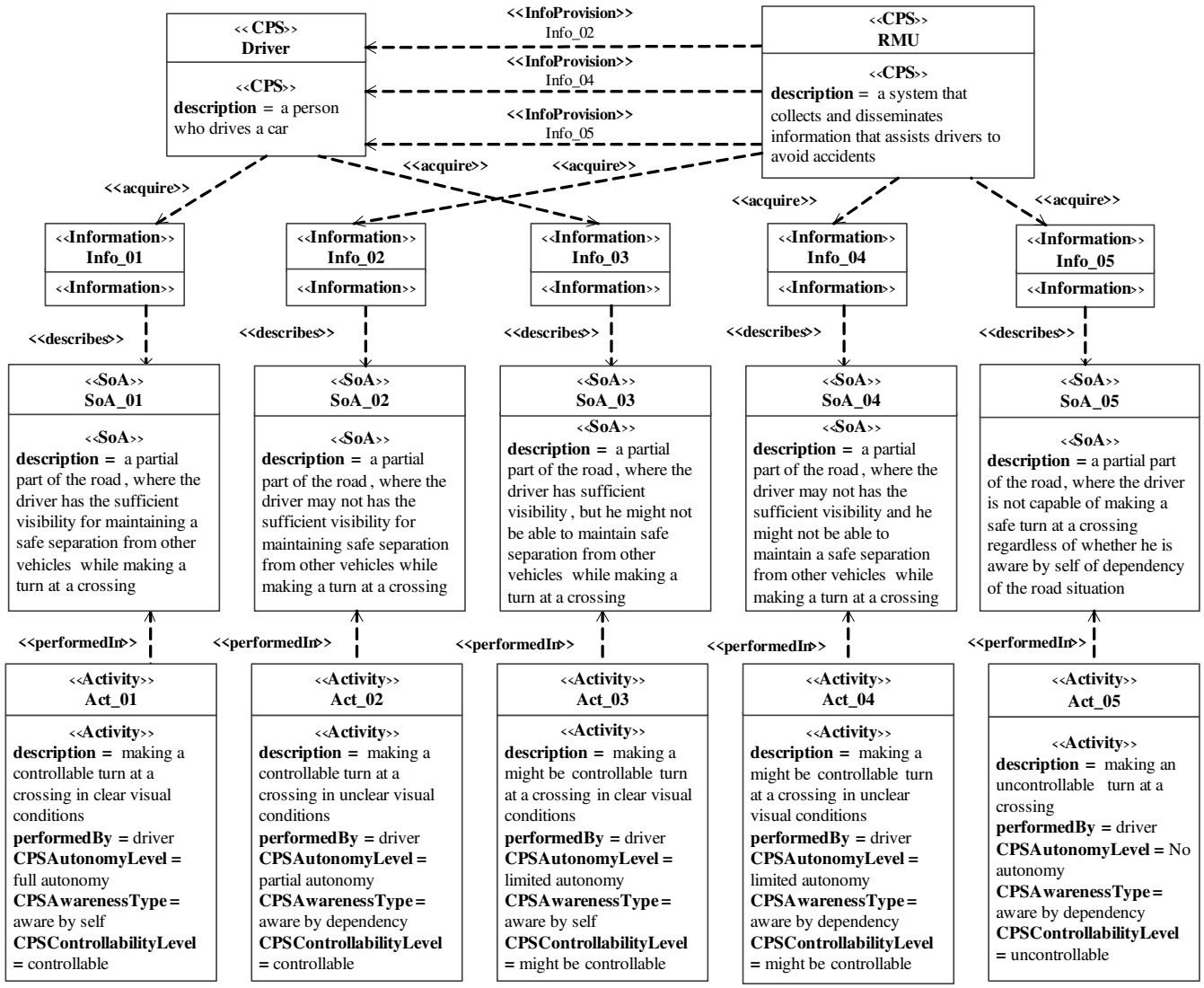



**FIGURE 7** Applying the UML profile for modeling the autonomy level for CPSoS

and only one *RMU* (to simplify the scenario), which have been represented as `CPSs`. The *driver* aims to perform five different types of making a turn at a crossing (represented as `activities`). We believe they cover almost all types of such activity that can be performed by a driver at a crossing , namely: 1- making a controllable turn at a crossing in clear visual conditions (full autonomy); 2- making a controllable turn at a crossing in unclear visual conditions (partial autonomy); 3- making a might be controllable turn at a crossing in clear visual conditions (limited autonomy); 4- making a might be controllable turn at a crossing in unclear visual conditions (limited autonomy); and 5- making an uncontrollable turn at a crossing (No autonomy). These `activities` are represented in Figure 7 as `Act_01-05`, along with a short description about each of them.

After modeling the *CPSs* along with their *activities*, we need to identify and model all *SoAs*, where each of these *activities* is performed (Step 2). To this end, we have modeled: `SoA_01-05` (shown in Figure 7), where `Act_01-05` are *performedIn* respectively.

These `SoAs` can be defined as follows. `SoA_01` describes a partial part of the road, where the driver has sufficient visibility for maintaining safe separation from other vehicles while making a turn at a crossing. `SoA_02` describes a partial part of the road, where the driver may not has sufficient visibility for maintaining safe separation from other vehicles while making a turn at a crossing. `SoA_03` describes a partial part of the road, where the driver has sufficient visibility, but he might not be able to maintain safe separation from other vehicles while making a turn at a crossing. `SoA_04` describes a partial part of the road, where the driver may not has sufficient visibility and he might not be able to maintain a safe separation from other vehicles while making a turn at a crossing. Last, `SoA_05` describes a partial part of the road, where the driver is not capable of making a safe turn at a crossing regardless of whether he is aware by self of dependency of the road situation. Note that `SoA_05` can be defined dynamically

by the cooperative driver overtaking assistance system by analyzing the location, speed and direction of the vehicles. Then, the system can infer whether an overtake is *controllable* or *uncontrollable* by the driver.

The third step is *awareness modeling*, in which we define and model the *awareness type* of *CPS* concerning each of its *activities*. First, we model the relations between *SoAs* and *information* describing them. We modeled five information items, namely `Info_01-05` that *describe* `SoA_01-05` respectively. Then, we need to model the relations between *CPSs* and *information* describing the *SoAs* in which they perform their activities. The *driver acquires* `Info_01` and `Info_03`. In this context, the *driver* is *aware by self* of both `SoA_01` and `SoA_03`. While the *RMU acquires* `Info_02`, `Info_04` and `Info_05`. Finally, we model the *information provision* among *CPSs* for *information* they need but they are not able to *acquire*. We have three *information provision* relationships from the *RMU* towards the *driver* concerning `Info_02`, `Info_04` and `Info_05`. Therefore, the *driver* is *aware by dependency* of `SoA_02`, `SoA_04` and `SoA_05`.

