SoSECIE Webinar

Welcome to the
2021 System of Systems Engineering Collaborators
Information Exchange (SoSECIE)

NDIA
National Defense Industrial Association

MITRE

We will start at 11AM Eastern Time

You can download today's presentation from the SoSECIE Website:

https://mitre.tahoe.appsembler.com/blog

To add/remove yourself from the email list or suggest a future topic or speaker, send an email to sosecie@mitre.org

NDIA System of Systems SE Committee

- **Mission**
  - To provide a forum where government, industry, and academia can share lessons learned, promote best practices, address issues, and advocate systems engineering for Systems of Systems (SoS)
  - To identify successful strategies for applying systems engineering principles to systems engineering of SoS
- **Operating Practices**
  - Face to face and virtual SoS Committee meetings are held in conjunction with NDIA SE Division meetings that occur in February, April, June, and August

NDIA SE Division SoS Committee Industry Chairs:
Mr. Rick Poel, Boeing
Ms. Jennie Horne, Raytheon
OSD Liaison:
Dr. Judith Dahmann, MITRE

Simple Rules of Engagement

- I have muted all participant lines for this introduction and the briefing.
- If you need to contact me during the briefing, send me an e-mail at sosecie@mitre.org.
- Download the presentation so you can follow along on your own
- We will hold all questions until the end:
  - I will start with questions submitted online via the CHAT window in Teams.
  - I will then take questions via telephone; State your name, organization, and question clearly.
- If a question requires more discussion, the speaker(s) contact info is in the brief.

Disclaimer

- MITRE and the NDIA makes no claims, promises or guarantees about the accuracy, completeness or adequacy of the contents of this presentation and expressly disclaims liability for errors and omissions in its contents.
- No warranty of any kind, implied, expressed or statutory, including but not limited to the warranties of non-infringement of third party rights, title, merchantability, fitness for a particular purpose and freedom from computer virus, is given with respect to the contents of this presentation or its hyperlinks to other Internet resources.
- Reference in any presentation to any specific commercial products, processes, or services, or the use of any trade, firm or corporation name is for the information and convenience of the participants and subscribers, and does not constitute endorsement, recommendation, or favoring of any individual company, agency, or organizational entity.

2021-2022 System of Systems Engineering Collaborators Information Exchange Webinars

Sponsored by MITRE and NDIA SE Division

April 6, 2021
Holistic architecture description for a future Global Health Assurance Systems of Systems
Adrián Unger

April 20, 2021
Leveraging Set-Based Practices to Enable Efficient Concurrency in Large Systems and Systems-of-Systems Engineering
Brian Kennedy

May 4, 2021
OUSD R&E: USD(R&E) Mission Engineering (ME) State of Practice
Elmer L. Roman

May 18, 2021
Application of Probabilistic Graph Models to Kill Chain and Multi-Domain Kill Web Analysis Problems
Jason Baker and Valerie Sitterle

https://www.mitre.org/capabilities/systems-engineering/collaborations/system-of-systems-engineering-collaborators

2021-2022 System of Systems Engineering Collaborators Information Exchange Webinars

Sponsored by MITRE and NDIA SE Division

June 1, 2021
Applying an MBSE Approach for Evaluating Shipyard Operations
David Jurkiewicz

June 15, 2021
Implementing a Digital Engineering Environment for Mission Engineering
Jason Anderson and Jeffrey Boulware

June 29, 2021
Digital Engineering: From Toolchain to Platform
Dr. Aleksandra Markina-Khusid

July 13, 2021
Developing Meta Systems Architectures for Leading Innovation with Complex Societal and Technical Challenges
Dr. Cihan Dagli

https://www.mitre.org/capabilities/systems-engineering/collaborations/system-of-systems-engineering-collaborators

Fuzzy Architecture Description for Handling Uncertainty in Systems-of-Systems in the Internet-of-Things

Flavio Oquendo, Ph.D., H.D.R.
Full Professor of Computing and Software-intensive Systems Engineering &
Research Director at UMR CNRS IRISA, France
flavio.oquendo@irisa.fr
http://people.irisa.fr/Flavio.Oquendo/

Systems-of-Systems Engineering Collaborators Information Exchange
NDIA & MITRE SoSECIE Webinar - March 23, 2021

Abstract of the Webinar

- **The Research Issue:** Uncertainty is intrinsically associated with the architectural design of Software-intensive Systems-of-Systems (SoS) by its very nature, e.g. a platooning of self-driving vehicles in Intelligent Transportation Systems (ITS)

- **The Research Question:** The consequent research question is thereby how to represent uncertainty in the description of an SoS architecture and subsequently use that representation to reason about SoS architectural properties

- **The Validated Solution:** To address this research question in the Internet-of-Things (IoT), the work presented in this webinar has investigated the notion of epistemic uncertainty (i.e. uncertainty due to partial knowledge) in the architectural design of SoSs in the IoT and extended an SoS Architecture Description Language with fuzzy concepts and constructs underlain by Fuzzy Theory, defining the Fuzzy SosADL
  - The effectiveness of Fuzzy SosADL on handling epistemic uncertainty has been demonstrated through experiments in SoS platooning architectures of self-driving vehicles

Outline

1. Introduction: The Problematics of Architecting SoS under Uncertainty in the Internet-of-Things (IoT)
2. Motivating Case Study: The Case of Vehicle Platooning as SoS in the Internet-of-Vehicles
3. Approach for Handling Uncertainty in SoS Architectural Design: Fuzzy Theory for Handling Epistemic Uncertainty
4. Background on Fuzzy Theory: Very Brief Overview of the Fundamental Concepts
5. Conceiving a Fuzzy SoS Architecture Description Language (ADL): Enhancing SosADL with Fuzzy Constructs for Describing Fuzzy SoS Architectures under Epistemic Uncertainty
6. Applying Fuzzy SosADL in the Motivating Case Study: Describing the SoS Architecture for Platooning of Self-Driving Vehicles with Fuzzy SosADL
7. Supporting Software Toolset: The Fuzzy SosADL Studio
8. Validation by Controlled Experiments: Comparing the Effectiveness of Described SoS Architectures under Uncertainty – "crisp" SosADL vs. Fuzzy SosADL
9. Summing Up

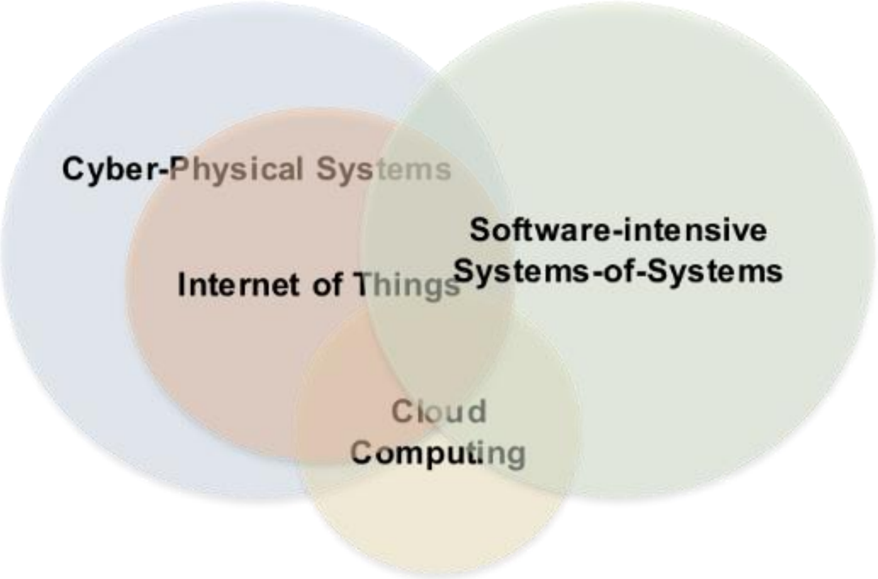
The Talk

1. Introduction

The Problematics of Architecting SoS under Uncertainty in the Internet-of-Things (IoT)

1.1 Introduction: Evolution from Software-intensive Single Systems (SiS) to Systems-of-Systems (SoS)

- Software-intensive systems have rapidly evolved
  - they were stand-alone systems in the past
  - they are often part of networked systems in the present
  - they are increasingly becoming systems of systems in the upcoming future
    - System-of-Systems (SoS)
      - e.g. Vehicle Platooning
- Key enabling platform for upcoming SoS
  - Internet-of-Things (IoT)
    - Internet-of-Vehicles (IoV)

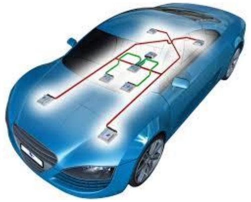


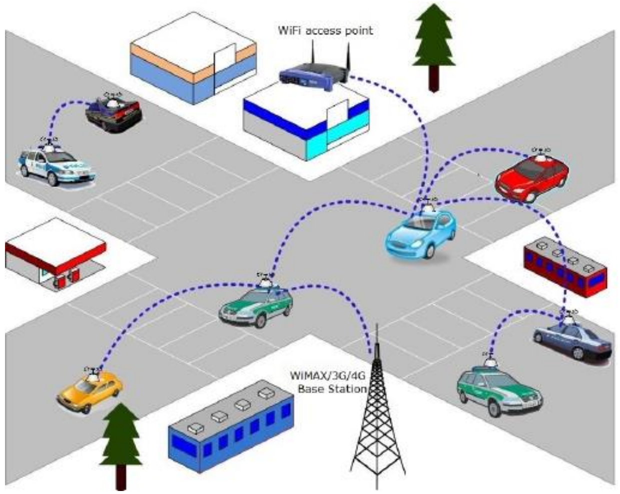
Cyber-Physical Systems

Internet of Things

Software-intensive Systems-of-Systems

Cloud Computing





1.2 Introduction: Kinds of Uncertainty in SoS

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

1.3 Problematics: How to Describe SoS Architectures in the IoT under Uncertainty?

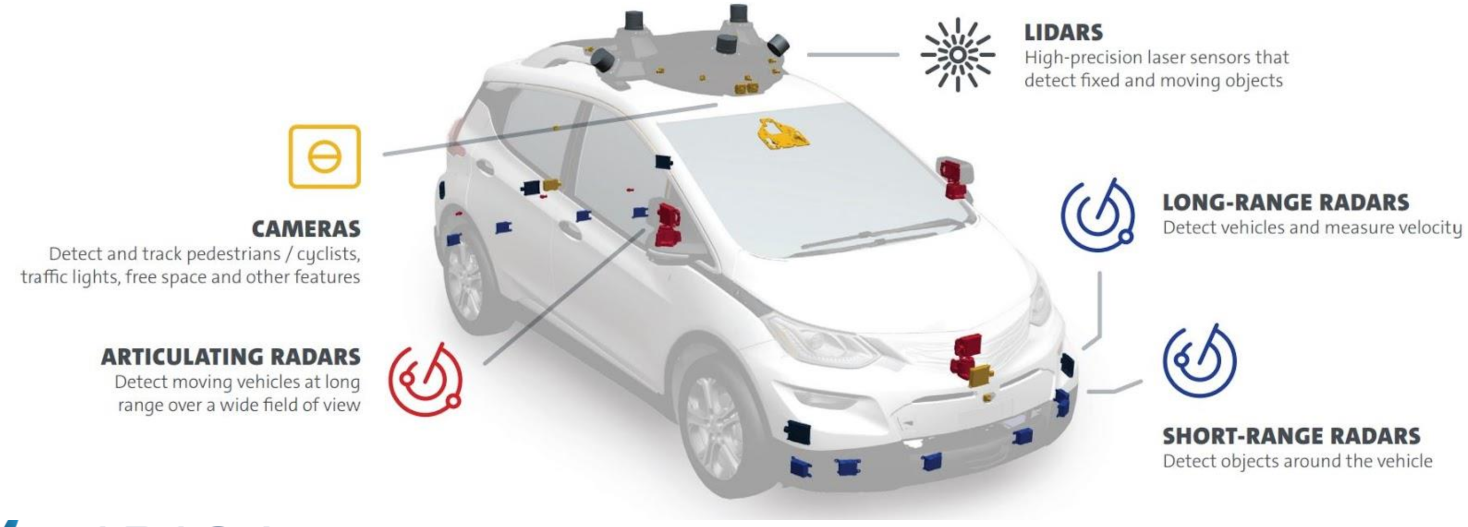
- SoS architects conceive SoS architectures at design-time specifying how SoS constituents will enable the creation of emergent behaviors at run-time
- Each system participating in an SoS in the IoT has only partial information about its local physical and virtual environments (possibly supported via ad-hoc networks)
  - Its local environment is perceived via its sensors and actuated upon via its actuators
  - Possibly including other systems in shared physical and virtual environments
- SoSs need to be designed and operated in the presence of epistemic uncertainty:
  - Measures from sensors are subject to epistemic uncertainty
  - Effects from actuators are subject to epistemic uncertainty
- The challenge in the description of an SoS architecture in the IoT is therefore to be able to handle the uncertainty raised by the partial information of constituent systems which will concretely operate in the SoS at run-time as well as by the partially known characteristics of the concrete operational environment where the SoS will actually operate

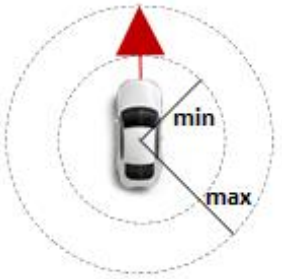
2. Motivating Case Study on the SoS Architecture Description under Uncertainty in the IoT

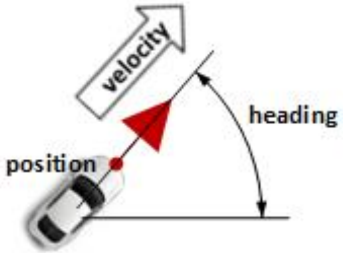
The Case of Vehicle Platooning as SoS in the Internet-of-Vehicles

