

Dealing with Uncertainty in Software Architecture on the Internet-of-Things with Digital Twins

Flavio Oquendo(✉)

IRISA – UMR CNRS 6074, University Bretagne Sud, Vannes, France
flavio.oquendo@irisa.fr

Abstract. When architecting Software-intensive Systems-of-Systems (SoS) on the Internet-of-Things (IoT), architects face two sorts of uncertainties. First, they have only limited knowledge about the operational environment where the SoS will actually be deployed. Second, the constituent systems which will compose the SoS might not be known a priori (at design-time) or their availability (at run-time) is affected by dynamic factors, due to the openness of the IoT. The consequent research question is thereby how to deal with uncertainty in the design of an SoS architecture on the IoT. To tackle this challenging issue, this paper addresses the notion of uncertainty due to partial information in SoS and proposes an enhanced SoS Architecture Description language (SosADL) for expressing SoS architectures on the IoT under uncertainty. The core SosADL is extended with concurrent constraints and the concept of digital twins coupling the physical and virtual worlds. This novel approach is supported by an integrated toolset, the SosADL Studio. Validation results demonstrate its effectiveness in an SoS architecture for platooning of self-driving vehicles.

Keywords: Architectural design uncertainty · Software architecture description · Systems-of-Systems (SoS) · Digital twins · Internet-of-Things (IoT)

1 Introduction

Uncertainty [29] is an intrinsic characteristic of complex systems, may it be human, natural, or engineered systems [33]. Indeed, complex systems are by essence unpredictable, thereby raising different types of uncertainties. This is particularly the case in the design of engineered systems, where uncertainty has been a major complicating factor.

Uncertainty may be classified as aleatory or epistemic. Aleatory uncertainty (a.k.a. as stochastic or objective uncertainty) results from uncontrollable phenomena that are uncertain by nature. Uncertainty is due to the inherent randomness of these phenomena. It is irreducible, and is generally expressed, in probability theory, by random variables or stochastic processes. Epistemic uncertainty (a.k.a. systematic or subjective uncertainty) results from the lack of knowledge about identified phenomena. Uncertainty is due to partial information. It is reducible, i.e. can be reduced by acquiring more refined data about the phenomenon.

© Springer Nature Switzerland AG 2019
S. Misra et al. (Eds.): ICCSA 2019, LNCS 11619, pp. 770–786, 2019.
https://doi.org/10.1007/978-3-030-24289-3_57

In the last three decades, research on design under uncertainty has received increasing attention, especially in Systems Engineering [1], and in recent years in Software Engineering [35]. Both types of uncertainty have been addressed, in particular aleatory uncertainty has been extensively studied.

Regarding Software Engineering, uncertainty is mainly associated to the design of software-intensive systems, and more recently of Software-intensive Systems-of-Systems (SoS) [15, 23] on the Internet-of-Things (IoT) [32]. Indeed, due to the open and very dynamic nature of the IoT, when architecturally designing an SoS on the IoT, an architect does not know which will be the actual IoT systems that will become constituents of the SoS, these being predominantly discovered at run-time. Thereby, the architecture description needs to deal with the uncertainty of the presence or absence of the required constituent systems. Furthermore, even for those that are present, their inherent independence magnifies uncertainty.

Moreover, the correct SoS architecture on the IoT depends not only on the constituent IoT systems but also, largely, on the operational environment where the SoS is deployed. Actually, an operational environment enabled by the IoT, where an SoS will perform at run-time, is inherently incompletely known at design-time [12], thereby information available at design-time is only partial.

The challenge, in the description of an SoS architecture on the IoT, is to be able to handle the uncertainty raised by partial information on the constituent systems which will concretely operate in the SoS at run-time as well as on the characteristics of the concrete operational environment where the SoS will actually operate.

Therefore, from the perspective of the software architecture, the key research question is how to deal with uncertainty in the description of an SoS architecture on the IoT.

This is particularly a hard question to answer because IoT is an open-world, thereby it is not feasible to exhaustively anticipate and resolve all possible situations that an SoS will face. These open environments expose the SoS to a myriad of cases that are impossible to predict.

To address this challenging issue, this paper examines the notion of uncertainty in SoS and presents the features that we have added to SosADL, a novel formal SoS Architecture Description Language (ADL), to be able to deal with uncertainty in the architectural design of SoSs on the IoT, enabling the description of SoS architectures that operate with only partially known IoT constituent systems in partially known environments on the IoT. In particular, in the released *SosADL for IoT under Uncertainty*, we leveraged the notion of digital twin [9] (replica of the physical counterpart of an IoT system or "thing" on the IoT) as the core architectural concept for representing SoS architectures on the IoT at design-time, and self-managing the SoS architectures once deployed on the IoT at run-time. In our approach, digital twins of "things" on the IoT coordinate their operation with other digital twins on the IoT in a shared operational environment based on the expression and enforcement of concurrent constraints.

This paper relies on previous publications on SosADL, which presented its concepts and notation [19] and its underlain formal theory [20]. It builds on the approach for copying with uncertainty in SoSs [26], on the mechanisms for raising emergent behaviors [22, 24], and on principles for self-organization [25].

This paper adds to these, the presentation of the SosADL enhancements bringing contributions beyond the state-of-the-art for dealing with uncertainty when architecting SoSs on the IoT, resulting in the novel *SosADL for IoT under Uncertainty*.

