# An architectural approach for safe cooperative autonomous vehicles

António Casimiro

casim@ciencias.ulisboa.pt

http://www.di.fc.ul.pt/~casim

LASIGE, Faculdade de Ciências,

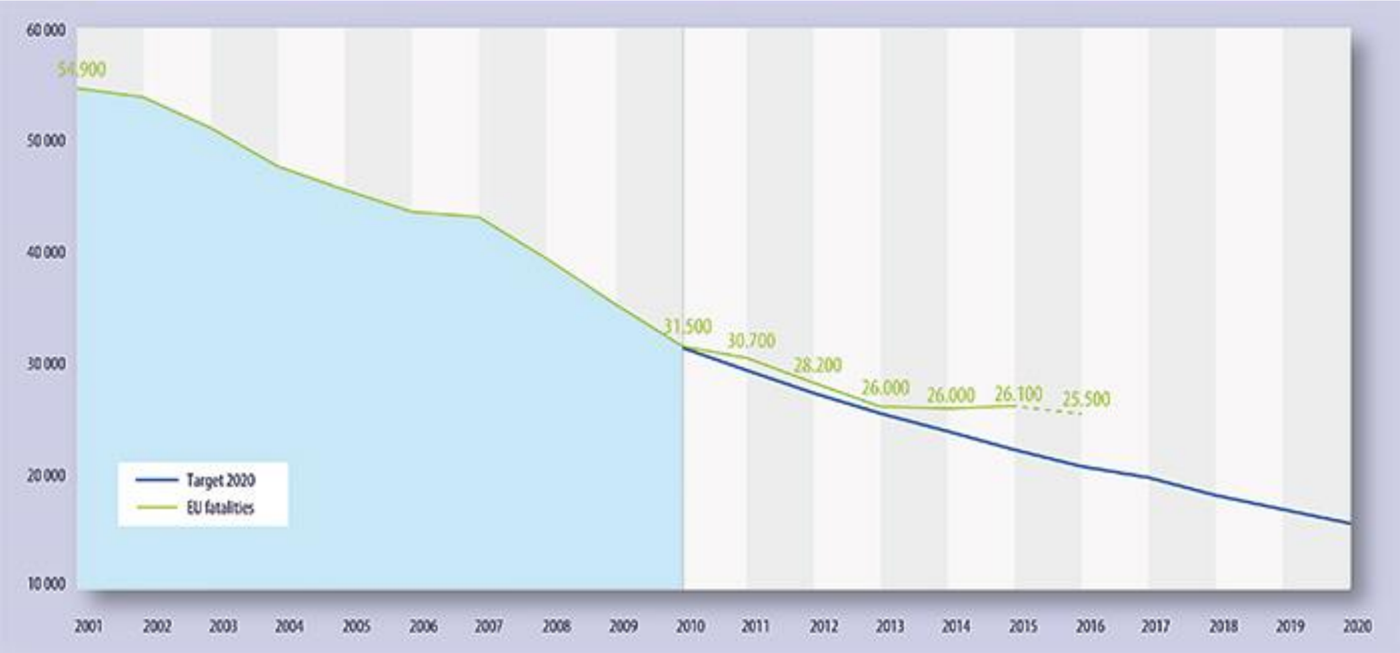
Universidade de Lisboa, Portugal

Ciências ULisboa

LASIGE driven by excellence

IFIP WG 10.4 Workshop on Assured Autonomy
Champéry, Switzerland
January 26, 2019

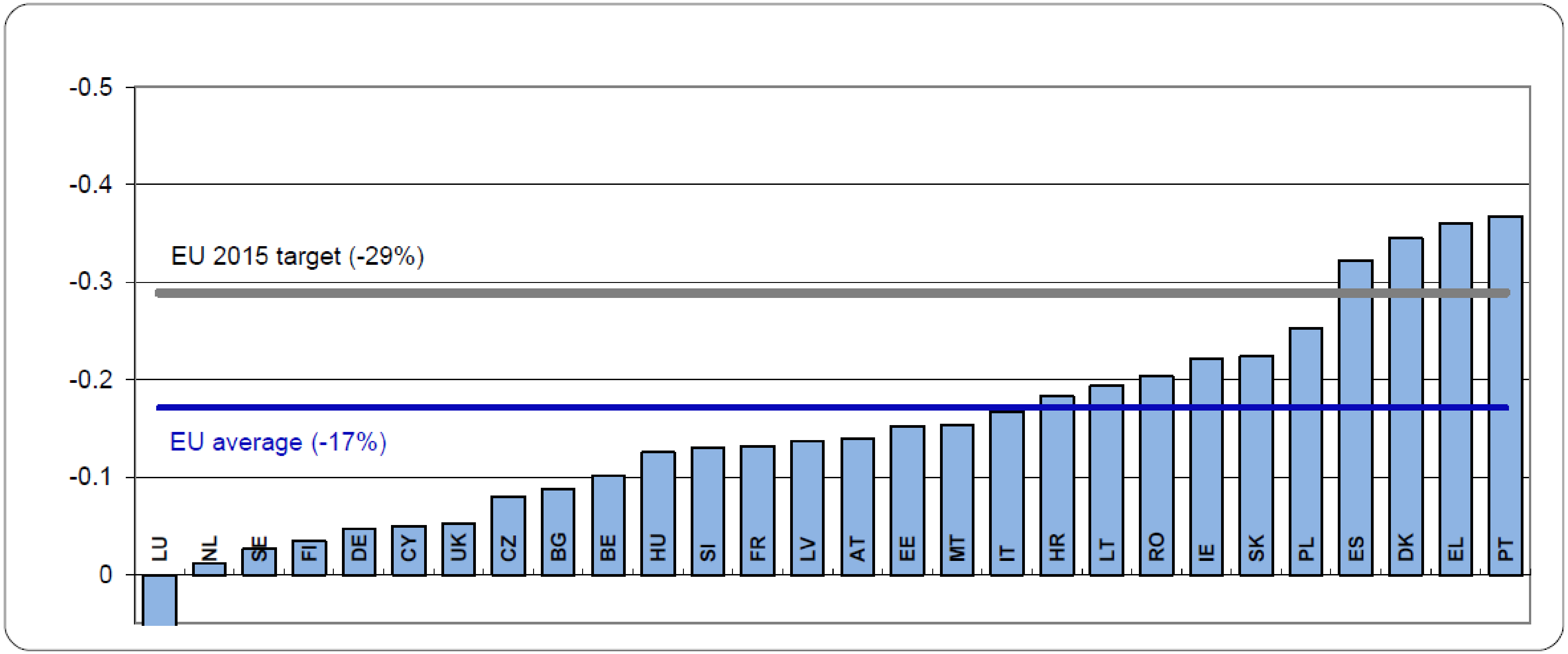