In the fourth step, we define and model the *controllability levels* for each *CPS* concerning each *activity* they aim to perform. Based on the description of the *activities* along with their related *SoAs*, we conclude that `Act_01` and `Act_02` are *controllable* by the *driver*, `Act_03` and `Act_04` are *might be controllable* by the *driver*, while `Act_05` is *uncontrollable* by the *driver*. In the fifth and final step, we define and model the *autonomy levels* for each *CPS* concerning each *activity* it aims to perform based on its type of *awareness* and its *controllability level* in accordance with Figure 3. To this end, the driver have *FullAutonomy* concerning activity `Act_01` (e.g., driver is *aware by self* and *activity* is *controllable*); *PartialAutonomy* concerning activity `Act_02` (e.g., driver is *aware by dependency* and *activity* is *controllable*); *LimitedAutonomy* concerning activity `Act_03` (e.g., driver is *aware by self* and *activity might be controllable*); *LimitedAutonomy* concerning activity `Act_04` (e.g., driver is *aware by dependency* and *activity might be controllable*); and *NoAutonomy* concerning activity `Act_05` (e.g., driver is *aware by dependency* and *activity* is *uncontrollable*).

### 4.3 | Analysis Phase

The analysis aims at verifying created models, which allows to not relying exclusively on the modeling capabilities of the profile to perform the required analysis. In particular, we have defined a set of properties of the design (shown in Table 1) expressed as OCL constraints, which specify logical constraints that can be used to verify the correctness and consistency of the model. More specifically, the model is correct and consistent if all of these properties hold. In case any of them has been violated, the software engineer/designer will be notified and he needs to modify the model accordingly to address such violation. Therefore, the process might be iterative until all the properties of the design hold.

## 5 | IMPLEMENTATION AND EVALUATION THROUGH SIMULATIONS

We applied the conceptual model, the profile and its application discussed above to implement a case study in a driving simulator. The case study was crafted to closely represent the running example used in this paper and to match the application of the UML profile in Figure 7. We mostly

**TABLE 1** Properties of the design

| | |
|---|---|
| **Pro1.** | Dependencies with the stereotype «`performedIn`» can only have a class with a stereotype «`Activity`» as a source of the dependency and a class with a stereotype «`SoA`» as a destination. |
| **Pro2.** | Dependencies with the stereotype «`describe`» can only have a class with a stereotype «`Information`» as a source of the dependency and a class with a stereotype «`SoA`» as a destination. |
| **Pro3.** | Dependencies with the stereotype «`acquire`» can only have a class with a stereotype «`CPS`» as a source of the dependency and a class with a stereotype «`Information`» as a destination. |
| **Pro4.** | Dependencies with the stereotype «`infoProvision`» can only have classes with a stereotype «`CPS`» as a source and a destination of the dependency. |
| **Pro5.** | The source and destination of dependencies with the stereotype «`infoProvision`» can not be the same «`CPS`» stereotype. |
| **Pro5.** | Dependencies with the stereotype «`infoProvision`» should define the «`Information`» stereotype that is subject to the provision. |
| **Pro6.** | Each class with a stereotype «`Activity`» should have at least one dependency with a stereotype «`performedIn`» pointing towards a class with a stereotype «`SoA`». |
| **Pro7.** | Each class with a stereotype «`Information`» should have at least one dependency with a stereotype «`describe`» pointing towards a class with a stereotype «`SoA`». |
| **Pro8.** | The CPS autonomy level of each class with a stereotype «`Activity`» should be determined in accordance with the Figure 3. |

realize this through coding an extension to the control system of an autonomous vehicle, in such a way that the code implements the described activities and relations. This allowed us to exercise the proposed approach and measure its benefits, especially in terms of reduced collisions.

## 5.1 | Realization of the running example

We select the Open Urban Driving Simulator Carla[44] (Car Learning to Act) to reproduce the example of Section 2. Carla has been implemented as an open-source layer over the Unreal Engine 4 (UE4,[48]) to support training, prototyping, and validation of autonomous driving models, including both perception and control. Carla includes urban layouts, several vehicle models, buildings, pedestrians, street signs, etc. The simulation platform supports flexible setup of sensor suites, and in particular we will exploit the RGB camera. Further, it provides to the user information on the various vehicles as position, orientation, speed, acceleration, as well as detailed data on collisions. Weather and time of day can also be specified[44].

Representing our running example in Carla requires to simulate towns with pedestrians and vehicles, undivided lanes, and bad weather conditions. As discussed in Section 2 these are the conditions that mostly lead to risky events.

Additionally, the implementation of RMUs is required; these can be simulated exploiting the *carla.World* API, which provides information on all elements at each simulation step. By filtering such information, it is possible to simulate RMUs that are dispatching information on nearby vehicles to a target car. We use Carla 0.9.6 and the towns Town01, Town02 which are two basic town layouts with all three-way junctions[44,45].

The car includes all the logic that implements the autonomy model. This logic allows elaborating the information received from the RMUs and enforce control on the vehicle, when needed. The logic of the autonomy model can be deactivated simply turning off information acquisition from the RMUs. Such information processing, elaboration and control enforcement are realized as a control function of the car that is executed at each simulation step as detailed in Section 5.2.

Finally, we represent our driver using a learned agent for autonomous driving. In other words, our driver is actually the AI (Artificial Intelligence) component that is able to autonomously drive a car in Carla. Amongst the various autonomous driving agents that exist for Carla, we reuse the one from[45] because i) the code is compliant with the Carla 0.9.6 release (released on July 2019), ii) it presents very good performances, and iii) it includes suites for benchmarking, data retrieval and analysis.

The driving system in[45] has excellent results, with a very small number of collisions whose responsibility is actually attributable to erroneous decision of the learned agent: in other words, the learned agent provided in[45] could be identified as an excellent driver. To prove the efficacy of our approach, we need instead a *distracted driver that makes mistakes*. For this reason, we create a new learned agent from[45], such that erroneous decisions are frequent. Our learned agent is obtained applying the training in[45] only for the privileged agent training and with values for training that are significantly lower than those recommended in[45]. It should be remarked that, to the extent of this paper, we are not interested in the principles of AI and the insights of the AI solution selected, while we are just interested in finding an autonomous driving model for Carla that we can exploit to represent a distracted driver.

## 5.2 | Implementation of the autonomy model

The logic for the autonomy model is realized exploiting the information on orientation and distance of cars with respect to one target car. Such realization has been devised after various attempts and preliminary runs to understand the behaviour of the simulator and the proper tuning of model parameters. The logic that implements the autonomy model is structured in three phases, which guarantees that a CPS can safely perform its own activities without endangering any other CPS that is operating in the same environment. These phases are executed iteratively at each simulation step as follows:

1. *information acquisition from RMUs and processing to capture the awareness type*;
2. *definition of the new autonomy state to capture controllability; based on the awareness type, the autonomy level is determined*;
3. *enforcement of control over the driver; considering the determined autonomy level of the driver, RMUs might interfere and control the vehicle*.