The remainder of this paper is organized as follows. Section 2 introduces a motivating case study for dealing with uncertainty in SoS on the IoT. Section 3 presents the enhancements to SosADL that enables the handling of uncertainty on the IoT. Section 4 demonstrates how the enhanced SosADL can be applied for describing SoS architectures on the IoT under uncertainty. In Sect. 5 we briefly present the implementation of SosADL Studio, now supporting uncertainty. In Sect. 6, we position our proposal for dealing with uncertainty in SoS architectural design with related work. To conclude, we summarize in Sect. 7 the main contributions of this paper and outline future work.

2 Motivating Case Study for Dealing with Uncertainty in SoS

For motivating the problematics of architectural design under uncertainty of SoSs on the IoT, let us introduce hereafter an excerpt of a pilot study [21] we have carried out to concretely present the issues of uncertainty and set the basis for the presentation of *SosADL for IoT under Uncertainty*.

We will focus on the Platooning SoS, part of emergency response, and its intended emergent behavior to create and maintain platoons of self-driving vehicles through self-organization under uncertainty in the framework of the IoT for Vehicles, a.k.a. Internet-of-Vehicles (IoV) [32]. Platooning is the process of automated vehicles (in this case self-driving vehicles) autonomously forming road convoys.

2.1 Essentials for Architecting Platoons of Self-Driving Vehicles on the IoT

The constituent systems of the Platooning SoS are self-driving vehicles, which embody radars (in particular lidars, light detection and ranging), sensors, driverless steering, and Vehicular Ad-hoc Networks (VANETs), depicted in Fig. 1 (left). Using these devices, the self-driving vehicle can sense information from the physical operational environment and from other vehicles, process this information, communicate it to other vehicles through VANETs as well as feed it to drivers (if any).

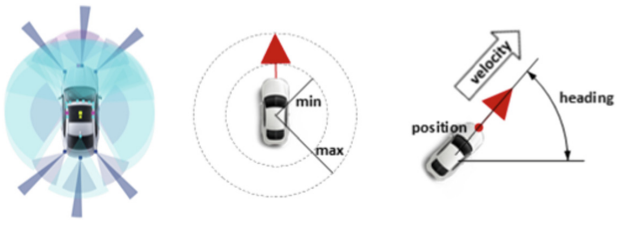

Fig. 1. Automated vehicles: (left) radars/lidars, sensors and VANETs; (middle) min safe separation and max perception range; (right) velocity in terms of speed and heading

As also depicted in Fig. 1 (middle), there are two critical zones around a vehicle: the minimal safe separation from vehicles in front or behind it and in general for any kind of obstacle (the min safe distance) and the maximal separation (the max perception range of the radars/sensors and of VANETs enabling to communicate with other vehicles in front or behind it).

As again depicted in Fig. 1 (right), each vehicle is equipped with GPS for positioning and with controls for heading and speed to determine and steer its velocity. These data may be transmitted via VANETs to the neighboring vehicles. In addition to real time position in terms of GPS coordinates, a connected vehicle may also share the GPS coordinate of its destination.

When platooning, a group of vehicles can safely travel closely together like in a convoy [13], as shown in Fig. 2.

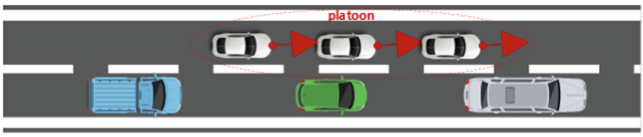

Fig. 2. Vehicles in a platoon: virtual leader and followers

More precisely, platooning is the process of vehicles autonomously following a leader to form a road convoy. It requires that each vehicle in the platoon control its velocity and the relative distance to the vehicle in front of it, and possibly also to the one behind of it. This is performed by interacting through radars/sensors as well as coordinating through VANETs.

2.2 Uncertainty in the Self-Driving Vehicle Platooning SoS

Let us now address an uncertainty that must be dealt with in Platooning SoSs, i.e. uncertainty due to imprecision (a form of partial information): on-board sensors in a vehicle have limited sensing capability and range, thereby providing imprecise measures. Also, environmental noise can alter the precision of the sensed information. Also, in VANETs, communication delay may affect the accuracy of the transmitted information.

It means that an SoS architect, while designing an SoS architecture for platooning needs to take into account that the sensed distance from a self-driving vehicle to the platoonmate in front of it is approximate. The SoS architect also needs to take into account that the min safe separation distance and the max perception range (shown in Fig. 1) may vary according to road surface and weather conditions (therefore they are uncertain). Moreover, the min safe separation distance varies with the speed. For instance, a vehicle needs ca. 14 m to fully stop (avoiding collision with the vehicle in front of it) if its speed is of 50 km/h once brakes are pressed down in dry weather on a well-maintained road pavement. It is of ca. 38 m if the speed is of 80 km/h and is of ca. 75 m if the speed is of 110 km/h and so on.

Indeed, for guaranteeing collision avoidance in the platoon, i.e. no collision at any time, the platoonmates must follow each other in a distance that is safe for braking to avoid the vehicle that is driving in front of it (considering the different uncertainties in sensed or received information). Moreover, in platooning, the distance between a platoonmate and the vehicle in front of it must be sufficient to discourage interference from other vehicles, i.e. avoiding to have other vehicles (not participating in the platoon) entering in-between two platoonmates causing the undesired split of the platoon.

The architecture of the Platooning SoS needs thereby to specify how different self-driving vehicles (unknown at design-time) will form platoons (different platoons may be formed), while coping with these different uncertainties, in particular related to the distance of neighboring self-driving vehicles. The platoon must make coordinated movements correctly and safely, being guaranteed collision free.