2.1 Architecting Platoons under Uncertainty in IoV: Self-Driving Vehicles as Platoonmates

- Self-driving vehicles are equipped with radars/lidars (light detection and ranging devices) and other sensors e.g. cameras, steering actuators, and Vehicular Ad-hoc Network (VANET)
- Using these devices, a self-driving vehicle can sense information from its operational environment, including other vehicles in its perception range, process this information and possibly communicate the processed information to other nearby vehicles through VANET, as well as command its steering actuators

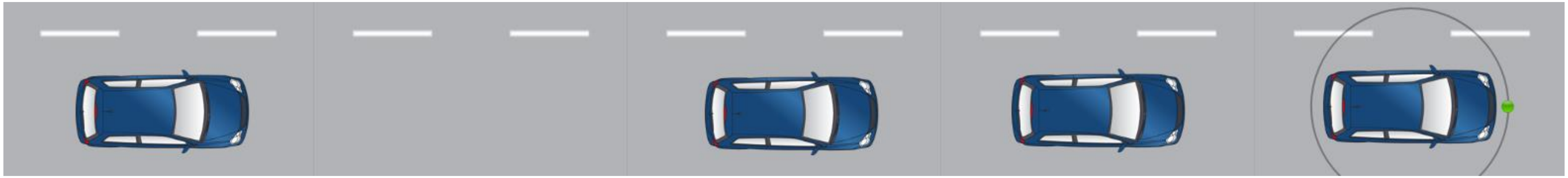

LIDARS
High-precision laser sensors that detect fixed and moving objects
CAMERAS
Detect and track pedestrians / cyclists, traffic lights, free space and other features
LONG-RANGE RADARS
Detect vehicles and measure velocity
ARTICULATING RADARS
Detect moving vehicles at long range over a wide field of view
SHORT-RANGE RADARS
Detect objects around the vehicle

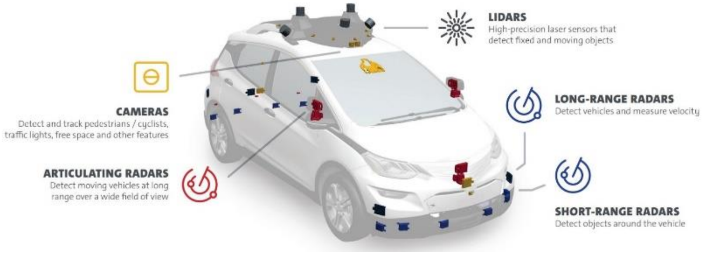


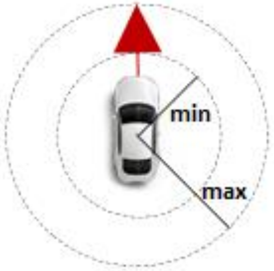


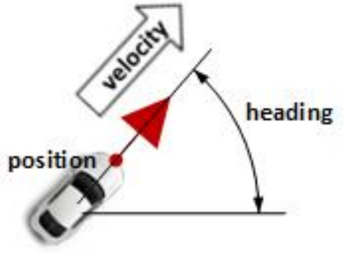
2.2 Architecting Platoons under Uncertainty in IoV: Platooning of Self-Driving Vehicles (a)

- **Vehicle Platooning:** platooning is the process of vehicles (in this case self-driving vehicles) autonomously forming road convoys (each vehicle in the platoon follows the platoonmate in front of it, except the leader that drives to the destination)
  - It requires that each vehicle in the platoon control its velocity (speed and heading) and the relative distance to the vehicle in front of it



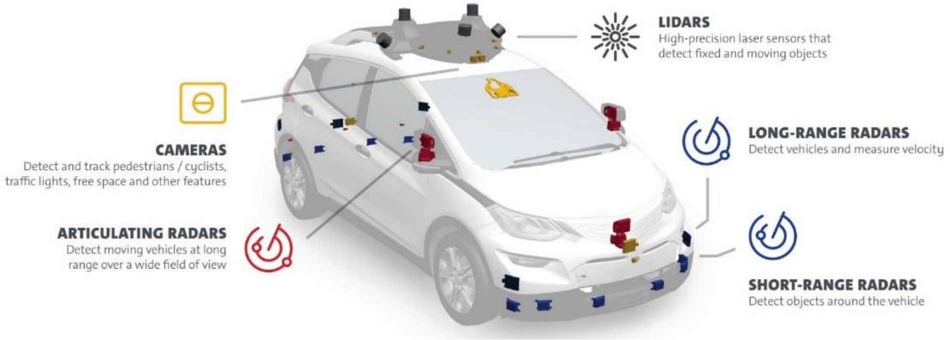

LIDARS
High-precision laser sensors that detect fixed and moving objects
CAMERAS
Detect and track pedestrians / cyclists, traffic lights, free space and other features
LONG-RANGE RADARS
Detect vehicles and measure velocity
ARTICULATING RADARS
Detect moving vehicles at long range over a wide field of view
SHORT-RANGE RADARS
Detect objects around the vehicle

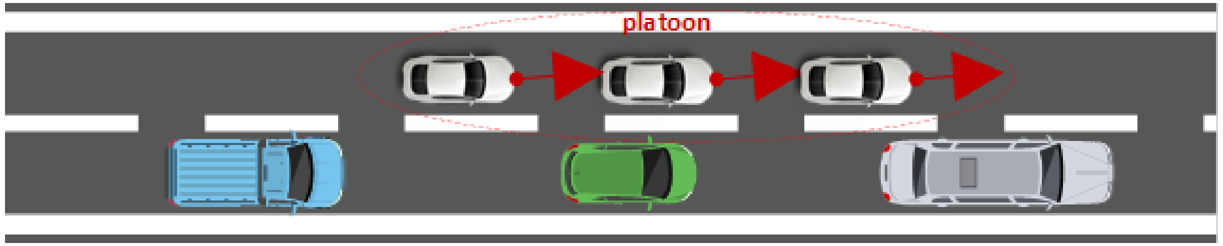




2.2 Architecting Platoons under Uncertainty in IoV: Platooning of Self-Driving Vehicles (b)

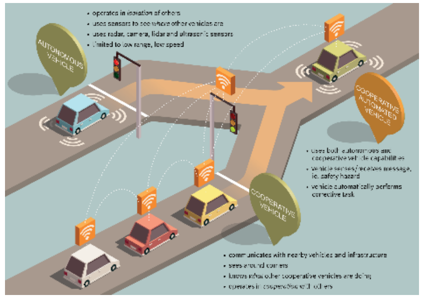
- Architects describe SoS architectures at design-time specifying how SoS constituents will architecturally enable to create and maintain emergent behaviors at run-time
  - In IoV (the “IoT of Vehicles”), the challenge is to coordinate different self-driving vehicles for performing together platooning through emergent behavior
- Challenge to describe Platooning as an SoS architecture in the IoV
  - The challenge in the architectural design of Platooning SoSs in the IoV is to describe how an SoS architecture is able to create, on the fly, and maintain platooning emergent behaviors from self-driving vehicles, where the actual vehicles and the operational environment are not known at design time and that the perception/sensing of the environment as well the actuation on the environment are subject to uncertainty


LIDARS
High-precision laser sensors that detect fixed and moving objects
CAMERAS
Detect and track pedestrians / cyclists, traffic lights, free space and other features
LONG-RANGE RADARS
Detect vehicles and measure velocity
ARTICULATING RADARS
Detect moving vehicles at long range over a wide field of view
SHORT-RANGE RADARS
Detect objects around the vehicle



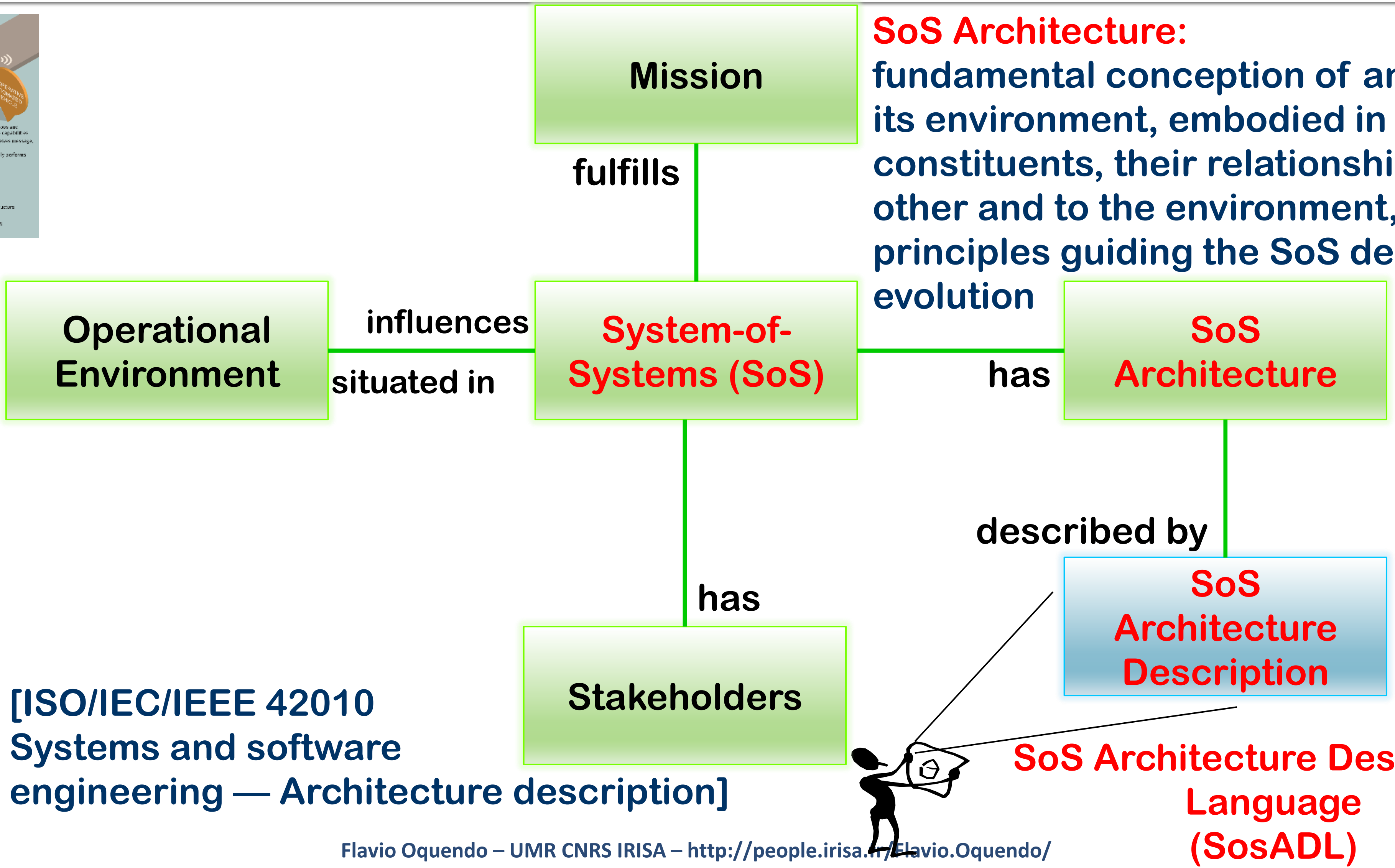
Fuzzy Theory for Handling Epistemic Uncertainty in SoS Architectural Design with an ADL (Architecture Description Language)

3.1 SosADL: Architecture Description Language (ADL) for Software-intensive Systems-of-Systems (SoS)



- operates in isolation of others
- uses sensors to see where other vehicles are
- uses radar, camera, lidar and ultrasonic sensors
- limited to low range, low speed
AUTONOMOUS VEHICLE
COOPERATIVE AUTOMATED VEHICLE
- uses both autonomous and cooperative vehicle capabilities
- vehicle senses/receives message, ie. safety hazard
- vehicle automatically performs corrective task
COOPERATIVE VEHICLE
- communicates with nearby vehicles and infrastructure
- sees around corners
- knows what other cooperative vehicles are doing
- operates in cooperation with others

SoS Architecture:
fundamental conception of an SoS in its environment, embodied in its constituents, their relationships to each other and to the environment, and principles guiding the SoS design and evolution



SoS Architecture Description Language (SosADL)

[ISO/IEC/IEEE 42010 Systems and software engineering — Architecture description]

3.2 SosADL: An ADL for SoS based on the concepts of Constituent Systems, Mediators, and Coalition

- SosADL: novel ADL for SoS
- SoS architecture description from different viewpoints:
  - Structure
  - Behavior, including Emergent Behavior
  - Analysis

Oquendo, F.: "Formally Describing the Software Architecture of Systems-of-Systems with SosADL", Proc. of the 11th IEEE System-of-Systems Engineering Conference (SoSE), Kongsberg, Norway, June 2016

Oquendo, F.: "Architecturally Describing the Emergent Behavior of Software-intensive System-of-Systems with SosADL", Proc. of the 12th IEEE System-of-Systems Engineering Conference (SoSE), Waikoloa, Hawaii, USA, June 2017

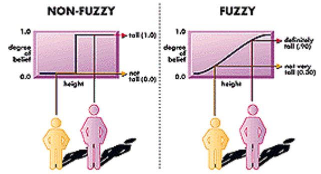
- SoS architecture description in terms of:
  - Constituent systems
    - locus of operation capabilities for enabling functionalities
  - Mediators
    - locus of mediation capabilities for enabling emergent behavior
  - Coalition
    - composition of constituent systems coordinated by mediators for fulling specified missions
      - resulting in overall emergent behaviors

3.3 Which Formal Theory for Handling Uncertainty?

- To enhance SosADL for handling Epistemic Uncertainty in the design of SoS architectures in the IoT, the question to investigate is:

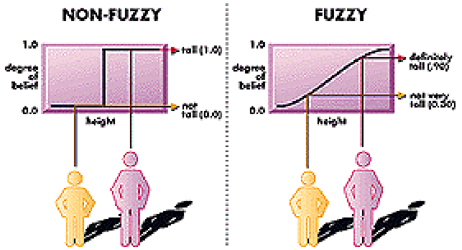
Which theory is the most suitable for addressing the needs of SoS architectural design to handle Epistemic Uncertainty?