We explore these three phases in details. In the first phase, the target car acquires information from the RMUs, selects the cars in front (i.e., the cars the bad driver could collide with), and computes the relative distance and orientation with respect to such cars.

In the second phase, the current state from the autonomy model is decided, based on rules on distance and orientation of cars in front and the capability of the driver to safely perform the activity. These rules and the consequent state changes are summarized in Figure 8, while Figure 9 presents a graphical and intuitive explanation of the rationale for such above-mentioned rules. In more details:

- The car starts in full autonomy and stays in this state as long as there are no risks of collision. The state *full autonomy* is entered again if the risk of collision is irrelevant. In this state, the driver has full visibility of the situation with no risks nearby: he is aware by self and the activity is controllable.

- The state *partial autonomy* is entered if there are cars nearby, but the risk of a collision is minimal. One of the following conditions must hold (and obviously, the conditions for the *limited autonomy* and *no autonomy* states described below are not satisfied): i) at least one car going in the same direction is at a distance between $14$ and $18$ meters, or ii) at least one car going in opposing direction is at a distance between $10$ and $12$ meters and the relative angle is smaller than $60°$. In this state, the driver does not have an entire, clear visibility of the road, but he is aware by dependency of any nearby car thanks to the RMUs, and there is enough room for reaction (his activity is controllable).

- The state *limited autonomy* is entered if there are cars nearby, and there is the significant risk of a collision. One of the following conditions must hold (and obviously, the conditions for the *no autonomy* state described below are not satisfied) : i) a car going in the same direction is at a distance smaller than $14$ meters, with relative angle smaller than $35°$, or ii) a car going in the opposite direction is at less than $10$ meters and the relative angle is smaller than $60°$. In this state, there are cars nearby and limited overall visibility of the situation, but there is still sufficient reaction time: the activity might be controllable.

- The state *no autonomy* is entered if there is an approaching collision. The following conditions must hold to enter this state: i) at least one of the cars going in the same direction of the target car is at a distance smaller than $8$ meters, with relative angle smaller than $35°$, or ii) at least one of the cars going in the opposite direction is at a distance smaller than $4$ meters and the relative angle is smaller than $45°$. In this situation, the activity is uncontrollable by the driver; in practice, it is necessary to take immediate actions to avoid collisions and this cannot be relied upon the driver any longer.

In the third phase, control actions are enforced over the driver, depending on the current state, as follows:

- Current state: FULL AUTONOMY. In this state, driver has full control over the car. The driver drives without restrictions, i.e., no enforced control actions from other CPSs.

- Current state: PARTIAL AUTONOMY. In this state, activities are controllable by the driver but he is aware by dependency. Therefore, the freedom of actions of the driver is partially reduced, to make sure that no dangerous situations arise. In our implementation, we assure that throttle is less than a given threshold. This is applied by the following simple code:

```
if control.throttle >0.8:
    control.throttle =0.8
```

- Current state: LIMITED AUTONOMY. In this state, the actions might be controllable by the driver (and he is aware by self or dependency). As a result, we further reduce the freedom of actions of the driver, to make sure that no dangerous situations arise. In our implementation, we further reduce throttle. This is applied by the following simple code:

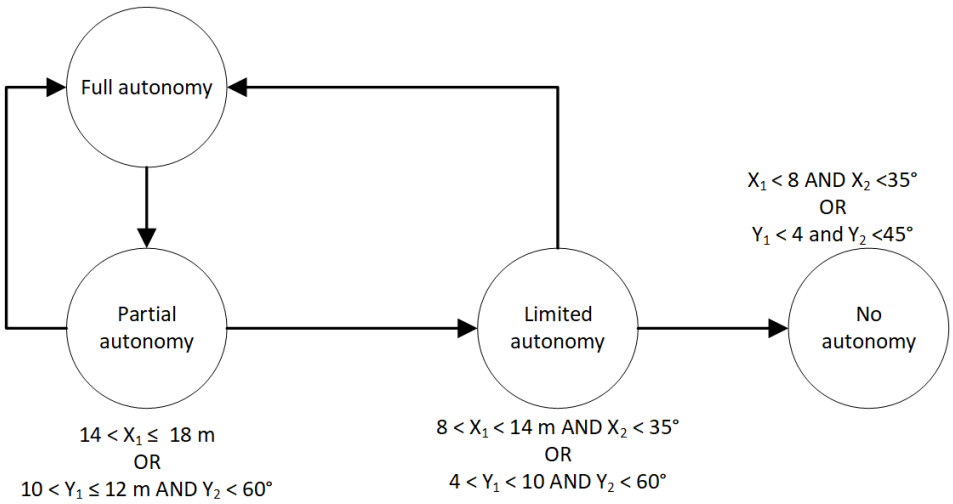



**FIGURE 8** Conditions for state change that are implemented in the autonomy model for a target car. By $X_1$ we represent distance of our target car with respect to a car going in the same direction. Instead $X_2$ is the relative angle from such car. By $Y_1$ we represent distance of our target car with respect to a car going in the opposite direction, while $Y_2$ is the angle from such car.