3 Extending SosADL for IoT Under Uncertainty

Over the years, many ADLs have been conceived for precisely describing software architectures where every aspect is known at design-time regarding both its constituents and the operational environment. As studied in [14], no current ADL is able to handle the uncertainty required for describing SoS architectures.

To fill this gap, we extended SosADL with new conceptual abstractions and constructs for describing SoS architectures on the IoT while dealing with uncertainty, resulting in the novel *SosADL for IoT under Uncertainty*.

3.1 Enhancing SosADL with Digital Twins for Architecting SoS on the IoT

To fill the identified gap, the first question to investigate is: what are the architectural abstractions suitable for architecting SoS on the IoT?

Conceptually, the IoT is essentially the Internet of "things" which exist in the physical world and are represented in the virtual world. A "thing" is formed of devices (sensors and actuators) which interact with the physical world through sensing and actuating as well as of a controller for managing its own behavior based on its devices, including networking devices (which may be for connecting to wireless or wired networks, mobile ad hoc or fixed, local or global). In fine, a "thing" or IoT system provides the basic functionalities or services of the IoT.

There are different notions and reference architectures for designing and implementing software-intensive systems or SoSs on the IoT. Among these, we adopted the notion of digital twin [9], which fits well the needs for coupling the physical and virtual worlds in terms of "things" or IoT systems, as required in SoSs on the IoT.

In fact, the notion of "twin" has been applied since the seventies by the NASA in the Apollo Program, which replicated the spacecrafts in order to solve technical problems during their mission. Recently, the technological progress has made possible the creation of an evolved "twin" notion, i.e. the "digital twin" (sometimes also called digital shadow). A digital twin provides a digital replica (in the Edge, Fog, or the Cloud) of its physical counterpart which is virtually indistinguishable from its physical

twin (in the sense that it dynamically replicates the behavior and properties of the physical twin as well as enriches the replica with additional information about the physical counterpart). The digital twin is indeed the dynamic digital representation of its physical counterpart. It can thereby be used to represent the physical counterpart during the design and operation of the SoS architecture on the IoT. In the virtual world, the digital twin represents the physical twin, which resides in the physical world. The digital twin is thereby a relevant architectural abstraction that is virtually indistinguishable from its physical counterpart.

The conceptual model of a digital twin contains three main parts: (i) physical twin in the real world; (ii) digital twin in the virtual world; and (iii) the connections of input data and output command that ties the digital and real counterparts together.

In general, the digital twin monitors the physical twin through data provided by sensors as well as information coming from human domain experts to maintain a utility replica of the physical entity. It acts back on the physical twin through actuators.

Digital twins have mostly been applied in the context of Industrial IoT [7]. In particular, in Industry 4.0, digital twins generally support simulation of the behaviors of their physical twins, in real-time, and sometimes are supported by augmented reality.

Also note that digital twin is a notion that has variation in terms of meaning and scope, which often depend of their purpose.

In SosADL for IoT, we apply the notion of digital twin as a concept for the architectural design of an SoS as well as a concept to dynamically reconfigure the architecture during the operation of the SoS.

Architecturally speaking, in SosADL for IoT, a digital twin is a digital replica of a real-world "thing" on the IoT aiming to provide a dynamic virtual representation of its physical counterpart. The concept of digital twin thereby drives the design of SoS architectures on the IoT by coordinating the digital twins of their constituent systems.

In the SoS architecture description, what are represented are therefore digital twins. In an SoS architecture description with SosADL for IoT:

- digital twins of IoT constituent systems are SoS architectural elements defined by intention (declaratively in terms of abstract systems) and instantiated at run-time (concretized);
- digital mediators are connectors of SoS architectural elements in the virtual world, defined by intention (declaratively in terms of abstract mediators) and created at run-time (concretized by the SoS) to achieve a goal, part of an encompassing mission (note that their architectural role is to mediate the interaction of digital twins of IoT constituent systems for creating emergent behavior);
- digital coalitions are SoS architectural compositions in the virtual world of mediated digital twins of IoT constituent systems, defined by intention (declaratively in terms of possible systems and mediators as well as the policies for their on-the-fly compositions) and evolutionarily formed at run-time (concretized) to achieve an SoS mission in an operational environment.

Coming back to our motivating case study, the SoS architecture description on the IoT of Vehicles (IoV) will specify how an SoS architecture is able to create, on the fly, and maintain emergent behaviors from elementary connected vehicles (represented by their digital twins), where the actual vehicles are not known at design time.

The connected vehicles (which are mobile "things" on the IoT) are represented and managed in the virtual world while casually connected with the real world. Through digital coalitions, mediations among "things" are architected to produce emergent behaviors, as it is the case in platooning of self-driving vehicles.

In this way, the proposed digital twin notion supported by SosADL for IoT provides the basis to formally architect evolutionary SoS on the IoT. In particular, it supports reasoning of platooning properties in the virtual world and its control-command in the real world through the IoV.

3.2 Handling Uncertainty for Architecting SoS on the IoT

Now, the second question to investigate is: which theory is suitable for addressing the needs for dealing with uncertainty in IoT-enabled operational environments.

There are in fact several theories for handling uncertainty. Regarding the design of systems, the main ones are the theories of probability (associated to statistics and usually applied to address aleatory uncertainty), of possibility (related to fuzziness and usually applied to address epistemic uncertainty), and the interval mathematics (associated to imprecision, and that can also be applied to address epistemic uncertainty) [1]. As the research issue we are addressing in this work is essentially the one of imprecision due to uncertainty in the physical measures of the physical environment where the constituents of an SoS are operating, the theory that suits is the one of intervals.