  - Possibility theory is the chief uncertainty theory devoted to the handling of partial information
  - Possibility is associated with fuzziness, either in the background knowledge or in the dataset for which possibility is asserted
- We adopted the Fuzzy Theory to extend SosADL as it satisfies the requirements for representing partial knowledge
  - Fuzzy Theory provides description concepts and reasoning mechanisms for addressing these uncertainties
  - E.g., in platooning, a self-driving vehicle may measure the distance from it to the self-driving vehicle it is following using a radar/lidar (uncertainty in sensors), and when it accelerates or decelerates, by using the throttle or brakes, the variation of the actual speed depends on the weather as well as on the conditions of the road surface (uncertainty in actuators)



3.4 Fuzzy Theory for Handling Uncertainty in SoS

- Fuzzy Theory is the suitable mathematical theory that:
  - On the one hand is able to represent partial information on the SoS operational environment
  - On the other hand provides several calculi for the fusion of partial information from different sources, e.g. different sensors of a self-driving vehicle, as well as for computing relevant steering commands to be executed by vehicle actuators
- Fuzzy Theory is composed of a collection of different related theories:
  - The seminal one is Fuzzy Sets, the basis for Fuzzy Logics, which underlines different approaches for Fuzzy Control Systems, supported by Fuzzy Inference Systems



4. Background on Fuzzy Theory: Fundamental Concepts

Very Brief Overview of the Fundamental Concepts of Fuzzy Theory

4.1 Fundamental Concepts of Fuzzy Theory

- Fuzzy Theory:
  - Fuzzy Sets (and Fuzzy Numbers)
  - Fuzzy Logics
  - Fuzzy Inference Systems

- Fuzzy Theory has been developed and applied since the sixties:
  - In 1965, Lotfi A. Zadeh published the Theory of Fuzzy Sets
  - Zadeh L. A.: “Fuzzy Sets”, Information and Control, Vol. 8, 1965, pp. 338-353 (seminal paper with over 48,000 citations)

- The Theory of Fuzzy Sets has given rise to over 50,000 patents just in Japan and the United States
  - Most of these applications apply Fuzzy Theory for addressing Uncertainty
  - Highly applied for Fuzzy Control Systems (control systems based on Fuzzy Logic)

4.2 Fuzzy Sets: Definition and Membership Function (a)

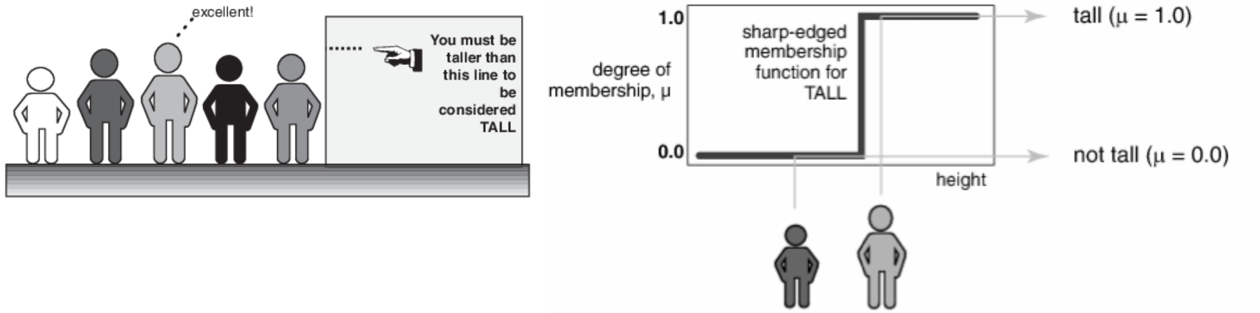
- **Classical sets (a.k.a. crisp sets):** Let the universe U be a nonempty set, a set S in U is characterized by its membership function $\chi(\mathrm{x})$, where $\mathrm{x} \in \mathrm{U}$

$$\chi : \mathrm{U} \to \{0, 1\} \text{ where } \mathrm{S} \subseteq \mathrm{U}$$

$$\chi(\mathrm{x}) = \begin{cases} 1 & \text{if } \mathrm{x} \in \mathrm{S} \subseteq \mathrm{U} \\ 0 & \text{if } \mathrm{x} \notin \mathrm{S} \subseteq \mathrm{U} \end{cases}$$

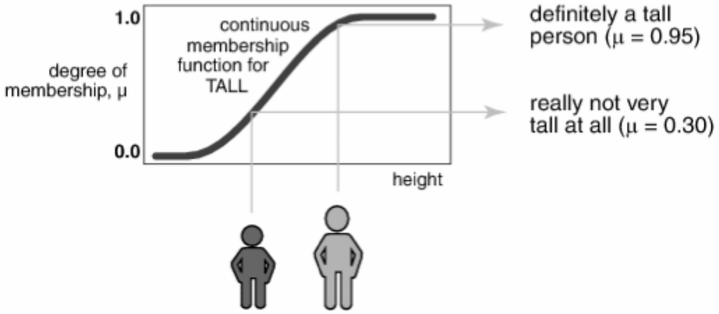
- Fuzzy sets can be seen as a generalization of classical sets by generalizing the concept of membership function allowing membership values between 0 and 1, including 0 and 1 themselves

- **Fuzzy sets:** Let the universe U be a nonempty set, a fuzzy set F in U is characterized by its membership function $\mu\mathrm{F}(\mathrm{x})$, which is interpreted as the degree of membership of element x in fuzzy set F for each $\mathrm{x} \in \mathrm{U}$


excellent!
You must be taller than this line to be considered TALL
1.0
0.0
degree of membership, μ
sharp-edged membership function for TALL
height
tall (μ = 1.0)
not tall (μ = 0.0)

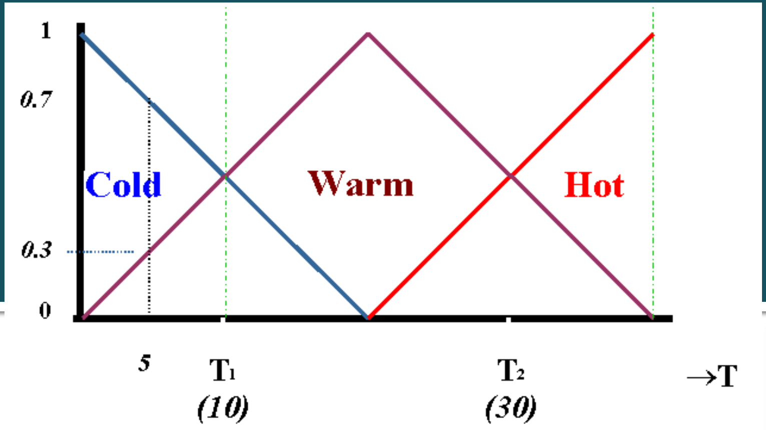
$$\mu\mathrm{F} : \mathrm{U} \to [0, 1]$$

$$\text{where } \mathrm{F} \subseteq \mathrm{U}$$



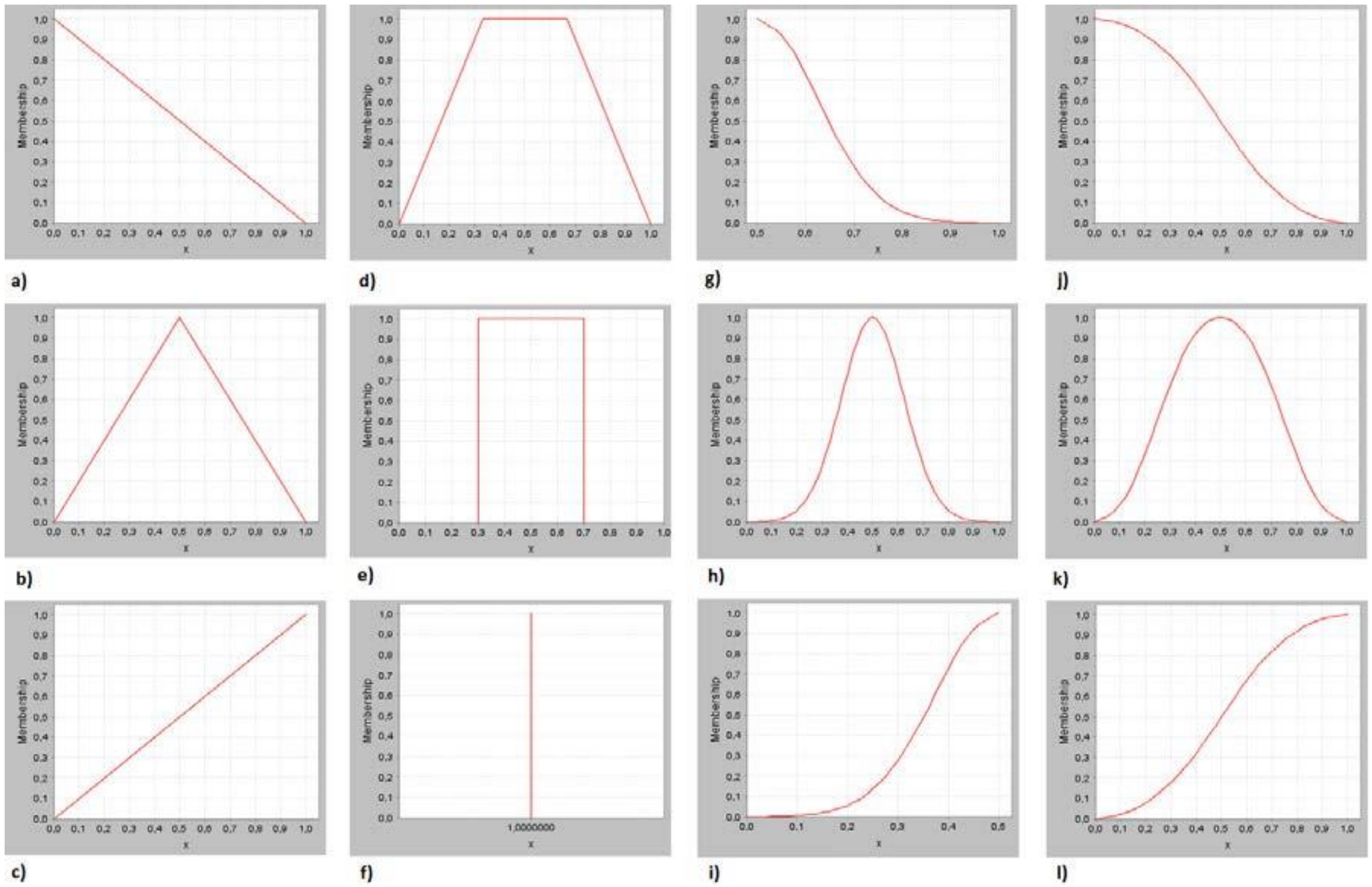
4. Background on Fuzzy Theory: Fundamental Concepts

4.2 Fuzzy Sets: Common Forms of Membership Functions (b)



- **Common forms of fuzzy set membership functions:**

a) left linear shape
b) triangular shape
c) right linear shape
d) trapezoidal shape
e) rectangular shape
f) singleton shape
g) left Gaussian shape
h) Gaussian shape
i) right Gaussian shape
j) z-shape
k) pi-shape
l) s-shape



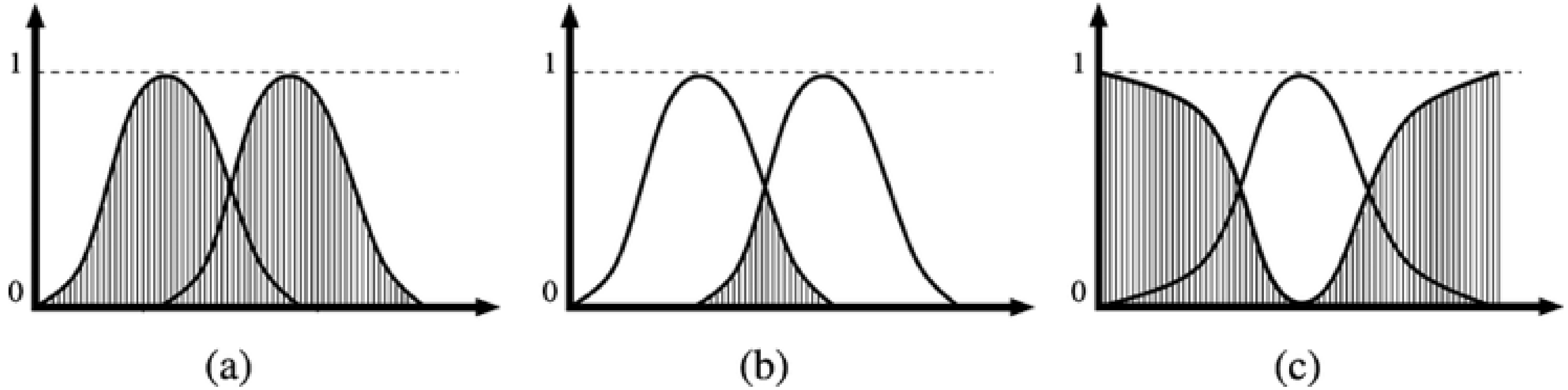
4.2 Fuzzy Sets: Fuzzy Operations with Membership Functions (c)

- **Standard fuzzy operations**

a) Maximum fuzzy union

b) Minimum fuzzy intersection

c) Fuzzy complement



4.3 Fuzzy Logics: Definition and Role of Fuzzy Implication (a)

- Classical Logics:
  - Based on Classical (Crisp) Sets

Classical logics are based on classical (crisp) sets and material implication:

- Let p and q be crisp propositions, $\mathrm{p} = (\mathrm{x}\ \mathit{is}\ \mathrm{A})$ and $\mathrm{q} = (\mathrm{y}\ \mathit{is}\ \mathrm{B})$, where A and B are crisp sets
- The full interpretation of the material implication $\mathrm{p} \Rightarrow \mathrm{q}$ is that the degree of truth of $\mathrm{p} \Rightarrow \mathrm{q}$ quantifies to what extent q is at least as true as p:

$$\mathrm{truth}(\mathrm{p} \Rightarrow \mathrm{q}) = \begin{cases} 1 & \text{if } \mathrm{truth}(\mathrm{p}) \leq \mathrm{truth}(\mathrm{q}) \\ 0 & \text{otherwise} \end{cases}$$

- Fuzzy Logics:
  - Based on Fuzzy Sets

Fuzzy logics are based on fuzzy sets and fuzzy implication:

- Let v and w be fuzzy propositions, $\mathrm{v} = (\mathrm{x}\ \mathit{is}\ \mathrm{P})$ and $\mathrm{w} = (\mathrm{y}\ \mathit{is}\ \mathrm{Q})$, where P and Q are fuzzy sets
- The full interpretation of the fuzzy implication $\mathrm{v} \Rightarrow \mathrm{w}$ is defined as the degree of truth of $\mathrm{v} \Rightarrow \mathrm{w}$ quantifying to what extent w is at least as true as v:

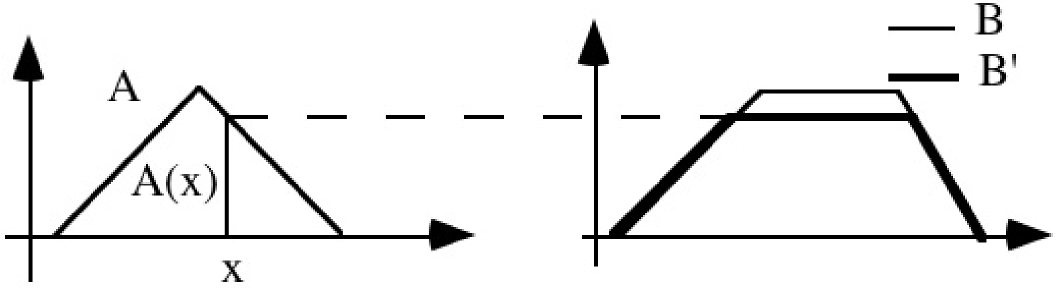
$$\mathrm{truth}(\mathrm{P(x)} \Rightarrow \mathrm{Q(y)}) = \begin{cases} 1 & \text{if } \mathrm{truth}(\mathrm{P(x)}) \leq \mathrm{truth}(\mathrm{Q(y)}) \\ 0 & \text{otherwise} \end{cases}$$

4.3 Fuzzy Logics: Inference Systems and Forms of Fuzzy Implication (b)

- Fuzzy reasoning in Fuzzy Logic
  - Fuzzy Logic provides a formal framework for reasoning in the face of uncertainty
  - Central to fuzzy reasoning is the representation of propositions as statements assigning fuzzy values to variables as well as fuzzy inference rules for fuzzy implication
- Often, the fuzzy implication is computed using the Mamdani's minimum operator to model causal relationship between fuzzy variables: $\mathrm{P(x)}\Rightarrow\mathrm{Q(y)} = \min\{\mathrm{P(x)},\mathrm{Q(y)}\}$
  - Other used fuzzy implication operators are: Early Zadeh, Łukasiewicz, Larsen, Standard Strict, Gödel, Gaines, Kleene-Dienes, Kleene-Dienes-Łukasiewicz, and Yager
- In Fuzzy Logic, the most applied fuzzy inference rule is the Generalized Modus Ponens: $\mathrm{P(x)}\Rightarrow\mathrm{Q(y)},\ \mathrm{P(x)} \vdash \mathrm{Q(y)}$
- A fuzzy inference system is a mechanism for mapping an input space to an output space using a Fuzzy Logic
  - each implication (i.e. $(\mathrm{x\ is\ A}) \Rightarrow (\mathrm{y\ is\ B})$) fires if its antecedent (i.e. $\mathrm{x\ is\ A}$) is different of 0 (i.e. it is not false)

4.3 Fuzzy Logics: Inference Systems and Forms of Fuzzy Implication (c)

- Inference using fuzzy implication:
  - Considering the input submitted to a fuzzy inference system, the degree of match of the input to an implication rule (which is the firing strength) is the membership degree of the input in the fuzzy set characterizing the antecedent part of the rule
  - The overall output is the weighted average of the individual rule outputs (given by the consequents), where the weight of a rule is its firing strength with respect to the input
- According to the applied fuzzy implication operators, there are different reasoning methods: Mamdani as well as Tsukamoto, Sugeno and Takagi, Larsen



Mamdani's fuzzy implication method

The Generalized Modus Ponens with Mamdani implication operator is defined as:

implication	x is A	$\Rightarrow$	y is B
premise	x is A'		
conclusion			y is B'

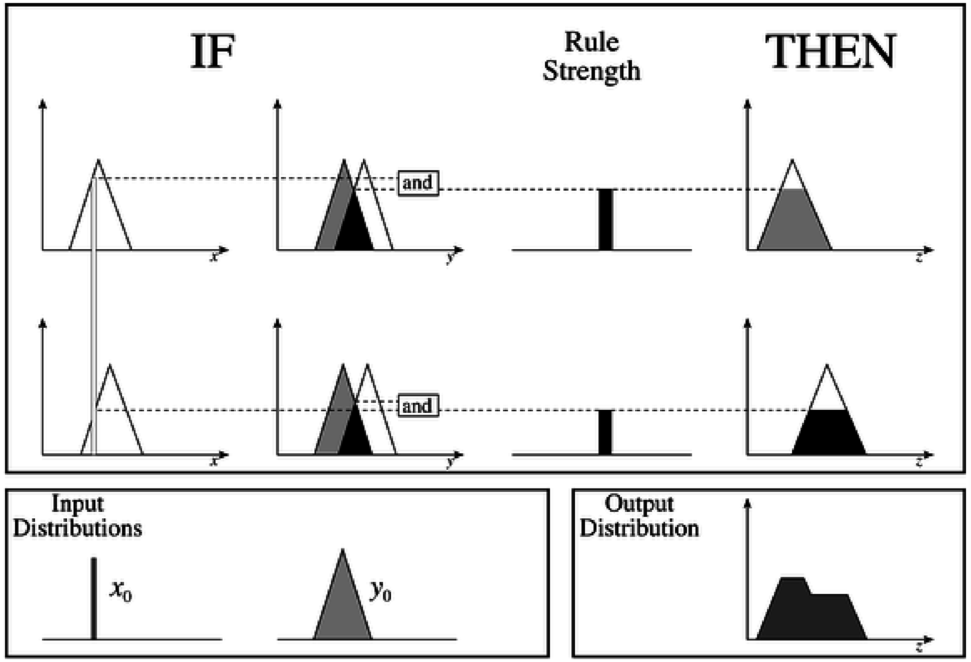
where: $B'(y) = \sup\{A'(x) \wedge A(x) \wedge B(y) \mid x \in \mathbb{R}\}, y \in \mathbb{R}$

- Note the causal relationship between x and y is known: y is a function of x, $y = f(x)$
  - Thereby, for a given x, $x = z$, we can compute that $y = f(z)$, enabling to compute $B'(y)$

4.4 Fuzzy Inference Systems as the Basis for Fuzzy SosADL

- Based on Fuzzy Theory, and in particular the notions of fuzzy sets, fuzzy implications, fuzzy logics, and fuzzy inference operators, fuzzy inference systems can be defined for reasoning on epistemic uncertainty in information as perceived by software-intensive systems
  - Based on the choices for membership functions of fuzzy sets, computation methods of fuzzy implications, and fuzzy inference operators, different fuzzy inference systems can be defined for fitting particular needs of different application domains
- The Fuzzy SoS Architecture Description Language (Fuzzy SosADL) is defined for supporting
  - The definition of different fuzzy inference systems
    - Fuzzy inference systems provide the suitable foundation for coping with epistemic uncertainty: expressing uncertainty as well as reasoning about uncertainties

  where an SoS Architecture Description defines a tailored Fuzzy Inference System for representing and reasoning about SoS architectures under Epistemic Uncertainty



5. Conceiving the Fuzzy SoS Architecture Description Language

Enhancing SosADL with Fuzzy Constructs for Describing SoS Architectures under Epistemic Uncertainty

5.1 Enhancing SosADL with Fuzzy Constructs: The Fuzzy SosADL

- To make possible to describe SoS architectures, subject to epistemic uncertainty, we defined Fuzzy SosADL by extending crisp SosADL with three sets of fuzzy constructs:
  - Fuzzification constructs
  - Fuzzy sets and fuzzy behavior constructs
  - Defuzzification constructs

  - The fuzzification and defuzzification constructs are necessary to bridge fuzzy mediators with constituent systems
    - In SoS architectures, mediators are dynamically created during SoS operation for raising suitable emergent behavior
  - Fuzzy sets and fuzzy behavior constructs provide the mechanisms for computing with uncertainty via fuzzy rules

5.2 Fuzzy Constructs: Fuzzy Actions in Fuzzy SosADL

- A fuzzy action can be a fuzzify action, a rulefy action, or a defuzzify action
- A fuzzify action converts a crisp value into a fuzzy value in function of fuzzy terms of a fuzzy datatype
- A rulefy action declares fuzzy implications to specify a behavior
    - It defines the operation, activation, and accumulation methods to be applied for implications
- A defuzzify action converts a fuzzy value into a crisp value, i.e. classical bivalent set, in which membership function returns 0 or 1

- Fuzzy actions are used in fuzzy rules

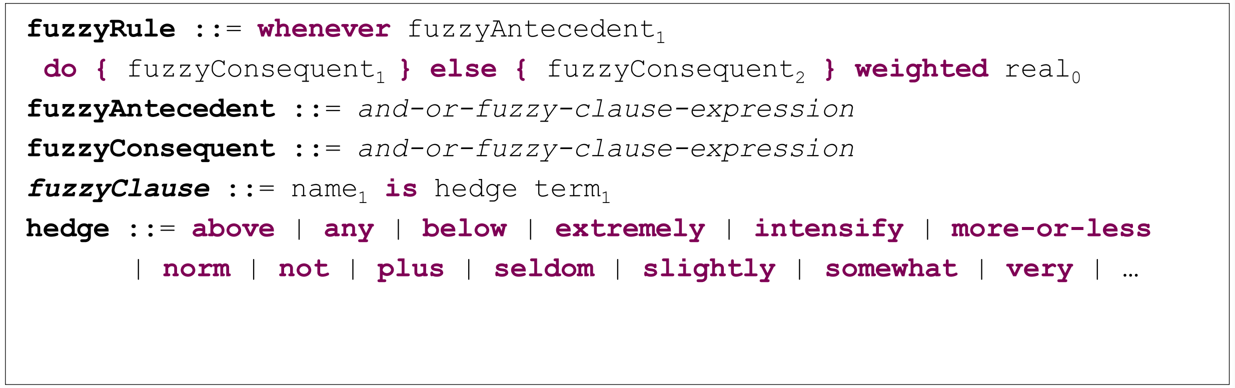
```
   fuzzyAction := fuzzifyAction | rulefyAction | defuzzifyAction
   fuzzifyAction ::= fuzzify name_1 { term_0 …, term_n }
   rulefyAction ::= rulefy name_1(name_0 : type_0 …,
   name_n : type_n) : [name_0 : type_0 …, name_n : type_n]
   aggregating by operationMethod_0..2 activating by activationMethod_0
   accumulating by accumulationMethod_0 { fuzzyRule_1 … fuzzyRule_n }
   defuzzifyAction ::= defuzzify name_1 [inf_1..sup_1]
   by defuzzifierMethod_1 { term_0 …, term_n } default real_0
   operationMethod ::= andMethod | orMethod

// t-norm for logical operator and

   andMethod ::= #and 'minimum' | 'product' | 'bounded difference'
   | 'drastic product' | 'einstein product' | 'hamacher product'
   | 'nilpotent minimum' | …

// t-conorm for logical operator or

   orMethod ::= #or 'maximum' | 'probabilistic sum' | 'bounded sum'
   | 'drastic sum' | 'einstein sum' | 'hamacher sum'
   | 'nilpotent maximum' | …
   activationMethod ::= andMethod
   accumulationMethod ::= orMethod
   defuzzifierMethod ::= 'mean of maxima' | 'left most maximum'
   | 'right most maximum' | 'centre of gravity' | 'centre of area'
   | 'centre of gravity on singletons' | …
```

Abstract syntax of fuzzy actions for Fuzzy SosADL

5.3 Fuzzy Rule Constructs in Fuzzy SosADL

- A fuzzy rule is a fuzzy expression in the form of whenever-do[-else] where the whenever part declares the antecedent of the implication, and the do[-else] part the consequent
  - They involve fuzzy variables and fuzzy operators
- A fuzzy expression is a n-ary relation from n fuzzy sets to the interval from 0 to 1 as well as when combined with the logical operators and, or, and not are also fuzzy expressions
- A rule weight is a value between 0 and 1 that states the weight degree of a fuzzy rule in a fuzzy inference
- A fuzzy inference is the process that uses fuzzy logic to map a given input to an output based on fuzzy rules

- **The nature of consequent part of fuzzy rules enables to define different kinds of fuzzy controllers, including: Mamdani, Takagi-Sugeno-Kang (TSK), Tsukamoto, and anYaRuleBase**



```
fuzzyRule ::= whenever fuzzyAntecedent_1
 do { fuzzyConsequent_1 } else { fuzzyConsequent_2 } weighted real_0
fuzzyAntecedent ::= and-or-fuzzy-clause-expression
fuzzyConsequent ::= and-or-fuzzy-clause-expression
fuzzyClause ::= name_1 is hedge term_1
hedge ::= above | any | below | extremely | intensify | more-or-less
       | norm | not | plus | seldom | slightly | somewhat | very | …
```