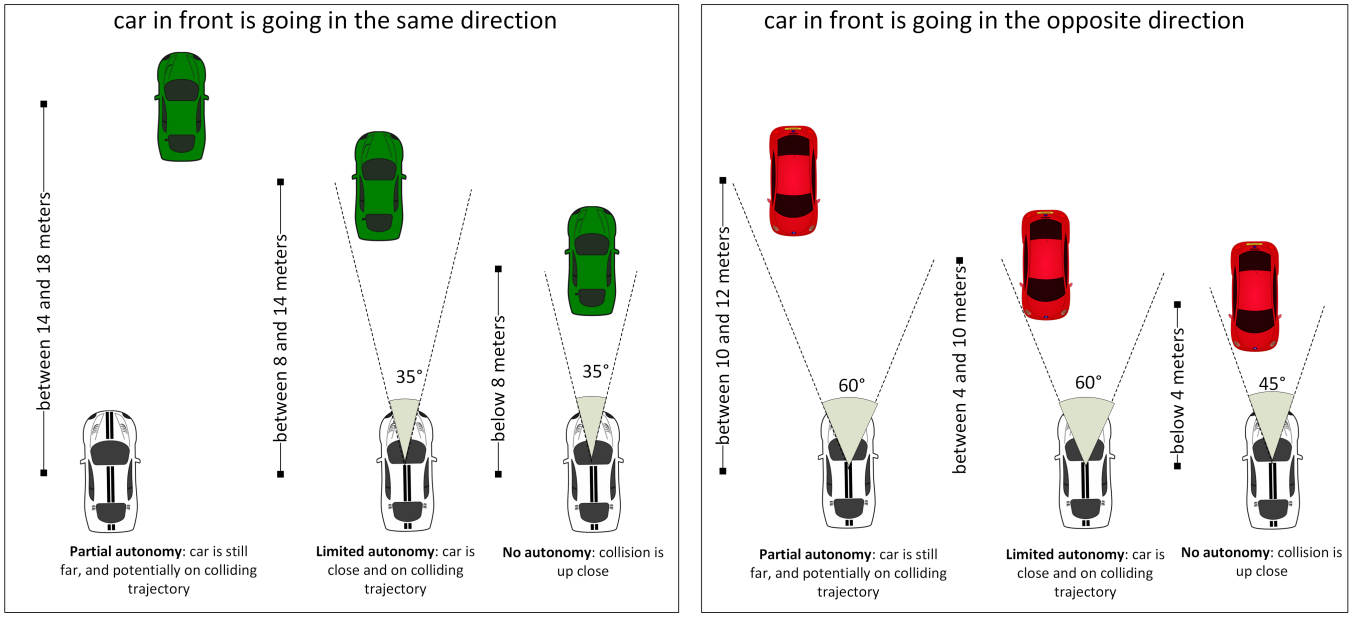



**FIGURE 9** Graphical explanation of the values in Figure 8 and adopted in our implementation of the model.

```
if control.throttle>0.7:
    control.throttle=0.7
```

- Current state: NO AUTONOMY. In this state, we forcefully restrict any action from the driver, activating brakes. This is applied by the following simple code:

```
control.throttle=0.0
control.brake=1.0
```

Note that this kind of reduction should be analyzed at the light of the characteristics of the simulator and the driving model (which has non-constant acceleration), where for example a car with max throttle bounded to $0.7$ has the effect of proceeding at a low speed.

We observe that this approach does not protect against pedestrians and objects (e.g., poles, walls), as the RMUs are not informing the car about them. However, they are still included in the simulations because they are essential elements of a city and consequently of the target CPSoS.

Finally, from a practical viewpoint, to implement this model we operated with the Carla APIs [50], and the files carla_utils.py and run_benchmark.py. The augmented control from the state Partial, Limited and No Autonomy is implemented directly in the file run_benchmark.py. The steps are the following. The output of the observation at a simulation step is achieved (i.e., the images from the camera, as well as the presence of other cars in front: this is done at a rate of 10 Frame per Seconds), and then the control is decided by the autonomous driving of [45] according to our privileged agent model *model64.th* [49] (this represent our bad driver). Then, the current autonomy level is computed, and the control is overwritten according to the rules above. Finally, the control is applied on the vehicle.

## 5.3 | Description of the Experiments

We exercise the learned agent, when the autonomy model is activated and when it is deactivated. The test plan is based on the *corl2017* benchmark from [44]. In this benchmark, a selected car has to reach a destination position $B$ from a starting position $A$ in a given time. Re-using the nomenclature from [44], the starting and destination positions are selected such that two test objectives are set:

- *Straight road*: Destination position $B$ is located straight ahead of the starting point $A$.
- *Turn road*: Destination position $B$ is one turn away from the starting position $A$.

In our test plan, reported in Table 2, each of the two test objective is exercised on the two towns Town01 and Town02 with two different weather conditions (hard rain at noon, and hard rain at sunset) and varying number of vehicles and pedestrians. Various test suites are defined, each one

consisting of 50 runs. Different runs have different starting and destination positions. We observe that the first eight test suites simulate towns without cars: we expect in this case that the results with and without the autonomy model are very similar. Instead, the other test suites include cars, and we expect a reduced number of collisions when the autonomy model is running.

**TABLE 2** List of planned experiments.

| Test Suite | Objective | Town | vehicles | pedestrians | executed runs |
|---|---|---|---|---|---|
| ST01-00 | Straight road | Town01 | 0 | 0 | 50 |
| TT01-00 | Turn road | Town01 | | | 50 |
| ST02-00 | Straight road | Town02 | | | 50 |
| TT02-00 | Turn road | Town02 | | | 50 |
| ST01-01 | Straightroad | Town01 | 0 | 80 | 50 |
| TT01-01 | Turn road | Town01 | | | 50 |
| ST02-01 | Straightroad | Town02 | | | 50 |
| TT02-01 | Turn road | Town02 | | | 50 |
| ST01-02 | Straight road | Town01 | 80 | 0 | 50 |
| TT01-02 | Turn road | Town01 | | | 50 |
| ST02-02 | Straight road | Town02 | | | 50 |
| TT02-02 | Turn road | Town02 | | | 50 |
| ST01-03 | Straight road | Town01 | 30 | 0 | 50 |
| TT01-03 | Turn road | Town01 | | | 50 |
| ST02-03 | Straight road | Town02 | | | 50 |
| TT02-03 | Turn road | Town02 | | | 50 |
| ST01-04 | Straight road | Town01 | 50 | 0 | 50 |
| TT01-04 | Turn road | Town01 | | | 50 |
| ST02-04 | Straightroad | Town02 | | | 50 |
| TT02-04 | Turn road | Town02 | | | 50 |
| ST01-05 | Straight road | Town01 | 50 | 50 | 50 |
| TT01-05 | Turn road | Town01 | | | 50 |
| ST02-05 | Straight road | Town02 | | | 50 |
| TT02-05 | Turn road | Town02 | | | 50 |

A run is concluded if either i) the destination position $B$ is reached, or ii) the car collides (against other vehicles, pedestrians, or surroundings as for example walls, street lightning poles, signposts), or iii) a timeout expires. The run is considered successful only if the destination position is reached.

With respect to the *corl2017* benchmark, we introduce the following modifications: i) the timeout is increased, and ii) the run is interrupted if our target car has a collision. This is aligned to the objective of our paper, which privilege travelling without collisions rather than minimizing travel time.

## 5.4 | Results

We present here results of the execution of the test suites in Table 2. Summarizing results, detailed log files, and videos of each individual run are available at[49].

We start our discussion from Table 3, that describes the results of the test suites that do not include cars and consequently set the baseline for our analysis. In these runs, the performances achieved with and without the autonomy model are very similar, as it is evident from Table 4 (consider the rows *without vehicle*). These differences are only due to the unavoidable randomness that is in the simulation environment itself, as for example small differences in the timing behaviour of the simulator in different runs, and that in some cases can lead to different outcomes.