There are in fact different interval mathematics [11]. The underlying principle of the interval theories is that of enclosing uncertain measured values in intervals. In our approach, we express intervals through constraints, e.g. to express the distance for the braking of a self-driving vehicle. Thereby, if the speed of a self-driving vehicle is 50 km/h, knowing that 14 m are needed to fully stop it (avoiding collision with the vehicle in front of it) in case of dry weather on a well-maintained road pavement, we represent the uncertainty related to road surface conditions by the constraint: "the min safe distance is between 12 m (in excellent-maintained road surfaces) and 18 m (in poorly-maintained road surfaces)". As weather also influences the min safe distance, if raining it becomes "between 14 m and 22 m" (ranging from excellent-maintained road surfaces to poorly-maintained ones).

More precisely, our approach to deal with uncertainty in SoS architecture description on the IoT is grounded on the constraint interpretation of intervals based on process algebra, the π-Calculus [16], and concurrent constraints [17].

Therefore, for dealing with uncertainty in SoS architectural design and especially in the description of designed SoS architectures on the IoT, SosADL was enhanced to enable the representation of partial information as well as to enable to reasoning on partial information to affect the emergent behavior of SoSs. By the decentralized nature of SoSs, these representation and reasoning mechanisms are expressed from the viewpoint of each digital twin of an IoT constituent system and of the digital mediators enabling their interactions (thereby influencing individual behaviors as well as the raising of emergent behaviors).

Based on the representation of uncertainty in terms of constraints, the subsequent question now is how these constraints can be manipulated by the concurrent IoT constituent systems of an SoS on the IoT.

Our approach to solve this question is two-fold: first, we extended the formal foundation of the core SosADL which is grounded on the π-Calculus [16] with concurrent constraints for representing and reasoning about partial information; second, we integrated three sorts of architectural abstractions to SosADL for IoT, i.e. digital twins, digital mediators, and digital coalitions, which manipulate concurrent constraints.

The novel π-Calculus for SoS [20], which extends the original π-Calculus with concurrent constraints, generalizes the π-Calculus with the notion of computing based on partial information.

In the π-Calculus for SoS, a behavior (of a digital twin of an IoT constituent system or of a digital mediator) is described by the actions that it can carry out as well as the constraints that it enforces or copes with. Two constructs have been added to SosADL to manipulate constraints: one is the *tell* construct which enables a digital entity to add a constraint to its operational environment in the virtual world, and the other is the *ask* construct which enables a digital entity to query if a constraint can be inferred from its operational environment in the virtual world. Other constructs have been extended to handle concurrent constraints, in particular the *compose* construct was enhanced to deal with concurrent composition of behaviors and constraints, and the *connection* construct to restrict the interface that a behavior can use to interact with others while enforcing constraints. The *untell* construct is also provided for removing a constraint from the digital environment.

Intuitively speaking, based on π-Calculus for SoS, a digital entity can *tell* to the digital environment about the pieces of information that it knows, while maintaining private information internally. A digital entity can also *ask* information from the digital environment that influences its own behavior or even *untell* previously told constraints.

4 Describing SoS Architecture for IoT Under Uncertainty

To demonstrate how *SosADL for IoT under Uncertainty* can be applied to architecturally describe SoS architectures on the IoT under uncertainty, we will present hereafter the architecture description of the Platooning SoS of self-driving vehicles introduced in the motivating case-study.

4.1 Emergent Behavior for the Platooning SoS of Self-Driving Vehicles

Fundamentally, in a Platooning SoS of self-driving vehicles, the emergent behavior of platooning results from three micro-scale behaviors of platoonmates (vehicles that are members of a platoon), as shown in Fig. 3, which together enforce the constraints that are required for enabling platoons (see [25] for details of self-organizing SoSs).

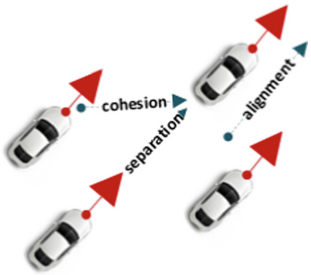

Fig. 3. Cohesion, separation, and alignment behaviors

The micro-scale behaviors for guaranteeing platooning are:

- *Cohesion behavior*: every platoonmate must steer to follow the platoonmate just in front of it (if any) while not losing the near platoonmate behind it (if any), thereby maintaining liaison;
- *Separation behavior*: every platoonmate must steer to avoid the near platoonmate in front of it, thereby avoiding rear collision;
- *Alignment behavior*: every platoonmate must steer to move towards the platoonmate in front of it, or towards the destination if no near platoonmate is in front of it while attempting to match velocity (heading and speed) with nearby platoonmates.

These micro-scale behaviors (cohesion, separation, and alignment), combined together determine the acceleration vector that drives a platoonmate. From the point of view of a vehicle, each micro-scale behavior engenders an independent request for a steering maneuver to be carried out by the self-driving vehicle.

Note that these micro-scale behaviors governing cohesion, separation, and alignment, constrained by the relative positions of nearby platoonmates, were demonstrated sufficient to guarantee the formation of platoons [31].

4.2 SoS Architecture Description with Uncertainty for UGV-Based Platooning

Now that we have introduced the three micro-scale behaviors providing the necessary and sufficient conditions to create and maintain the emergent behavior of vehicle platooning, let us demonstrate how to apply *SosADL for IoT under Uncertainty* to describe the SoS architecture of vehicle platooning under uncertainty using digital twins as well as explain the operational semantics of the resulting SoS architecture during the formation and maintenance of platoons. Recall that each self-driving vehicle has its own autopilot and may drive in autonomous mode to a destination.