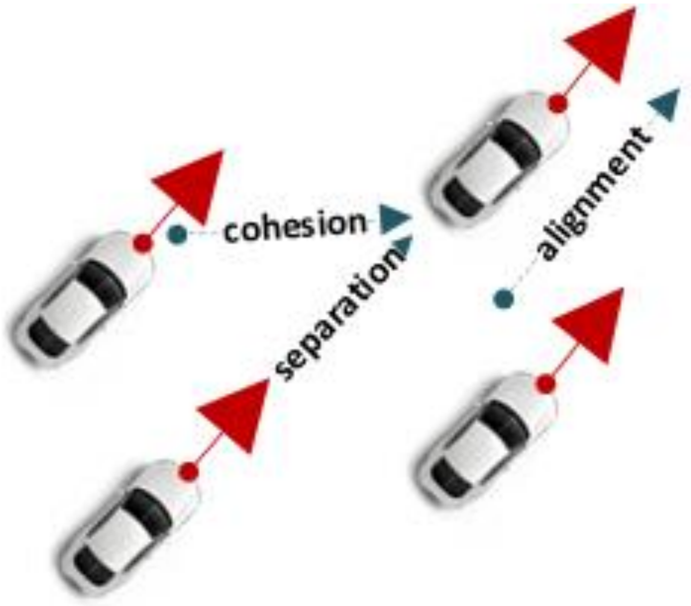
Abstract syntax of fuzzy rules for Fuzzy SosADL

6. Applying Fuzzy SosADL in the Motivating Case Study

Describing the SoS Architecture for “Platooning of Self-driving Vehicles” with SosADL enhanced with Fuzzy Theory for handling Epistemic Uncertainty

6.1 Fuzzy Architecture Description for Platooning SoS: Micro-scale Behaviors for Creating the Emergent Behavior of Platooning

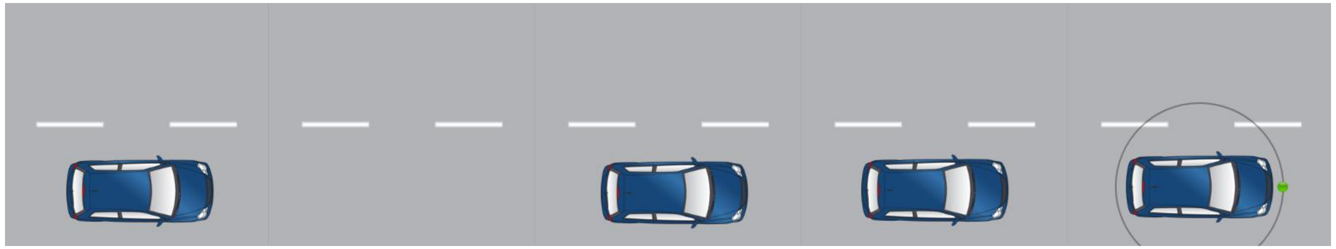
- In a Platooning SoS, the **emergent behavior of platooning results from three micro-scale behaviors of platoonmates** which together enforce the constraints that are required for enabling self-organization of platoonmates:
  - **Cohesion behavior:** every platoonmate must steer to follow the platoonmate just in front of it (if any)
  - **Separation behavior:** every platoonmate must steer to avoid the near platoonmate in front of it, thereby avoiding 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



Oquendo, F.: "Formally Describing Self-organizing Architectures for Systems-of-Systems on the Internet-of-Things", Proc. of the 12th European Conference on Software Architecture (ECSA), Springer, Madrid, Spain, September 2018

6.2 Fuzzy Architecture Description for Platooning SoS: Constituents and Mediators in Vehicle Platoon

- The Platooning SoS architecture is described with Fuzzy SosADL in terms of:
  - Self-driving vehicles: they are the SoS constituents identified at run-time – constituent systems are declared at design-time by their system capabilities in Fuzzy SosADL
  - Fuzzy mediators for platooning: they are the SoS mediators created at run-time – they are declared at design-time by their mediation capabilities as fuzzy theories in Fuzzy SosADL
    - Mediators are created at run-time (concretized by the SoS) to achieve a goal (in this case forming and maintaining platoons), part of an encompassing mission
    - From the viewpoint of the self-driving vehicle, each mediator generates an independent request for a steering maneuver to be executed by the mediated self-driving vehicle as a result of a micro-scale behavior
  - The architectural role of fuzzy mediators is to mediate the interaction of constituent systems for creating the requested emergent behavior of e.g. platooning under uncertainty

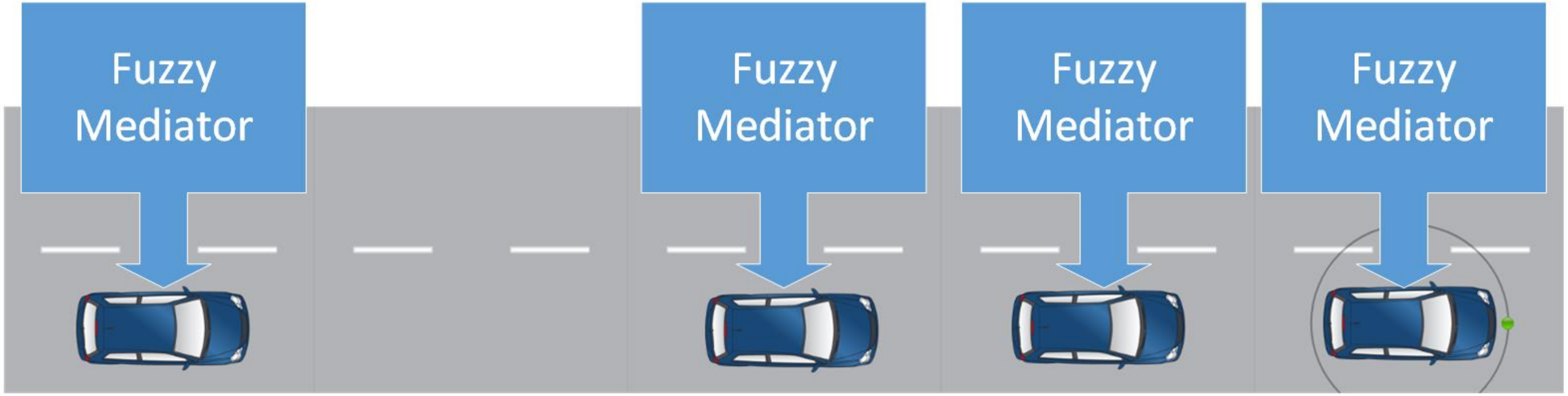


6.3 Fuzzy Architecture Description for Platooning SoS: Fuzzy Mediator for Platooning

- Fuzzy mediators are defined in SosADL enhanced with Fuzzy Theory
  - A self-driving vehicle (as platoonmate) uses its sensors, including radars/lidars, to acquire information about its physical environment and its neighbors
  - The fuzzy datatypes are declared to express uncertainty in input values from sensors and output commands for actuators
  - Based on the distance and relative positions of two sequent self-driving vehicles, the fuzzy mediator decides which maneuvers to apply to the mediated self-driving vehicle to create and maintain platooning, regulating its distance to platoonmates
  - The fuzzy rulesets are declared to be able to handle fuzzy values (defined by fuzzy sets) for representing the relative position of the mediated self-driving vehicle as well as the positions of the other self-driving vehicles in neighborhood – they provide the mechanism for computing the maneuvers to apply to the mediated self-driving vehicle, i.e. the steering commands

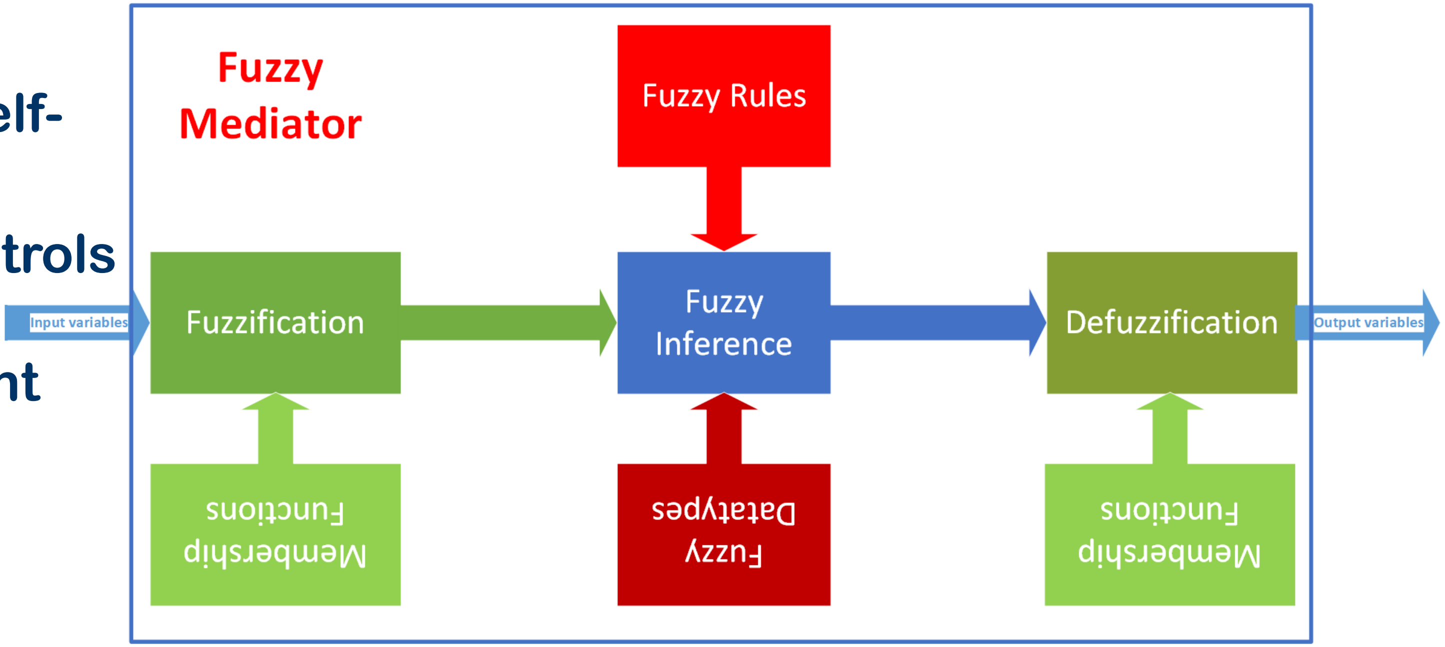
6.4 Fuzzy Architecture Description for Platooning SoS: Fuzzy Mediator per Platoonmate in the SoS

- Architecturally, fuzzy mediators are declared as part of the definition of an SoS coalition
- In the case of platooning, each fuzzy mediator mediates how a self-driving vehicle interacts with other self-driving vehicles based only on its local view of the operational environment to achieve the platooning emergent behavior
- Fuzzy mediators behave as coordinative controllers in complement of the own self-driving vehicle controllers

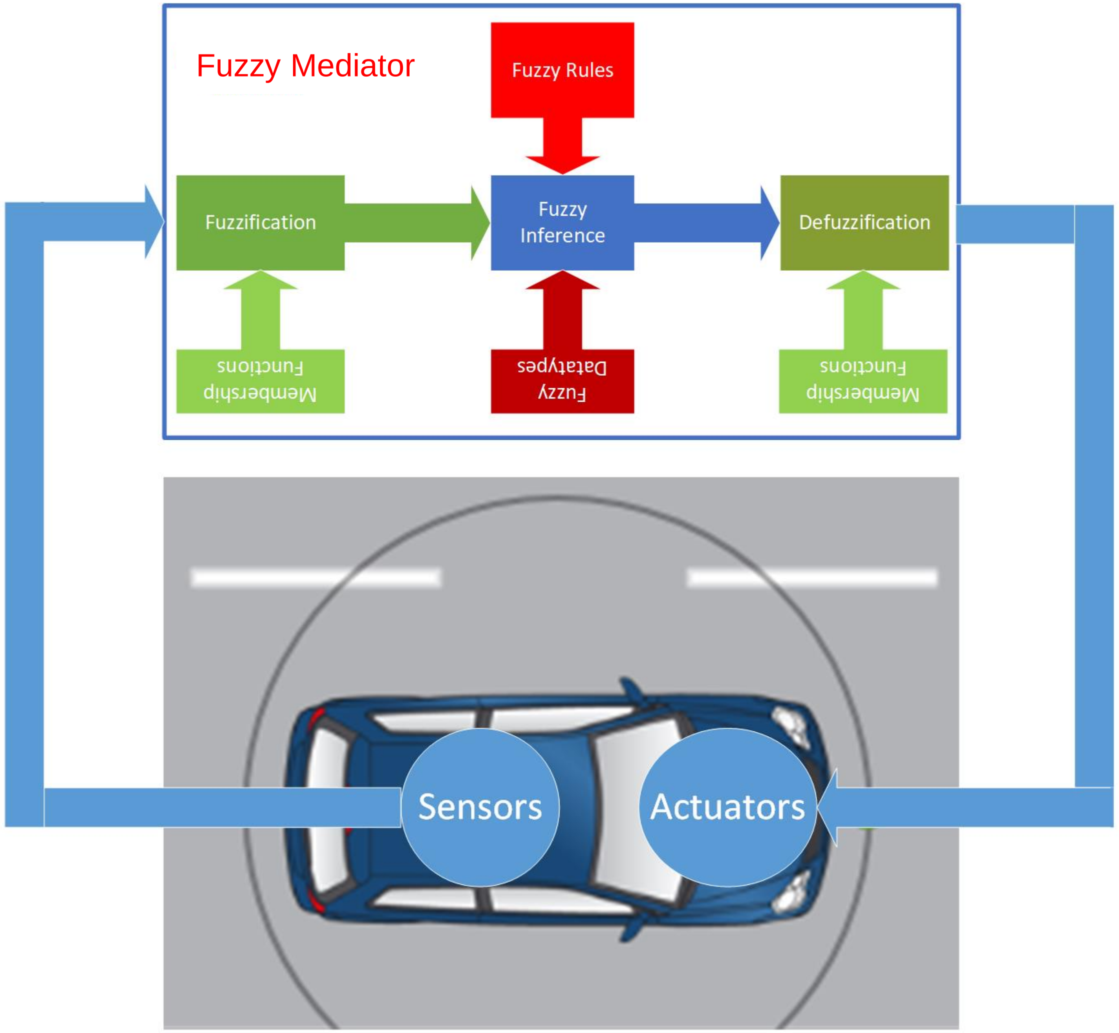


6.5 Fuzzy Architecture Description for Platooning SoS: Inside of Fuzzy Mediators (a)

- The **Fuzzy Mediator** continuously take input values from sensors of the self-driving vehicle and compute the steering commands contributing to SoS platooning
  - **Fuzzification** of the crisp values received from the sensors of the self-driving vehicle
  - **Fuzzy inference** of which fuzzy controls to apply for platooning in terms of cohesion, separation, and alignment based on the applied fuzzy rules according to the defined fuzzy datatypes and their membership functions
  - **Defuzzification** of the computed fuzzy values for obtaining crisp values to command the self-driving vehicle