We use Table 4 also to start discussing on the test suites that include cars i.e., where the presence of the autonomy model should create differences. The first observation is that, on average, the autonomy model allows reducing the remaining distance to the destination point $B$, but at

**TABLE 3** Results with the *autonomy model* in place for the test suites that do not include vehicles (50 runs per test suite) .

| Test Suite | Success Rate % | Successful runs | Sum of remaining distance to destination points (m) | Simulated time (seconds) | Number of collided runs | Timeouts |
|---|---|---|---|---|---|---|
| ST01-00 | 52 | 26 | 4180 | 14823 | 24 | 0 |
| TT01-00 | 20 | 10 | 8628 | 112534 | 20 | 20 |
| ST02-00 | 44 | 22 | 1533 | 10885 | 28 | 0 |
| TT02-00 | 44 | 22 | 4434 | 56990 | 17 | 11 |
| ST01-01 | 48 | 24 | 4223 | 13940 | 26 | 0 |
| TT01-01 | 22 | 11 | 8849 | 87831 | 24 | 15 |
| ST02-01 | 42 | 21 | 1625 | 10094 | 28 | 1 |
| TT02-01 | 40 | 20 | 4335 | 31188 | 29 | 1 |

**TABLE 4** Differences with and without the autonomy model, for test suites that do not include cars (listed in Table 3).

| *Average values for the individual run:* | | **without autonomy** | **with autonomy** |
|---|---|---|---|
| average remaining distance to destination point (meters) | without cars | 94,5 m | 95,2 m |
| | with cars | 100,0 m | 89,7 m |
| average duration (seconds) | without cars | 846 s | 853 s |
| | with cars | 534 s | 1202 s |

the price of a significant increase in the duration of the run. This is in general intuitive, as the application of the autonomy model causes a reduction in speed to favor safety over performance. Duration is increased by the autonomy model also because there are deadlock conditions, where cars are unable to progress until the timeout of the run is reached. This happens for example if the target car enters the opposing lane, and stops in front of an approaching car. The simulated cars do not have the logic to solve this situation, and remain still until timeout expires. This situation is explained in Figure 10 and Figure 11, using frames extracted from the recorded videos, for a corresponding run of test suite ST02-05 executed with and without autonomy.


**FIGURE 10** Without the autonomy model, the run terminates with a collision.


**FIGURE 11** With the autonomy model in place, there is no collision but the cars will keep facing each other until timeout.

We now detail results for the suites that include cars. Complete results for the remaining test suites are reported in Table 5. It can be observed that, using the autonomy model, the number of collisions is reduced by $31,7\%$ . This is also highlighted in Figure 12. The remaining collisions are essentially collisions against walls, pedestrians, or when the vehicle is still but is bumped by other cars e.g., because it stops in the middle of a crossing. Instead, as previously discussed, the number of timeouts is significantly increased. This is the main reason for the consequent increase of the simulation time. Also, the autonomy model slightly increases the number of successful runs; as a consequence, this results in a reduction of the remaining distance to the destination points.

To give a clear understanding of the behaviour of the simulations with and without the autonomy model, we show examples with the support of frames extracted from the recorded videos. Figure 13 is a collision scenario without the autonomy model, while Figure 14 is a similar scenario but with the autonomy model. Without the autonomy model, the distracted driver bumps the car in front. From left to right, the figures show that

**TABLE 5** Results for test suites that include cars.

| | without the autonomy model (50 runs per suite) | | | | |
|---|---|---|---|---|---|
| **Suite Name** | **Successful runs** | **Remaining Distance to Destination point (meters)** | **Simulated Time (seconds)** | **Collided runs** | **timeouts** |
| ST01-02 | 26 | 3831 | 15061 | 24 | 0 |
| TT01-02 | 10 | 9428 | 40905 | 38 | 2 |
| ST02-02 | 22 | 1698 | 9481 | 28 | 0 |
| TT02-02 | 12 | 5268 | 22381 | 37 | 1 |
| ST01-03 | 27 | 4245 | 21400 | 22 | 1 |
| TT01-03 | 11 | 9269 | 80941 | 27 | 12 |
| ST02-03 | 22 | 1608 | 9601 | 28 | 0 |
| TT02-03 | 20 | 4906 | 29638 | 27 | 3 |
| ST01-04 | 25 | 4445 | 12422 | 25 | 0 |
| TT01-04 | 11 | 8830 | 57419 | 36 | 3 |
| ST02-04 | 22 | 1657 | 10455 | 28 | 0 |
| TT02-04 | 18 | 4798 | 30068 | 31 | 1 |
| ST01-05 | 24 | 4171 | 15704 | 26 | 0 |
| TT01-05 | 11 | 9370 | 41232 | 38 | 1 |
| ST02-05 | 20 | 1634 | 10101 | 30 | 0 |
| TT02-05 | 16 | 4859 | 20308 | 34 | 0 |
| **TOTAL** | **297** | **80 Km** | **118,7 Hours** | **479** | **24** |
| | **with the autonomy model (50 runs per suite)** | | | | |
| ST01-02 | 24 | 4385 | 41821 | 21 | 5 |
| TT01-02 | 10 | 8898 | 128189 | 21 | 19 |
| ST02-02 | 22 | 1528 | 36474 | 19 | 9 |
| TT02-02 | 20 | 4665 | 93574 | 13 | 17 |
| ST01-03 | 25 | 4444 | 28201 | 21 | 4 |
| TT01-03 | 11 | 8967 | 119274 | 18 | 21 |
| ST02-03 | 22 | 1520 | 22390 | 24 | 4 |
| TT02-03 | 18 | 4757 | 65978 | 21 | 11 |
| ST01-04 | 27 | 4114 | 23067 | 22 | 1 |
| TT01-04 | 12 | 8658 | 116423 | 20 | 18 |
| ST02-04 | 22 | 1533 | 39006 | 18 | 10 |
| TT02-04 | 20 | 4440 | 74528 | 16 | 14 |
| ST01-05 | 24 | 4520 | 23540 | 25 | 1 |
| TT01-05 | 12 | 8389 | 94014 | 29 | 9 |
| ST02-05 | 23 | 1480 | 45046 | 14 | 13 |
| TT02-05 | 16 | 5030 | 58224 | 25 | 9 |
| **TOTAL** | **308** | **77 Km** | **280 Hours** | **327** | **165** |

the distracted driver is not reducing throttle and is not braking, until it ultimately collides with the car in front. Command FOLLOW in the figures stands for *Follow the lane* and it is an input to the trained agent to implement the objective of the run on reproducible routes.

The four images of Figure 14 show the state transition amongst the different autonomy level, following the state machine of Figure 4. Throttle is progressively reduced, independently on the decisions of the driver, and the car stops in time.