We will first focus on the digital mediators enforcing the micro-scale behaviors required for creating and maintaining platoons (we will not present the interface of the digital twins of the IoT constituent systems of the platooning SoS, i.e. the self-driving vehicles, due to page limit). Then, we will describe the abstract architecture of the SoS on the IoT as a whole in terms of a digital coalition for achieving the macro-scale platooning behavior.

Let us now declare, in *SosADL for IoT under Uncertainty*, the digital mediator *Platooning* which will make possible the emergence of the platooning macro-scale behavior of the SoS of self-driving vehicles.

As shown in Listing 1, *Platooning* is described as a digital mediator abstraction: we declare the duties of the mediated self-driving vehicle in terms of position and command and the mediation among self-driving vehicles participating in the platoon as the steering behavior. By creating concretions, a digital mediator will be synthesized for each digital twin of a self-driving vehicle that participates in the fleet.

By the application of the three micro-scale behaviors, i.e. separation, alignment, and cohesion, commanding the digitally mediated vehicle, every self-driving vehicle in the platoon, will stepwise behave to align and get closer to neighboring self-driving vehicles, while avoiding collision.

```
//use user-defined library for route information
with RouteInformation
//declaring the platoon mediating behavior
digital mediator Platooning(max:Distance,min:Distance) is {
 //determining the position of the mediated vehicle
 duty position is{
  //inputting vehicle's latitude and longitude
  connection coordinate is in{Coordinate}
  //inputting vehicle's direction with heading angle
  connection heading is in{Heading}
  //inputting vehicle's velocity
  connection speed is in{Speed}}
 duty command is{
  //commanding for steering the mediated vehicle
  connection drive_ctrl is out{[heading:Heading,speed=Speed]}}
 abstraction separation(…) is {…}
 abstraction alignment(…) is {…}
 abstraction cohesion(…) is {…}
 behavior steering(…) is {…}
}
```

Listing 1. Digital mediator declaration in SosADL for IoT under Uncertainty

Note that a platoon may increase of size by having vehicles (which are not in the platoon) maneuvering to join the platoon or decrease of size by having vehicles in the platoon maneuvering to leave it.

Let us now describe the digital mediating behavior in Listing 2. Once the steering behavior receives the position of the mediated self-driving vehicle (firstly getting its coordinate, and secondly its heading), the *tell* construct asserts the received position into its local environment, enabling sharing with other digitally mediated self-driving vehicles. Next, the *ask* construct is used to ask for the positions of platoonmates that are in the perception range of the digitally mediated self-driving vehicle. With the result of the query stored, the digital mediator first checks if there is any. If not, i.e. if no other self-driving vehicle is in the neighborhood of the digitally mediated self-driving vehicle, the mediated self-driving vehicle continues to drive in the same direction, moving forward. If there are self-driving vehicles in the neighborhood of the digitally mediated self-driving vehicle, the platooning mediator behavior looks for the nearest platoonmate. It then checks whether the distance to the mediated self-driving vehicle is

less than the minimum separation value (*min*, passed in parameter). If it is the case, the steering behavior applies the separation micro-scale behavior. If it is not, the steering behavior commands the digitally mediated self-driving vehicle applying the alignment micro-scale behavior and then the cohesion micro-scale behavior based on the positions of the neighboring platoonmates.

Let us now describe the SoS architecture for forming the platooning in Listing 3. It is described by declaring first, the digital twins of SoS constituent systems (they are the digital entities of the IoT constituent systems that can participate in the SoS), second, the digital mediators (they are created and managed for coordinating the constituent systems via their digital twins), and third, the digital coalitions that can be formed to achieve the SoS emergent behavior of vehicle platooning.

```
behavior steering() is {
 repeat {
  //get the coordinate of the mediated self-driving vehicle
  via position::coordinate receive vehicle_coordinate
  //get the heading of the mediated self-driving vehicle
  via position::heading receive vehicle_heading
  //get the speed of the mediated self-driving vehicle
  via position::speed receive vehicle_speed
  //tell the position to other digital mediators
  tell vehicle is [coordinate=vehicle_coordinate,
   heading=vehicle_heading,speed=vehicle_speed]
  //ask if any vehicle is in the neighborhood
  value neighbors is ask vehicle suchthat {
   //in the radar range of the self-driving vehicle, i.e. < max
   vehicle::coordinate::distance(vehicle_coordinate)<max}
  //if there are vehicles in the radar range
  if (neighbors::notEmpty()) then {
   //identify nearest neighbor in front
   value front_mate is neighbors::nearest_in_front()
   //identify nearest neighbor in back
   value back_mate is neighbors::nearest_in_back()
   //if too close in front, i.e. < min separation
   if (vehicle_coordinate::distance(front_mate::coordinate)<min)
    then //separate to avoid collition
     separation(vehicle,front_mate)
    else {//align by velocity matching
     alignment(vehicle,front_mate,back_mate)
     //maintain nearest mates in range
     cohesion(vehicle,front_mate,back_mate)}
  }
}
```

Listing 2. Platoon mediating behavior in SosADL for IoT under Uncertainty

The SoS architecture description, shown in Listing 3, comprises the declaration of a sequence of digital twins of IoT constituent systems complying with the system abstraction of self-driving vehicle and a sequence of digital mediators conforming with the mediator abstraction of *Platooning* (as declared in Listing 1).