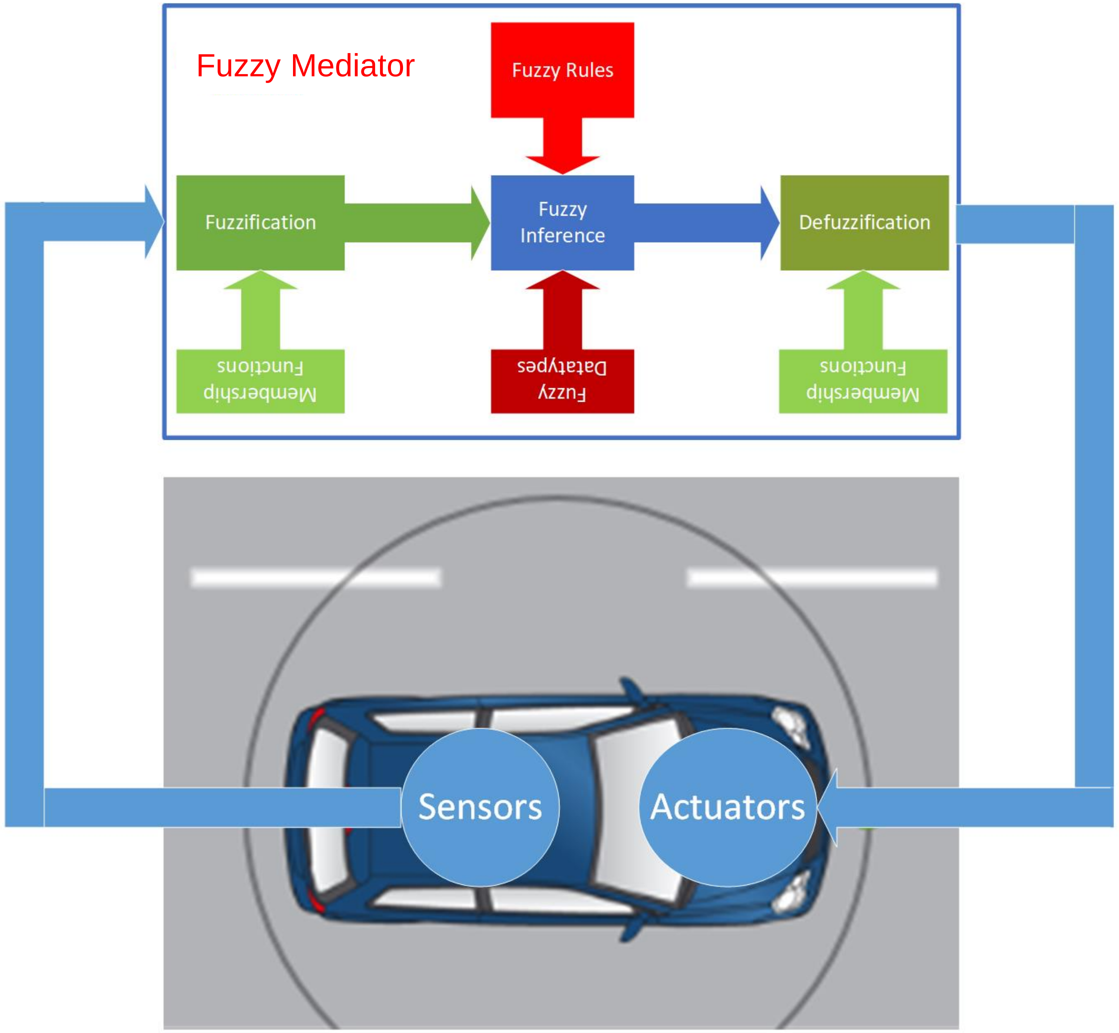


6.5 Fuzzy Architecture Description for Platooning SoS: Inside of Fuzzy Mediators (b)



- With the data received from the radar/lidar, the **Fuzzy Mediator** applies the three rule sets, i.e. **cohesion, separation, alignment**, to determine how the mediated self-driving vehicle must behave
- Each resulting **fuzzy value** determines through a **vector of speed and heading** how the self-driving vehicle will be commanded to achieve cohesion, separation, and alignment

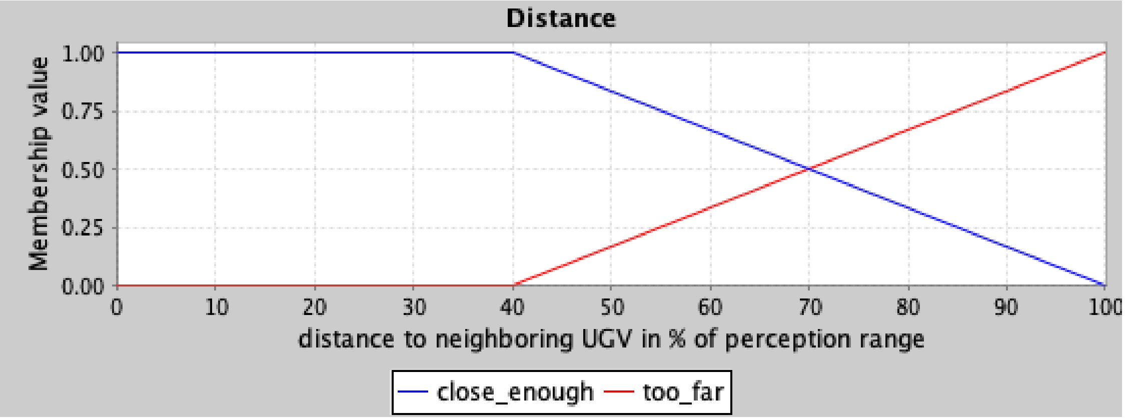
6.6 Fuzzy Architecture Description for Platooning SoS: Fuzzy Datatypes for Sensors (a)



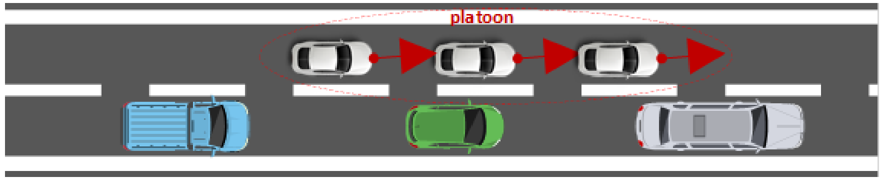
- A platoonmate use its sensors to acquire information about its physical environment and its neighbors
- As there is an intrinsic imprecision on the measurements, fuzzy datatypes are needed for:
  - measuring the distance to other platoonmates in the neighborhood, hereafter named Distance
  - measuring the relative heading of other platoonmates in the neighborhood in terms of angular offset in degrees, hereafter named AngularOffset

6.6 Fuzzy Architecture Description for Platooning SoS: Fuzzy Datatype for Distance (b)

- The Distance fuzzy datatype expresses the fuzzy distance to another self-driving vehicle in the neighborhood, as there are intrinsic uncertainties on the distance measurements from radar/lidar as well as on the weather conditions: close_enough, too_far
- E.g., Distance is expressed in terms of percentage (%) of the platoonmate perception range



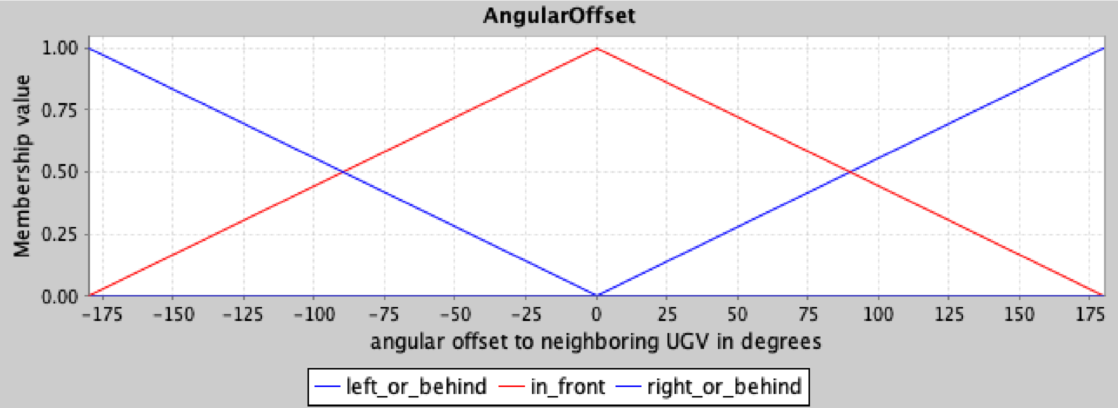
```
//Fuzzy is the fuzzy SosADL library
with Fuzzy
//user-defined library for platoonmates
library FuzzyDatatypesForPlatooning is {
 //Fuzzy set for distance 'close enough'
 fuzzy value close_enough on [0,100] is {
  (0,1),(40,1),(100,0)}
 //Fuzzy set for distance 'too far'
 fuzzy value too_far on [0,100] is {
  (0,0),(40,0),(100,1)}
 //Fuzzy datatype for platoonmate distance
 fuzzy datatype Distance on [0,100] is {
  close_enough,too_far}
  with {…}
 …
}
```



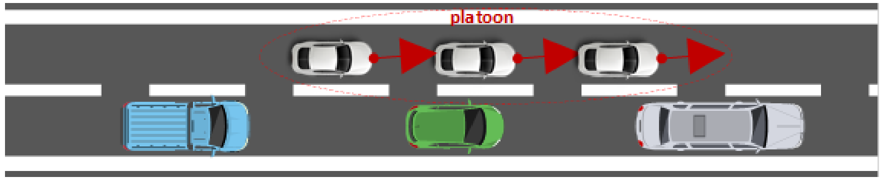
6. Fuzzy Architecture Description for Platooning SoS

6.6 Fuzzy Architecture Description for Platooning SoS: Fuzzy Datatype for Angular Offset (c)

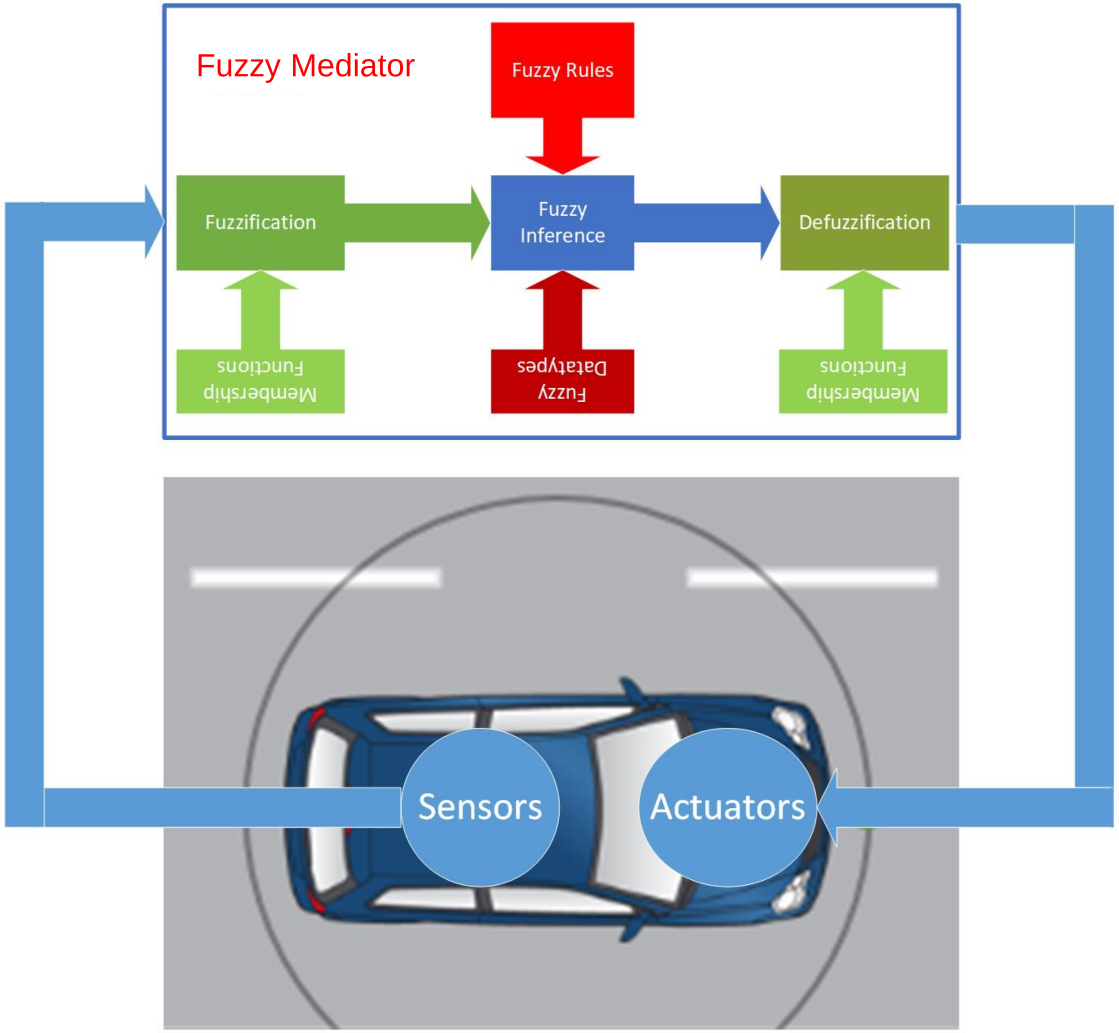
- The AngularOffset fuzzy datatype expresses the angular offset of a given self-driving vehicle relative to another one according to its perception position: left_or_behind, in_front, right_or_behind
- Angular Offset is expressed in terms of degrees (°) of platoonmate perception angle
  - E.g., If the angular offset to the perceived neighboring platoonmate is 0° of the perception angle, then the perceived platoonmate is 'in front' (membership degree is 1)



```
with Fuzzy
library FuzzyDatatypesForPlatooning is {
  …
 //Fuzzy set for angular offset 'left or behind'
 fuzzy value left_or_behind on [-180,180] is {
  (-180,1),(0,0),(180,0)}
 //Fuzzy set for angular offset 'right or
behind'
 fuzzy value right_or_behind on [-180,180] is {
  (-180,0),(0,0),(180,1)}
 fuzzy value in_front on [-180,180] is {
  (-180,0),(0,1),(180,0)}
 //Fuzzy datatype for platoonmate angular offset
 fuzzy datatype AngularOffset on [-180,180] is {
  left_or_behind,in_front,right_or_behind}
  with {…}
 …
}
```