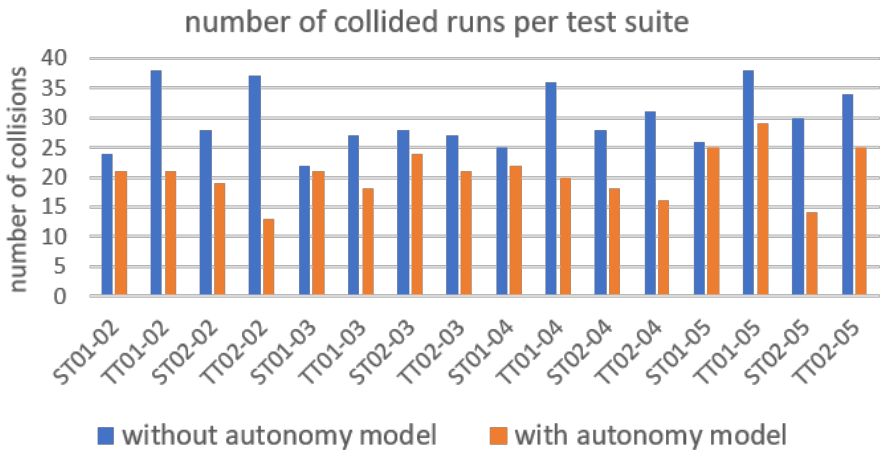



**FIGURE 12** Total collided runs per test suite with and without the autonomy model.

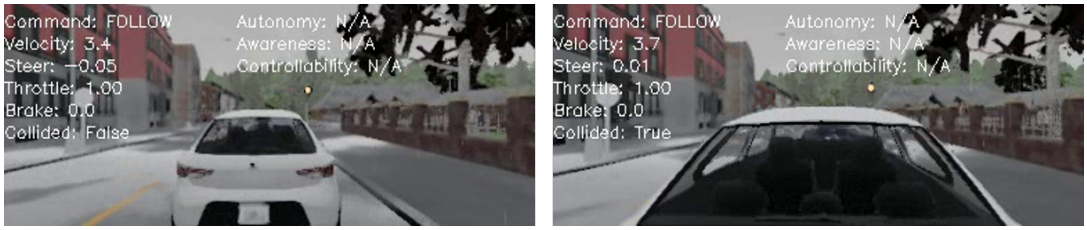
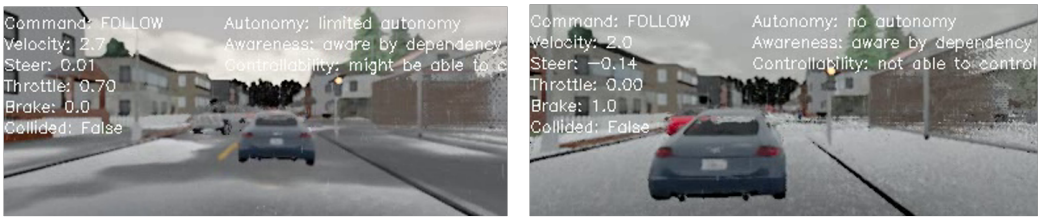



**FIGURE 13** Sample bump scenario when the autonomy model is not enabled.


a) No vehicles are close. Throttle is decided in full autonomy by the vehicle

b) There is a vehicle in front, throttle is reduced to 0.80 (correcting the decision of the driver)




c) There is a vehicle in front, throttle is reduced to 0.70 (correcting the decision of the driver)

d) There is a vehicle very close in front, throttle is reduced to 0.00 and brakes are activated.

**FIGURE 14** Using the autonomy model, the car stops in time.

## 6 | RELATED WORK

The autonomy literature is vast and covers a wide range of aspects, from autonomy in the biological systems to robotics. Here we restrict ourselves to the aspects that are relevant to our work, specifically about robotic systems and adjustable autonomy in complex systems. Several authors are dealing with autonomy in their works. For instance, some authors refer to the autonomy as the ability to operate without human intervention [27,28]. Other authors put emphasis on the ability of a system to change its behavior in response to unanticipated events during the operation [29,30]. The concept of adaptive autonomy is not new and has been introduced by various authors (e.g., [31,32,33]). For example, Danks et al. [31] consider autonomy as the ability to identify contexts in a changing environment and then select and execute appropriate functions in ways that conform to relevant and potentially context-specific norms, constraints, or desiderata.

On the other hand, Castelfranchi [21] considers the autonomy concept as the relation between autonomy from physical context, like environment, and autonomy from social context, i.e. from other agents. In our work, we have particular interest when the autonomy of CPSs can be adapted according to certain changes in their operational environment as well as their dependencies on one another. In [33], autonomy has been defined in terms of dependence theory, and several dimensions of autonomy have been discussed. Moreover, Falcone and Castelfranchi [34] propose to analyze the adjustable autonomy both by considering the level of delegation allowed to the subsystem and the possibility of this subsystem to adjust itself its own autonomy by expanding or restricting the received delegation level. Thus, one could distinguish between (i) performance or executive autonomy, where the agent is not allowed to decide anything but the execution of the entire delegated plan; (ii) planning autonomy, where the agent is allowed to plan by itself, to choose its own plan to obtain the goal delegated; (iii) goal autonomy (the agent is allowed to have/find goals).

Schumann et al. [32] present the first steps towards a dynamic regulation of autonomous decision taking in order to design adaptive decision support systems that try to solve the conflict between local autonomy and global system performance. Ning Tan et al. [35] propose a framework for robot-inclusive environments. The framework is based on the robot-inclusiveness concept, a taxonomy for robot-environment interaction, and design criteria to develop robot-inclusive environments to support autonomous robots. Therefore, the autonomy level is related to robot-inclusiveness level of the environment.

Collaborative autonomy between high-level behaviors and human operators have been proposed in [36]. In this work, a framework called FlexBE is used to provide autonomy of robots collaborating with human operators. More specifically, autonomy is realized as high-level control approach, where a human operator is responsible to adjust the autonomy level of the robots. In addition, Michal A. Goodrich et al. [37] propose a system that is designed to allow a robot to be situated in the environment and to initiate its own responses based on its perceptions. The authors propose five levels of autonomy: fully autonomous, goal-based autonomy, waypoints and heuristics, intelligent teleoperation, and dormant.

Salado [38] address the implications of intended abandonment, when a CS can intentionally leave the SoS. He argues this is a natural consequence of the inherent belonging and autonomy characteristics of CSs. Therefore, it is necessary to define and measure the operational effectiveness of the SoS to deal with the risk of abandonment. Finally, Brooks [39] describes a layered control system for an intelligent autonomous mobile robot. In his work, Brooks defines levels of competence for autonomous robots. For example, level 1 competence is to avoid contact with objects. Once achieving level 1 competence it is possible to add level 2 competence on top of level 1, where the process goes on until the desired level of competence is achieved, which is called subsumption architecture.

Liu et al. [40] present a review on variable autonomy of unmanned systems. They present a classification of variable autonomy systems based on adjustable autonomy, adaptive autonomy, and mixed-initiative decision, when the decision-making is made by users, computer, and both user and computer, respectively. They identify four main open problems in the field: system structure, perception model, user modeling, decision-making method for autonomy level. Zhao et al. [41] proposed an adjustable autonomy method for human-UAVs collaboration based on Fuzzy Cognitive Maps (FCM) to adjust the four levels of autonomy. The adjustable autonomy is realized by the ALFUS framework to quantify the autonomy based on three contextual metrics: mission complexity, environment complexity, and human interaction.

Horcas et al. [42] introduce a reconfigurable vehicle controller agent for autonomous vehicles based on context information, that in this work is weather conditions. They use the two parameters time headway and acceleration to predict the behavior of the autonomous vehicles under normal conditions, light rain, and heavy rain situations. Hoffmann et al. [43] argue that a combination of traditional control theory with learning strategies, what they call computational embodiment and computational self-awareness, is needed to dynamically adapt goals and actions. This can be achieved by automated abstracted observations at run-time and dynamic learning, for context-dependent adaptation.