Based on the digital twins of these systems and mediator abstractions, the digital coalition for creating emergent behavior is declared, named as *platoon*, as shown in Listing 3. In particular, the digital coalition of self-driving vehicles is described as a sequence of digital twins of self-driving vehicles where each digital twin has an

associated steering digital mediator created in the digital coalition, with the specified perception range distance (*max*) and minimum separation distance (*min*) as parameters. The emergent behavior of the digital coalition, i.e. *platoon*, is produced by the macro-scale behavior created by supervenience from the mediating behaviors, which apply the micro-scale behaviors according to each situation.

It is indeed by the application of the mediating behaviors, commanding the digitally mediated self-driving vehicles by applying the defined micro-scale behaviors, that every self-driving vehicle in the fleet, will stepwise behave to align and get closer to neighboring vehicles, while avoiding collision. At the macro-scale level, the fleet of self-driving vehicles will progressively form and maintain the platoon.

It is worth noting that the platoon which is formed is the result of the application of the micro-scale behaviors to each self-driving vehicle in the platoon via its digital twin.

```
//use the IoV system abstraction and Platooning mediator abstraction
  with SelfDrivingVehicle,Platooning
  architecture PlatooningSoS(max:Distance,min:Distance) is {
   coalition platoon is compose{
    fleet is sequence{SelfDrivingVehicle()}
    platooning is sequence{Platooning(max,min)}
   } binding {
      forall {vehicle in fleet suchthat
        exists{one steer in platooning suchthat
          unify one{steer::position} to one{vehicle::position}
          unify one{steer::command} to one{vehicle::command}}}}
  }
```

Listing 3. SoS architecture description for platooning in SosADL for IoT under Uncertainty

Each self-driving vehicle is not aware of the rest of the platoon, except for its own neighbors, generally composed of the platoonmates in front of and behind itself. For instance, the self-driving vehicle in the platoon is not aware of the size of the platoon, i.e. of how many members has a platoon.

Besides, it is worth pointing out that it is the locality of the rules that ensures the scalability of the emergent behavior of platoon in terms of digitally mediated twins.

5 Implementation of SosADL Studio for IoT with Uncertainty

The complexity inherent to SoS architectures on the IoT under uncertainty calls for the automated support of SoS architectural design by software tools. Indeed, for supporting the architecture-centric formal development of SoSs using *SosADL for IoT under Uncertainty*, we have developed an SoS Architecture Development Environment, named SosADL Studio [27].

This toolset is constructed as plugins in Eclipse. It provides a model-driven architecture development environment where the SosADL meta-model is defined in EMF/Ecore (http://eclipse.org/modeling/emf/), with the textual concrete syntax

expressed in Xtext (http://eclipse.org/Xtext/), the graphical concrete syntax developed in Sirius (http://eclipse.org/sirius/), and the type checker implemented in Xtend (http://www.eclipse.org/xtend/), after having being proved using the Coq proof assistant (http://coq.inria.fr/) [28].

The type-checked architecture is then converted in terms of input-output Symbolic Transition Systems to feed UPPAAL (http://www.uppaal.org/) for model checking.

At that point, the SoS architecture is guaranteed well-formed (through the type checker) and well-behaved (through the model checker).

To support uncertainty in SoS architecture descriptions by the resolution of concurrent constraints of the checked SoS architecture, a constraint solving mechanism was implemented on top of the Kodkod Solver (http://alloy.mit.edu/kodkod/). It is used to solve the system of architectural constraints formulated in the architectural model at run-time (based on *tell*, *ask*, *untell* constructs) and to "find" (i.e. dynamically create) mediated structures that satisfy these constraints. It operates thereby as an architectural model finder, enabling to discover SoS concrete architectures which comply with the given SoS abstract architecture and operational environment, based on Propositional Satisfiability Theory (SAT) [10]. The concretization of the SoS architecture is incremental and based on the discover of digital twins on the IoT. Each time a digital twin of a constituent system become known by the SoS, a better SAT solution may be computed, reconfiguring the digital coalition of the SoS. Kodkod Solver indeed support model finding in partially defined models, which can be incrementally instantiated.

The remaining issue is thereby to guarantee that the architected emergent behaviors are able to be produced and to guarantee that they will behave correctly under uncertainty, as required.

To provide this guarantee, SosADL Studio supports simulation of SoS architectures. It takes as input the SoS architecture description in terms of digital twins, its operational environment (via concrete specifications of environmental gates given how constituent systems sense local environments), and produces different simulation models in Discrete Event System Specification (DEVS, http://www.ms4systems.com/).

For supporting verification of the emergent behaviors of SoS architectures under uncertainty, we have conceived a novel logic, named DynBLTL [30], for expressing properties of SoS architectures under a multivalued logic and verifying these properties with statistical model checking using PLASMA (http://project.inria.fr/plasma-lab/) [4].

Through the different features of the SosADL Studio, an SoS architect is able to guarantee different SoS architectural properties on the IoT in the presence of uncertainty, in particular enforcing safety [28].

6 Related Work

Uncertainty is a topic of research in different disciplines, including social sciences, natural sciences, and engineering [33]. In engineering, uncertainty is intrinsically related to design, termed "design under uncertainty". The notion of uncertainty has not, however, the same meaning in these different disciplines, encompassing, as a matter of fact, a multiplicity of concepts.

Recently, uncertainty has been the subject of research in software-intensive systems engineering [6]. In particular, [35] proposes a taxonomy of uncertainty for dynamically self-adaptive software-intensive systems, focusing on cyber-physical systems and their operation in unpredictable physical environments.