6.7 Fuzzy Architecture Description for Platooning SoS: Fuzzy Datatypes for Actuators (a)

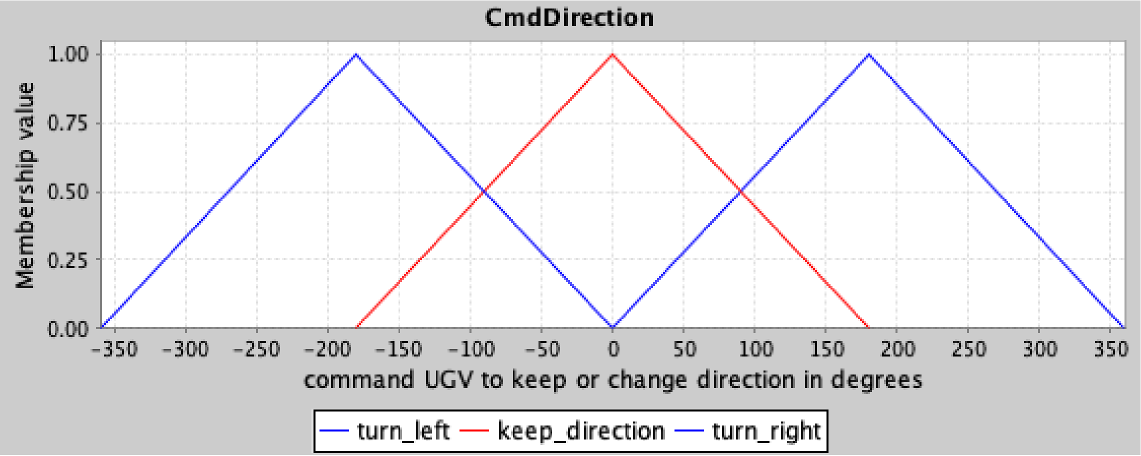


- A platoonmate use its actuators to steer the self-driving vehicle in its physical environment
- As there is an intrinsic imprecision on the steering maneuvers, fuzzy datatypes are needed for commanding:
  - the heading of the platoonmate in the platoon, hereafter named CmdDirection
  - the speed of the platoonmate in the platoon, hereafter named CmdSpeed

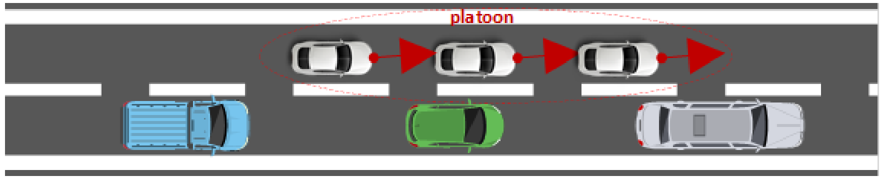
6. Fuzzy Architecture Description for Platooning SoS

6.7 Fuzzy Architecture Description for Platooning SoS: Fuzzy Datatype for commanding Heading (b)

- The CmdDirection fuzzy datatype expresses the commands regarding the heading of a self-driving vehicle in the platoon
- It is declared by three interval-based fuzzy sets: turn_left, turn_right and keep_direction



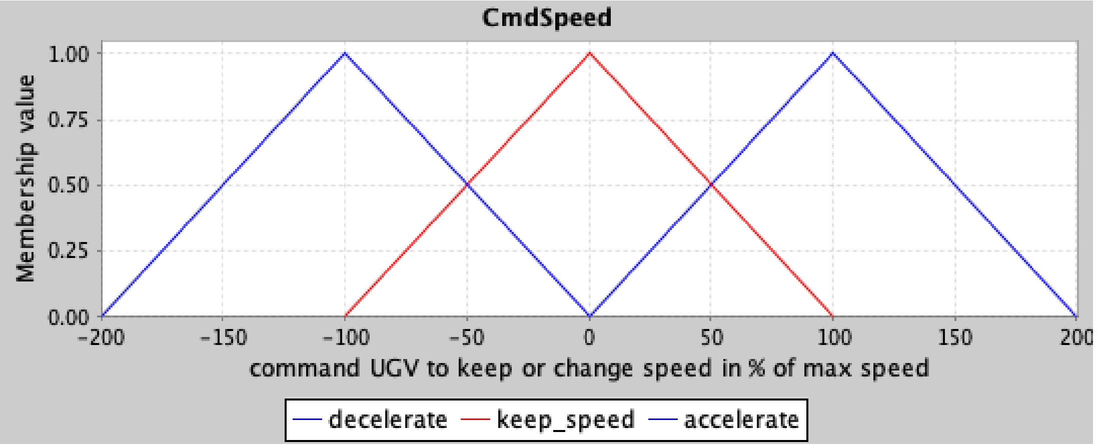
```
with Fuzzy
library FuzzyDatatypesForPlatooning is {
 …
 //Fuzzy set for commanding ‘turn left’
 fuzzy value turn_left on [-360,360] is {
  (-360,0),(-180,1),(0,0),(360,0)}
 //Fuzzy set for commanding ‘turn right’
 fuzzy value turn_right on [-360,360] is {
  (-360,0),(0,0),(180,1),(360,0)}
 //Fuzzy datatype for commanding ‘keep direction’
 fuzzy value keep_direction on [-360,360] is {
  (-360,0),(-180,0),(0,1),(180,0),(360,0)}
 //Fuzzy datatype to command platoonmate heading
 fuzzy datatype CmdDirection on [-360,360] is {
  turn_left,keep_direction,turn_right}
  with {…}
 …
}
```



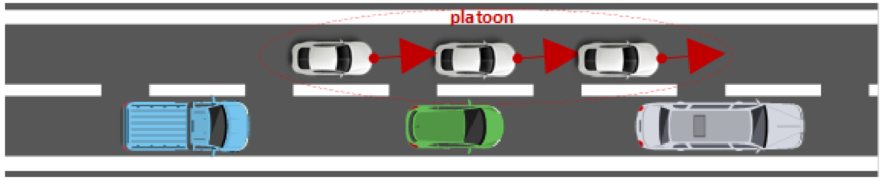
6. Fuzzy Architecture Description for Platooning SoS

6.7 Fuzzy Architecture Description for Platooning SoS: Fuzzy Datatype for commanding Speed (c)

- The CmdSpeed fuzzy datatype expresses the commands regarding the speed of a self-driving vehicle in the platoon
- It is declared by three interval-based fuzzy sets: decelerate, keep_speed, and accelerate



```
with Fuzzy
library FuzzyDatatypesForPlatooning is {
 …
 //Fuzzy set for platoonmate speed ‘decelerate’
 fuzzy value decelerate on [-200,200] is {
  (-200,0),(-100,1),(0,0),(200,0)}
 //Fuzzy set for platoonmate speed ‘accelerate’
 fuzzy value accelerate on [-200,200] is {
  (-200,0),(0,0),(100,1),(200,0)}
 //Fuzzy datatype for speed ‘keep speed’
 fuzzy value keep_speed on [-200,200] is {
  (-200,0),(-100,0),(0,1),(100,0),(200,0)}
 //Fuzzy datatype to command platoonmate speed
 fuzzy datatype CmdSpeed on [-200,200] is {
  decelerate,keep_speed,accelerate}
  with {…}
 …
}
```



6. Fuzzy Architecture Description for Platooning SoS

6.8 Fuzzy Architecture Description for Platooning SoS: Rulesets for mediating behaviors (cohesion, separation, alignment) (a)

- **Fuzzy ruleset for cohesion:**
  - If a self-driving vehicle is already close enough to another self-driving vehicle, it keeps its current heading (fuzzy rule c1) and its current speed (fuzzy rule c3)
  - If a self-driving vehicle is already too far from another self-driving vehicle, it keeps its current heading (fuzzy rule c2) and its current speed (fuzzy rule c4)
  - …

```
fuzzy ruleset cohesion(distance:Distance):
[cmd_direction:Cmd_direction,cmd_speed:Cmd_speed]
aggregate by #and cmd_direction #and cmd_speed {
  //rule c1:
  when distance is close_enough
   then cmd_direction is keep_direction
  //rule c2:
  when distance is too_far
   then cmd_direction is keep_direction
  //rule c3:
  when distance is close_enough
   then cmd_speed is keep_speed
  //rule c4:
  when distance is too_far
   then cmd_speed is keep_speed
  …
}
```

6. Fuzzy Architecture Description for Platooning SoS

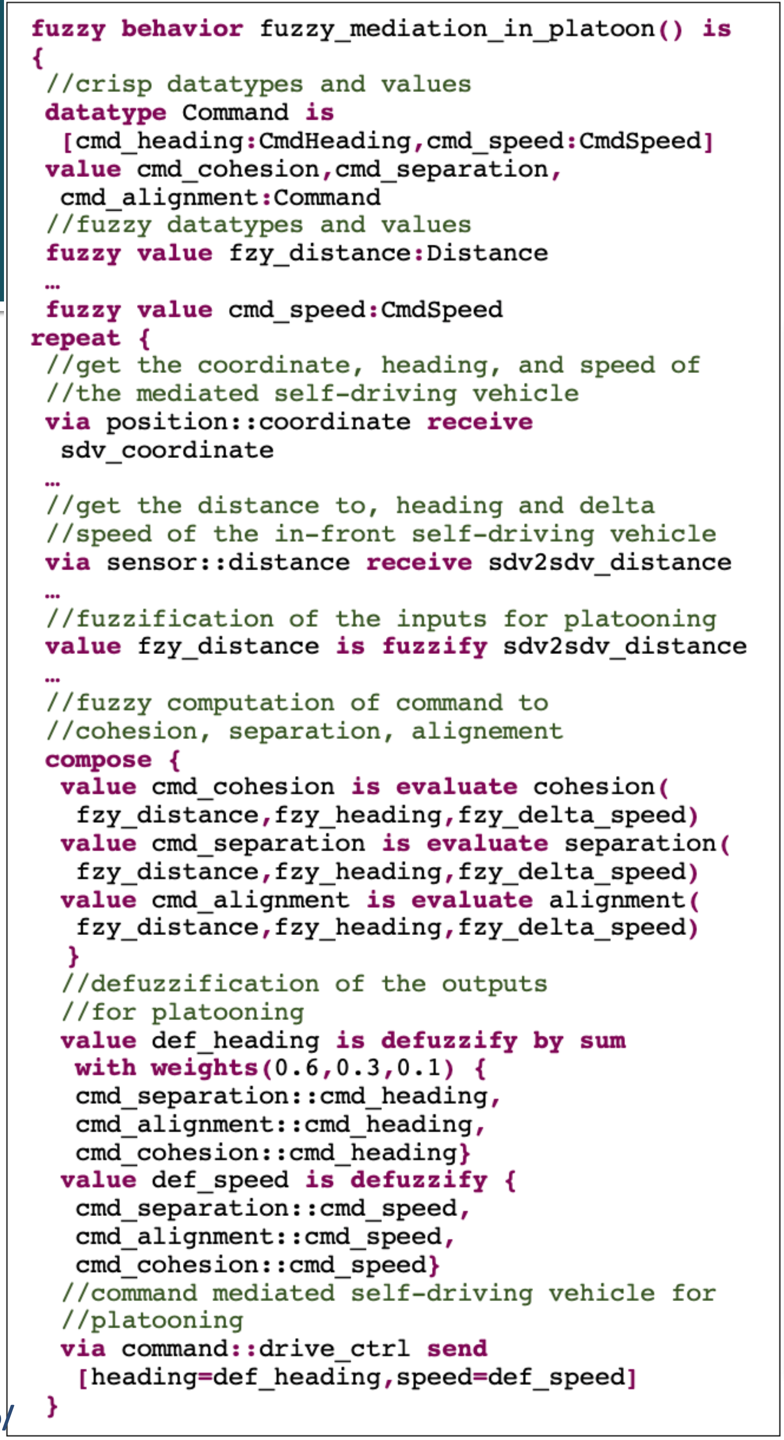
6.8 Fuzzy Architecture Description for Platooning SoS: Rulesets for mediating behaviors (cohesion, separation, alignment) (b)