Similarly with the works described above, we consider the dynamic regulation for the autonomy in CPSoS. Nevertheless, differently from the existing solutions, we propose to define the level of autonomy for each CPS not only based on either its type of awareness concerning its operational environment or the level of controllability for the activity to be performed. In fact, we propose to define the autonomy level for CPS as a combination of both of these attributes. Additionally, we have provided a model-based approach that can be used for modeling and analyzing the autonomy levels for CPSs.

## 7 | CONCLUSIONS AND FUTURE WORK

In a Cyber-Physical System-of-Systems, the autonomy of individual Cyber-Physical Systems may result in a lack of coordination, while perusing their own goal and the overall goals of the System-of-Systems. In this paper, we advocate that in order to facilitate the integration of Cyber-Physical Systems within the overall context of their CPSoS, we may need to adjust their level of autonomy in a way that enables them to coordinate their activities to avoid any conflict among one another.

To such extent, we formulate in this paper a conceptual model where the autonomy level is derived from the ability of the user to control its action, and from its awareness on the environment. Moreover, we propose a model-based approach, supported by a UML profile, for modeling and analyzing the autonomy level of Cyber-Physical Systems based on their type of awareness of their operational environment as well as their controllability concerning the activities they aim to perform. In addition, we illustrated the application of the proposed approach with simulations concerning driving in a town with other cars and pedestrians. Programming the simulator, we are able to represent a car whose control system implements the relations and criteria as established in the UML profile to decide and adjust its autonomy level, depending on its awareness of the environment as well as its level of controllability of its activities. Simulations show that the number of collisions is reduced when autonomy concepts as presented in this paper are in place.

The guiding example used in this paper is from the automotive domain, but it was chosen carefully to include the main aspects of a Cyber-Physical System-of-Systems with respect to the autonomy level of Cyber-Physical Systems based on awareness and controllability. In fact, the model and the corresponding profile are formulated and build in general terms, and are not specific to the automotive domain. Therefore, the approach can be applied to other Cyber-Physical Systems (e.g., rail transport systems, smart grid, Industry 4.0, etc.) as well, in all cases where the autonomy level of some of their Cyber-Physical Systems might need to be adjusted to guarantee their safe performance while pursuing their activities.

For the future work, we are planning to extend the conceptual model by integrating the notion of governance within the CPSoS design, which will allow us to define rules that can be used for clearly specifying who can adjust the autonomy of a CPS as well as how and when it can be adjusted. In particular, we plan to incorporate the concepts of *governance* and *control* into the proposed model, where the first can be used as a basis for specifying the later. Moreover, we intend to further investigate other aspects that might influence the autonomy levels of CPSs, and extend our approach accordingly to cover the new extensions in the conceptual model. We also aim to better validate our approach by applying it to several case studies from various domains (e.g., Industry 4.0).

## ACKNOWLEDGMENT

This work has been partially supported by the project H2020-MSCA-RISE-2018 ADVANCE (Addressing Verification and Validation Challenges in Future Cyber-Physical Systems).

## References

1. M. Jamshidi, “System of systems engineering - new challenges for the 21st century,” *Aerospace and Electronic Systems Magazine, IEEE*, vol. 23, no. 5, pp. 4–19, 2008.
2. M. W. Maier, “Architecting Principles for Systems-of-Systems,” *INCOSE International Symposium*, vol. 6, no. 1, pp. 565–573, jul 1996.
3. A. Bondavalli, S. Bouchenak, and H. Kopetz, *Cyber-physical systems of systems : foundations – a conceptual model and some derivations: the AMADEOS legacy*. Springer, 2016.
4. M. Gharib, L. Da Silva, H. Kavalionak, and A. Ceccarelli, “A Model-Based Approach for Analyzing the Autonomy Levels for Cyber-Physical Systems-of-Systems,” in *Proceedings - 8th Latin-American Symposium on Dependable Computing, LADC*. IEEE, 2018, pp. 135-144.
5. S. Levin and J. C. Wong, “Self-driving Uber kills Arizona woman in first fatal crash involving pedestrian | Technology | The Guardian,” p. 1, 2018. [Online]. Available: https://www.theguardian.com/technology/2018/mar/19/uber-self-driving-car-kills-woman-arizona-tempe
6. P. Dourish and V. Bellotti, “Awareness and Coordination in Shared Workspaces,” in *Proc. Intl. Conf. on Computer-Supported Cooperative Work*, no. November. ACM, 1992, pp. 107–114.
7. T. A. Faculty, W. K. Edwards, and I. P. Fulfillment, “Coordination Infrastructure in Collaborative Systems,” Ph.D. dissertation, Citeseer, 1995.
8. ISO, “26262: Road vehicles-Functional safety,” *International Standard ISO/FDIS*, vol. 26262, 2011.

9. OMG-OCL, "Object Constraint Language," Tech. Rep. May, 2014. [Online]. Available: http://www.omg.org/spec/OCL/2.4/

10. T. F. Golob and W. W. Recker, "Relationships Among Urban Freeway Accidents, Traffic Flow, Weather, and Lighting Conditions," *Journal of Transportation Engineering*, vol. 129, no. 4, pp. 342–353, jul 2003.

11. B. Whiffen, P. Delannoy, and S. Siok, "Fog : Impact on Road Transportation and Mitigation Options," *National Highway Visibility Conference*, pp. 1–12, 2004.

12. M. Abdel-Aty, A.-A. Ekram, and H. Huang, "A Study on Visibility Obstruction Related Crashes Due To Fog and Smoke," *12th World Conference for Transportation Research*, pp. 1–17, 2010.

13. G. Hegeman, R. Van Der Horst, K. Brookhuis, and S. Hoogendoorn, "Functioning and acceptance of overtaking assistant design tested in driving simulator experiment," *Transportation Research Record*, vol. 2018, pp. 45–52, dec 2007.

14. W. Birk, E. Osipov, and J. Eliasson, "iRoad—Cooperative Road Infrastructure Systems for Driver Support," *World Congress and Exhibition on Intelligent Transport Systems and Services*, 2009.

15. M. Gharib, P. Lollini, and A. Bondavalli, "A conceptual model for analyzing information quality in System-of-Systems," in *2017 12th System of Systems Engineering Conference, SoSE 2017*. IEEE, 2017, pp. 1–6.

16. S. Rogers, C.-N. Fiechter, and P. Langley, "An adaptive interactive agent for route advice," in *Proceedings of the third annual conference on Autonomous Agents*. ACM, 1999, pp. 198–205.

17. J. J. Odell, H. V. D. Parunak, M. Fleischer, and S. Brueckner, "Modeling agents and their environment," in *Agent-oriented software engineering III*. Springer, 2003, pp. 16–31.

18. C. E. Martin and K. S. Barber, "Multiple, simultaneous autonomy levels for agent-based systems," in *Proceedings of the fourth international conference on control, automation, robotics, and vision*. Citeseer, 1996, pp. 1318–1322.