Since the rise of IoT, the nature of SoS has evolved, and uncertainty has become a critical issue, increasingly been cited in SoS roadmaps for the coming years, e.g. the European Roadmap for SoS Engineering [5], which expresses the needs for constructing Cyber-Physical SoSs on the IoT. Notably, the INCOSE Systems Engineering Vision for 2025 [12] highlights the need to cope with uncertainty at analysis, architectural design, and operation of SoSs.

More recently, the OMG (an industrial standard organization) has highlighted the importance of dealing with uncertainty during the lifecycle of a software-intensive system or SoS, and launched an initiative for supporting uncertainty modeling in its model engineering standards [18].

Over the years, in Software Engineering, various techniques were studied for documenting, specifying, and managing uncertainty in software-intensive systems [6, 8].

More particularly in software-intensive systems or SoS, uncertainty has been predominantly studied in requirements engineering. In particular, the RELAX requirements specification language [34] was defined to express requirements under uncertainty based on fuzzy branching temporal logic. RELAX supports the explicit specification of environmental uncertainty in requirements statements. FLAGS [2] extended KAOS [3] using possibility theory to mitigate the uncertainty of the goals. RELAX and FLAGS do not address architectural design, but are able to express requirements under uncertainty that are relevant to be used for architectural design.

While uncertainty is inherent to SoS [15] and has been addressed in requirements engineering, the study of uncertainty in SoS architecture is still highly unexplored [14].

Indeed, if the importance of dealing with uncertainty when designing software-intensive systems and SoS architectures is widely recognized, only a dearth of techniques having been proposed for handling uncertainty in software architecture description [26]. Notably, none has proposed a formal Architecture Description Language (ADL) enabling to deal with uncertainty in SoS architecture description [14].

Actually, many researches were conducted to address the issue of formally describing the architecture of single software-intensive systems. However, no existing ADL is able to describe even single systems architecture under uncertainty, being limited by their formal foundations which require complete information at design-time.

The significant difference of the formal basis of ADLs for single systems and the π-Calculus for SoS underlying *SosADL for IoT under Uncertainty* is the treatment of partial information (i.e. each SoS constituent has incomplete information on the state of the operational environment). While in single systems, all components have access to complete information, in SoS, all constituents have access to imperfect information by nature. In SoSs, partial information contributes to systematic uncertainty.

In summary, based on the study of the state-of-the-art, *SosADL for IoT under Uncertainty* is positioned as a pioneering ADL, being the first formal ADL enabling the expression of systematic uncertainty in SoS architecture description relying on concurrent constraints and constraint solving mechanisms, being specialized for IoT relying on digital twins and digital mediators and coalitions.

7 Conclusion and Future Work

In this paper we have raised the issue of uncertainty in the architectural design of SoSs, and identified the major sources of uncertainty for architecting SoSs on the IoT. In particular, we faced up the research challenge of how to deal with uncertainty in the design of an SoS architecture on the IoT. To solve this issue, this paper proposed enhancements to SosADL. On the one hand, the underlying theory of SosADL was enhanced with concurrent constraints for expressing uncertainty due to imprecision (partial knowledge) and on the other hand, novel architectural abstractions were integrated into SosADL for treating the duality of "things" on the IoT, through the notion of digital twin, synchronously coupled with its physical counterpart, and their coordination using digital mediators, for forming digital coalitions to produce the required SoS emergent behaviors.

Using the enhanced SosADL presented in this paper, *SosADL on the IoT under Uncertainty*, SoS architectures on the IoT are abstractly described by declaring the constraints which must be satisfied when the concrete architecture is synthesized in a specific operational environment where "things" are IoT systems communicating via the IoT. The behavior of digital twins of constituent systems as well as the behavior of digital mediators are specified by handling constraints representing and reasoning about partial information.

SosADL and more recently this enhanced SosADL have been applied in several case studies and pilots where the suitability of SosADL and the supporting toolchain has been validated, including SoS architecture descriptions under uncertainty on the IoT.

On-going and future work is mainly related with the application of *SosADL for IoT under Uncertainty* in industrial-scale projects. They include in particular joint work with SEGULA for applying the extended SosADL to architect SoSs in the domain of the Industrial IoT. Description of SoS architectures on the IoT under uncertainty, and their validation and verification using the SosADL toolchain, are main threads of these pilot projects.

References

1. Ayyub, B., Klir, G.: Uncertainty Modeling and Analysis in Engineering and the Sciences. Chapman & Hall, Boca Raton (2006)
2. Baresi, L., Pasquale, L., Spoletini, P.: Fuzzy goals for requirements-driven adaptation. In: 18th IEEE RE, Sydney, Australia, September 2010
3. Cailliau, A., van Lamsweerde, A.: Handling knowledge uncertainty in risk-based requirements engineering. In: 23rd RE, Ottawa, Canada (2015)
4. Cavalcante, E., Quilbeuf, J., Traonouez, L.-M., Oquendo, F., Batista, T., Legay, A.: Statistical model checking of dynamic software architectures. In: Tekinerdogan, B., Zdun, U., Babar, A. (eds.) ECSA 2016. LNCS, vol. 9839, pp. 185–200. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48992-6_14
5. CPSoS: European Research and Innovation Agenda on Cyber-Physical Systems-of-Systems 2016–2025 (2016). http://www.cpsos.eu/roadmap/