- Fuzzy ruleset for cohesion: behavior for getting closer to platoonmates
  - If a self-driving vehicle is too far from another self-driving vehicle and its platoonmate is positioned in front of it, it accelerates to get closer (fuzzy rule c5)
  - If a self-driving vehicle is too far from another self-driving vehicle and its platoonmate is positioned on its left side or behind it, it turns left to get closer (fuzzy rule c6) and decelerate (fuzzy rule c7)
  - If a self-driving vehicle is too far from another self-driving vehicle and its platoonmate is positioned on its right side or behind it, it turns right to get closer (fuzzy rule c8) and decelerate (fuzzy rule c9)

```
  …
 //rule c5:
 when distance is too_far and
      position is in_front
  then cmd_speed is accelerate
 //rule c6:
 when distance is too_far and
      position is left_or_behind
  then cmd_direction is turn_left
 //rule c7:
 when distance is too_far and
      position is left_or_behind
  then cmd_speed is decelerate
 //rule c8:
 when distance is too_far and
      position is right_or_behind
  then cmd_direction is turn_right
 //rule c9:
 when distance is too_far and
      position is right_or_behind
  then cmd_speed is decelerate
}
```

6.9 Fuzzy Architecture Description for Platooning SoS: Fuzzy Mediating Behavior

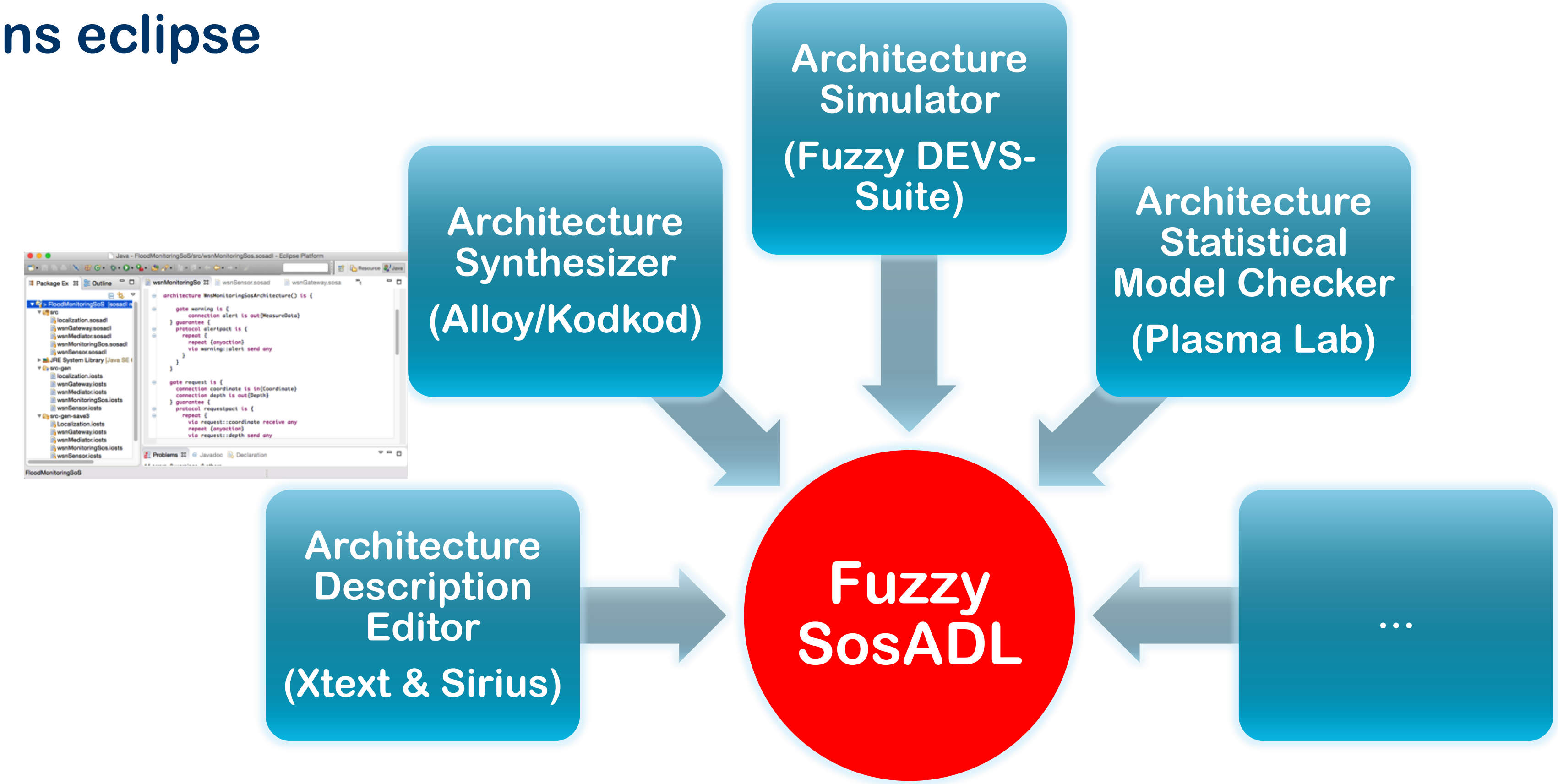
- **The fuzzy behavior for mediation in platoon:**
  - First the mediator that is created to control a given self-driving vehicle receives the position of the self-driving vehicle in terms of its coordinate (latitude and longitude), then its heading and its speed
  - Then, it receives from the mediated self-driving vehicle (equipped with radar/lidar to perceive other self-driving vehicles in its neighborhood) the distance to the closest self-driving vehicle in front of it, its relative position, and the difference of speed between the two sequent self-driving vehicles
  - Then, with the data got from the radar/lidar, **it applies the three rule sets (cohesion, separation, alignment) to determine how the mediated self-driving vehicle must behave**
  - Each fuzzy value determines through a vector of speed and heading how the self-driving vehicle will be commanded



```
fuzzy behavior fuzzy_mediation_in_platoon() is
{
 //crisp datatypes and values
 datatype Command is
  [cmd_heading:CmdHeading,cmd_speed:CmdSpeed]
 value cmd_cohesion,cmd_separation,
  cmd_alignment:Command
 //fuzzy datatypes and values
 fuzzy value fzy_distance:Distance
 …
 fuzzy value cmd_speed:CmdSpeed
repeat {
 //get the coordinate, heading, and speed of
 //the mediated self-driving vehicle
 via position::coordinate receive
  sdv_coordinate
 …
 //get the distance to, heading and delta
 //speed of the in-front self-driving vehicle
 via sensor::distance receive sdv2sdv_distance
 …
 //fuzzification of the inputs for platooning
 value fzy_distance is fuzzify sdv2sdv_distance
 …
 //fuzzy computation of command to
 //cohesion, separation, alignement
 compose {
  value cmd_cohesion is evaluate cohesion(
   fzy_distance,fzy_heading,fzy_delta_speed)
  value cmd_separation is evaluate separation(
   fzy_distance,fzy_heading,fzy_delta_speed)
  value cmd_alignment is evaluate alignment(
   fzy_distance,fzy_heading,fzy_delta_speed)
  }
  //defuzzification of the outputs
  //for platooning
  value def_heading is defuzzify by sum
   with weights(0.6,0.3,0.1) {
   cmd_separation::cmd_heading,
   cmd_alignment::cmd_heading,
   cmd_cohesion::cmd_heading}
  value def_speed is defuzzify {
   cmd_separation::cmd_speed,
   cmd_alignment::cmd_speed,
   cmd_cohesion::cmd_speed}
  //command mediated self-driving vehicle for
  //platooning
  via command::drive_ctrl send
   [heading=def_heading,speed=def_speed]
 }
```

7. Fuzzy SosADL Studio: Formal Architectural Development of SoSs in the IoT under Uncertainty using Fuzzy SosADL

- The Fuzzy SosADL Studio supports the application of Fuzzy SosADL
  - Plugins eclipse



Java - FloodMonitoringSoS/src/wsnMonitoringSos.sosadl - Eclipse Platform

```
architecture WnsMonitoringSosArchitecture() is {

    gate warning is {
        connection alert is out{MeasureData}
   } guarantee {
    protocol alertpact is {
      repeat {
        repeat {anyaction}
        via warning::alert send any
      }
    }
   }

   gate request is {
    connection coordinate is in{Coordinate}
    connection depth is out{Depth}
   } guarantee {
    protocol requestpact is {
      repeat {
        via request::coordinate receive any
        repeat {anyaction}
        via request::depth send any
```

Oquendo, F; Buisson, J.; Leroux, E.; Moguérou, G.: “A Formal Approach for Architecting Software-intensive Systems-of-Systems with Guarantees”, Proc. of the 13th IEEE System-of-Systems Engineering Conference (SoSE), Paris, France, June 2018

8. Validation by Controlled Experiment: Comparing Crisp vs. Fuzzy Architecture Descriptions

- The purpose of the experiment was to compare the effectiveness of the described SoS architectures when uncertainty is handled in SoS architecture description (with Fuzzy SosADL) against the case where uncertainty is "not" handled (with crisp SosADL)
- In the execution of the experiment, 10 SoS architects of 3 different companies located in 2 countries analyzed (each one) two descriptions of the designed SoS architecture for self-driving vehicle platooning
  - They received each two virtual machines with prepared simulation kits for evaluating these architectural descriptions regarding platooning emergent behavior
  - In the controlled experiment:
    - The SoS platoon architecture described with Fuzzy SosADL resulted in 0% collision (collision free in all simulations) and 10% of split (in high traffic caused by vehicles not pertaining to the platoon)
    - The SoS platoon architecture described with crisp SosADL resulted in 15% of collisions and 33% of split

- Overall, Fuzzy SosADL enables the description of SoS architectures under uncertainty that are more efficient (safer and more stable) than the original crisp SosADL

9. Summing Up

- In the work presented in this webinar, we have faced up the research question: "How to handle epistemic uncertainty in the description of SoS architectures?"
- To address that research question, we enhanced the SosADL with Fuzzy Theory for explicitly representing epistemic uncertainty in the description of SoS architectures through fuzzy sets, fuzzy datatypes, fuzzy actions, and fuzzy rulesets
- With Fuzzy SosADL, based on that fuzzy representation of uncertainty, the application of the conceived fuzzy mediating behaviors defined in terms of fuzzification, fuzzy inference, and defuzzification enables to compute the mediating commands
  - Through these mediating commands, each mediated constituent system is controlled to contribute to achieve the required emergent behavior
- Moreover, Fuzzy Theory supports to reason about SoS architectural properties under uncertainty
- Controlled experiment (vehicle platooning) comparing described SoS architectures under epistemic uncertainty shows the effectiveness of the defined Fuzzy SosADL

Bibliography on the SoS Architecture Description Language

1. Oquendo F.: “Formally Describing the Software Architecture of Systems-of-Systems with SosADL”, Proceedings of the 11th IEEE System-of-Systems Engineering Conference (SoSE 2016), IEEE, Kongsberg, Norway, June 2016
2. Oquendo F.: “Formally Describing the Architectural Behavior of Software-intensive Systems-of-Systems with SosADL”, Proceedings of the 21st International Conference on Engineering of Complex Computer Systems (ICECCS 2016), IEEE, Dubai, United Arab Emirates, November 2016
3. Oquendo F., Buisson J., Leroux E., Moguérou G.: “A Formal Approach for Architecting Software-intensive Systems-of-Systems with Guarantees”, Proceedings of the 13th IEEE System-of-Systems Engineering Conference (SoSE 2018), IEEE, Paris, France, June 2018
4. Oquendo F.: “Coping with Uncertainty in Systems-of-Systems Architecture Modeling on the IoT with SosADL”, Proceedings of the 14th IEEE System-of-Systems Engineering Conference (SoSE 2019), IEEE, Anchorage, Alaska, USA, May 2019
5. Oquendo F.: “Dealing with Uncertainty in Software Architecture on the Internet-of-Things with Digital Twins”, Proceedings of the 19th International Conference Computational Science and Its Applications (ICCSA 2019), Springer, Saint Petersburg, Russia, July 2019
6. Oquendo F.: "Architecting Systems-of-Systems of Self-Driving Cars for Platooning on the Internet-of-Vehicles with SosADL", Proceedings of the 2nd IFIP International Conference on Internet of Things (IoT 2019), Springer, Tampa, FL, USA, October 2019
7. Oquendo F.: “Architecting Exogenous Software-intensive Systems-of-Systems in the Internet-of-Vehicles with SosADL”, Systems Engineering, Wiley, Vol. 22(6), November 2019
8. Oquendo F.: “Fuzzy Architecture Description for Handling Uncertainty in IoT Systems-of-Systems”, Proceedings of the 15th IEEE System-of-Systems Engineering Conference (SoSE 2020), IEEE, Budapest, Hungary, June 2020

Thank You

Questions?

Fuzzy Architecture Description for Handling Uncertainty in Systems-of-Systems in the Internet-of-Things

Flavio Oquendo, Ph.D., H.D.R.
Full Professor of Computing and Software-intensive Systems Engineering &
Research Director at UMR CNRS IRISA, France
flavio.oquendo@irisa.fr
http://people.irisa.fr/Flavio.Oquendo/

Systems-of-Systems Engineering Collaborators Information Exchange
NDIA & MITRE SoSECIE Webinar - March 23, 2021