19. V. R. Lesser, "Cooperative multiagent systems: A personal view of the state of the art," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 11, no. 1, pp. 133–142, 1999.

20. M. Luck and M. D'Inverno, "A Formal Framework for Agency and Autonomy." in *ICMAS*, vol. 95, 1995, pp. 254–260.

21. C. Castelfranchi, "Guarantees for autonomy in cognitive agent architecture," in *Intelligent agents*. Springer, 1995, pp. 56–70.

22. , "Founding Agents' Autonomy on Dependence Theory," in *The 14th European Conference on Artificial Intelligence (ECAI )*, vol. 1, 2000, pp. 353–357.

23. J. R. FRENCH, "A Formal Theory of Social Power," *Social Networks*, vol. 63, no. 3, pp. 35–48, 1956.

24. M. Gharib, P. Lollini, A. Ceccarelli, and A. Bondavalli, "Dealing with Functional Safety Requirements for Automotive Systems: A Cyber-Physical-Social Approach," in *The 12th International Conference on Critical Information Infrastructures Security (CRITIS)*. Lucca: Springer International Publishing, 2017.

25. M. Gharib, P. Lollini, M. Botta, E. Amparore, S. Donatelli, and A. Bondavalli, "On the Safety of Automotive Systems Incorporating Machine Learning based Components: A Position Paper," in *International Conference on Dependable Systems and Networks Workshop (DSN-W)*. IEEE. 2018.

26. B. Selic, "A Systematic Approach to Domain-Specific Language Design Using UML," in *Proceedings of the 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'07)*. IEEE, may 2007, pp. 2–9.

27. M. Machin, J. Guiochet, H. Waeselynck, J.-p. Blanquart, M. Roy, and L. Masson, "SMOF: A Safety Monitoring Framework for Autonomous Systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, no. 99, pp. 1–14, 2016.

28. J. M. Beer, A. D. Fisk, and W. A. Rogers, "Toward a Framework for Levels of Robot Autonomy in Human-Robot Interaction," *Journal of Human-Robot Interaction*, vol. 3, no. 2, p. 74, 2014.

29. D. P. Watson and D. H. Scheidt, "Autonomous systems," *Johns Hopkins APL technical digest*, vol. 26, no. 368-376, p. 9, 2005.

30. S. Nahavandi, "Trusted Autonomy Between Humans and Robots: Toward Human-on-the-Loop in Robotics and Autonomous Systems," *IEEE Systems, Man, and Cybernetics Magazine*, pp. 10–17, 2017.

31. D. Danks and A. J. London, "Regulating autonomous systems: Beyond standards," *IEEE Intelligent Systems*, vol. 32, no. 1, pp. 88–91, 2017.

32. R. Schumann, A. Lattner, and T. Zur, "Regulated Autonomy: A Case Study" *Proceedings of the Conference Track "Intelligente Systeme zur Entscheidungsunterstutzung" at the Multikonferenz Wirtschaftsinformatik (MKWI 2008), Munich, Germany, SCS Publishing House*, pp. 83–98, 2008.

33. C. Castelfranchi, "Founding agent's' autonomy'on dependence theory," in *Proceedings of the 14th European Conference on Artificial Intelligence*, 2000, pp. 353–357.

34. R. Falcone and C. Castelfranchi, "Levels of Delegation and Levels of Adoption as the basis for Adjustable Autonomy," in *Congress of the Italian Association for Artificial Intelligence*. Springer, 1999, pp. 273–284.

35. N. Tan, R. E. Mohan, and A. Watanabe, "Toward a framework for robot-inclusive environments," *Automation in Construction*, vol. 69, pp. 68–78, 2016.

36. D. Conner, S. Kohlbrecher, P. Schillinger, A. Romay, A. Stumpf, S. Maniatopoulos, H. Kress-Gazit, and O. von Stryk, "Collaborative autonomy between high-level behaviors and human operators for control of complex tasks with different humanoid robots," *Springer Tracts in Advanced Robotics*, vol. 121, 2018.

37. J. Crandall and M. Goodrich, "Experiments in adjustable autonomy," *IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*, vol. 3, pp. 1624–1629, 2001.

38. A. Salado, "Abandonment: A natural consequence of autonomy and belonging in systems-of-systems," in *2015 10th System of Systems Engineering Conference, SoSE 2015*, 2015, pp. 352–357.

39. R. A. Brooks, "A Robust Layered Control System for a Mobile Robot. AI Memo 864," *ieeexplore.ieee.org*, 1985.

40. G. Liu, Q. Zou, J. Wu, H. Zhang and R. Zhang, "A Review on Variable Autonomy of Unmanned Systems," *2018 Chinese Automation Congress (CAC)*, Xi'an, China, 2018, pp. 2253-2258.

41. Z. Zhao, C. Wang, Y. Niu, L. Shen, Z. Ma and L. Wu, "Adjustable Autonomy for Human-UAVs Collaborative Searching Using Fuzzy Cognitive Maps," *2019 2nd China Symposium on Cognitive Computing and Hybrid Intelligence (CCHI)*," Xi'an, China, 2019, pp. 230-234.

42. Horcas, JM, Monteil, J, Bouroche, M, Pinto, M, Fuentes, L, Clarke, S. "Context-dependent reconfiguration of autonomous vehicles in mixed traffic.," *J Softw Evol Proc*. 2018; 30:e1926.

43. H. Hoffmann, A. Jantsch and N. D. Dutt, "Embodied Self-Aware Computing Systems," *Proceedings of the IEEE*. 2020.

44. Dosovitskiy, Alexey and Ros, German and Codevilla, Felipe and Lopez, Antonio and Koltun, Vladlen, "CARLA: An Open Urban Driving Simulator", *Conference on Robot Learning*, pp1–16, 2017.

45. Dian Chen and Brady Zhou and Vladlen Koltun and Philipp Krähenbühl, Learning by Cheating, Conference on Robot Learning (CoRL), 2019

46. Ceccarelli, Andrea and Bondavalli, Andrea and Froemel, Bernhard and Hoeftberger, Oliver and Kopetz, Hermann, "Basic Concepts on Systems of Systems", Cyber-Physical Systems of Systems: Foundations – A Conceptual Model and Some Derivations: The AMADEOS Legacy, Bondavalli, Andrea and Bouchenak, Sara and Kopetz, Hermann (Eds.), Springer International Publishing, 2016, pp. 1–39

47. Kopetz, Hermann, et al. "Emergence in Cyber-Physical Systems-of-Systems (CPSoSs)." Cyber-Physical Systems of Systems. Springer, Cham, 2016. 73-96.

48. Unreal Engine, www.unrealengine.com

49. Logs and videos of the runs https://drive.google.com/open?id=1qmnB5SgBGWytGAWYoTeOfBcQW7o7Isnx

50. Carla 0.9.6 API https://carla.readthedocs.io/en/latest/python_api/