6. Esfahani, N., Malek, S.: Uncertainty in self-adaptive software systems. In: de Lemos, R., Giese, H., Müller, H.A., Shaw, M. (eds.) Software Engineering for Self-Adaptive Systems II. LNCS, vol. 7475, pp. 214–238. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-35813-5_9
7. Tao, F., Zhang, M., Nee, A.Y.C.: Digital Twin Driven Smart Manufacturing. Academic Press, Cambridge (2019)
8. Garlan, D.: Software engineering in an uncertain world. In: ACM Future of Software Engineering Research, Santa Fe, NM, USA, November 2010
9. Grieves, M.: Virtually Perfect: Driving Innovative and Lean Products through Product Lifecycle Management. Space Coast Press, Cocoa Beach (2011)
10. Guessi, M., Oquendo, F., Nakagawa, E.Y.: Checking the architectural feasibility of systems-of-systems using formal descriptions. In: 11th IEEE SoSE, Kongsberg, Norway (2016)
11. Hubbard, D.W.: How to Measure Anything, 3rd edn. Wiley, Hoboken (2014)
12. INCOSE: SE Vision 2025 (2014). www.incose.org/AboutSE/sevision
13. Jia, D., Lu, K., Wang, J., Zhang, X., Shen, X.: A survey on platoon-based vehicular cyber-physical systems. IEEE Commun. Surv. Tutor. **18**(1), 263–284 (2016)
14. Klein, J., van Vliet, H.: A systematic review of system-of-systems architecture research. In: 9th ACM QoSA, Vancouver, Canada, June 2013
15. Maier, M.W.: Architecting principles for systems-of-systems. Syst. Eng. J. **1**(4), 267–284 (1998)
16. Milner, R.: Communicating and Mobile Systems: The π-Calculus. Cambridge Press, Cambridge (1999)
17. Olarte, C., Rueda, C., Valencia, F.D.: Models and emerging trends of concurrent constraint programming. Int. J. Constraints **18**, 535–578 (2013)
18. OMG: Precise Semantics for Uncertainty Modeling, Request For Proposal, OMG Document ad/2017-12-01, December 2017
19. Oquendo, F.: Formally describing the software architecture of systems-of-systems with SosADL. In: 11th IEEE SoSE, Kongsberg, Norway, June 2016
20. Oquendo, F.: The π-calculus for SoS: novel π-calculus for the formal modeling of software-intensive systems-of-systems. In: Communicating Process Architectures (CPA), vol. 69 (2016)
21. Oquendo, F.: Case study on formally describing the architecture of a software-intensive system-of-systems with SosADL. In: 15th IEEE SMC, Budapest, Hungary, October 2016
22. Oquendo, F.: Formally describing the architectural behavior of software-intensive systems-of-systems with SosADL. In: 21st IEEE ICECCS, Dubai, UAE, November 2016
23. Oquendo, F.: Software architecture challenges and emerging research in software-intensive systems-of-systems. In: Tekinerdogan, B., Zdun, U., Babar, A. (eds.) ECSA 2016. LNCS, vol. 9839, pp. 3–21. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48992-6_1
24. Oquendo, F.: Architecturally describing the emergent behavior of software-intensive system-of-systems with SosADL. In: 12th IEEE SoSE, Waikoloa, Hawaii, USA, June 2017
25. Oquendo, F.: Formally describing self-organizing architectures for systems-of-systems on the internet-of-things. In: Cuesta, C.E., Garlan, D., Pérez, J. (eds.) ECSA 2018. LNCS, vol. 11048, pp. 20–36. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00761-4_2
26. Oquendo, F.: Coping with uncertainty in systems-of-systems architecture modeling. In: 14th IEEE SoSE, Anchorage, Alaska, USA, May 2019
27. Oquendo, F., Buisson, J., Leroux, E., Moguérou, G., Quilbeuf, J.: The SosADL architect studio. In: SiSoS 2016, Copenhagen, DK. ACM, November 2016
28. Oquendo, F., Buisson, J., Leroux, E., Moguérou, G.: A formal approach for architecting software-intensive systems-of-systems with guarantees. In: 13th IEEE SoSE, Paris, France, June 2018

29. Oxford Dict. https://en.oxforddictionaries.com/definition/uncertainty
30. Quilbeuf, J., Cavalcante, E., Traonouez, L.-M., Oquendo, F., Batista, T., Legay, A.: A logic for the statistical model checking of dynamic software architectures. In: Margaria, T., Steffen, B. (eds.) ISoLA 2016. LNCS, vol. 9952, pp. 806–820. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-47166-2_56
31. Reynolds, C.W.: Flocks, herds, and schools: a distributed behavioral model, in computer graphics. In: 14th SIGGRAPH, Anaheim, USA (1987)
32. Roca, D., Nemirovsky, D., Nemirovsky, M., Milito, R., Valero, M.: Emergent behaviors in the internet-of-things: the ultimate ultra-large-scale system. In: IEEE Micro, vol. 36, no. 6, November–December 2016
33. Thunnissen, D.P.: Uncertainty classification for the design and development of complex systems. In: 3rd Predictive Methods Conference (PMC), Newport Beach, CA, USA, June 2003
34. Whittle, J., Sawyer, P., Bencomo, N., Cheng, B.H.C., Bruel, J.-M.: RELAX: a language to address uncertainty in self-adaptive systems requirement. Requir. Eng. J. **15**(2), 177–196 (2010)
35. Zhang, M., Selic, B., Ali, S., Yue, T., Okariz, O., Norgren, R.: Understanding uncertainty in cyber-physical systems: a conceptual model. In: Wąsowski, A., Lönn, H. (eds.) ECMFA 2016. LNCS, vol. 9764, pp. 247–264. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42061-5_16