# EU road fatalities 2001-2016



Source: EC, Mobility and Transport, 2017

To bring these numbers close to zero, vehicles must become increasingly autonomous

# Evolution between 2010 and 2015



Source: EC, Mobility and Transport, 2017

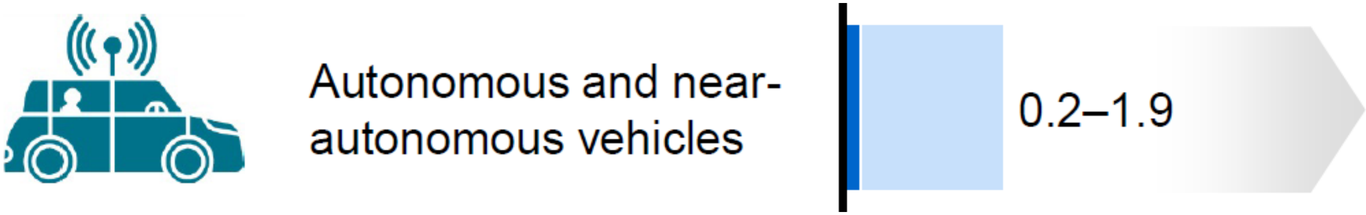
Conclusion?

Some countries had too many fatalities in 2010!

# Economic impact

- Estimated potential economic impact of autonomous vehicles in 2025:

Between $200 billion and **$1.9 trillion**

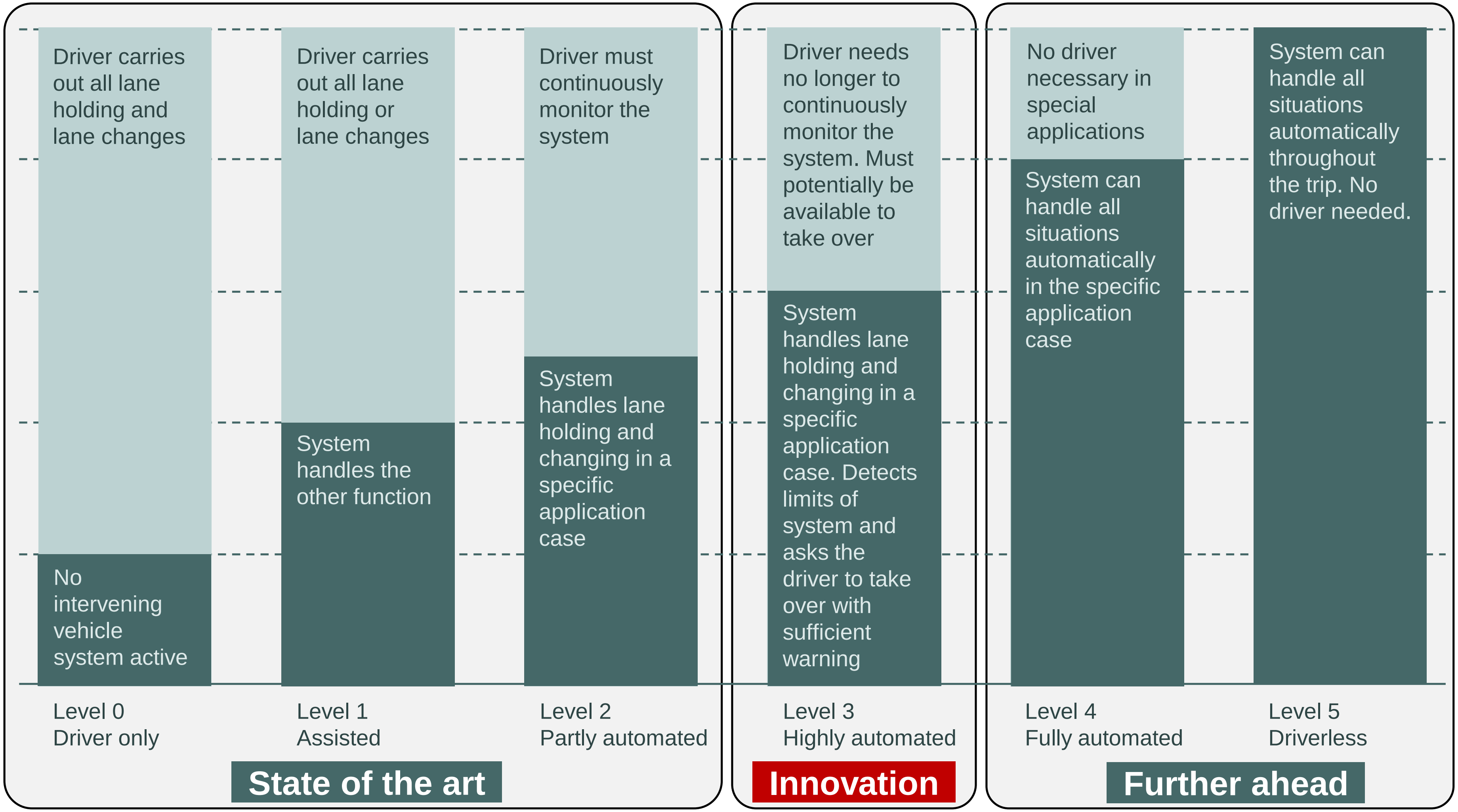


- Compare with 2013 numbers for:
  - Automobile industry revenue: $4 trillion
  - Aviation aircraft industry revenue: $155 billion

Source: McKinsey, May 2013

# Classification of autonomy

https://www.vda.de/de/themen/innovation-und-technik/automatisiertes-fahren.html



Driver carries out all lane holding and lane changes

No intervening vehicle system active

Level 0
Driver only

Driver carries out all lane holding or lane changes

System handles the other function

Level 1
Assisted

Driver must continuously monitor the system

System handles lane holding and changing in a specific application case

Level 2
Partly automated

**State of the art**

Driver needs no longer to continuously monitor the system. Must potentially be available to take over

System handles lane holding and changing in a specific application case. Detects limits of system and asks the driver to take over with sufficient warning

Level 3
Highly automated

**Innovation**

No driver necessary in special applications

System can handle all situations automatically in the specific application case

Level 4
Fully automated

System can handle all situations automatically throughout the trip. No driver needed.

Level 5
Driverless

**Further ahead**

# Are we getting there?

- Autonomous vehicles are getting increasingly autonomous, and increasingly safe
  - Google self-driving car
    - https://www.youtube.com/watch?v=TsaES--OTzM
  - Volvo self-driving car
    - https://www.youtube.com/watch?v=bJwKuWz_lkE
  - BMW, GM, Audi, Tesla, … and now also UBER!!

# Yes, but still at a significant cost!

- For **safety**, these prototype vehicles rely on:
  - **Local sensor data** – easier to ensure dependable operation, no network dependency
  - **Expensive hardware and redundancy** – for accurate context awareness and reliability
  - **Restricted operation environments** – to reduce possible hazards
  - **Restricted functional performance** – to reduce resource requirements, severity of incidents and hence safety requirements

# Google self-driving car restricted functional performance

"It struck me as cautious. It drove **slowly** and deliberately, and I got the impression that it's more likely to **annoy other drivers** than to harm them."

http://theoatmeal.com/blog/google_self_driving_car

# Google said in 2011

That the Google cars would be able to drive anywhere a car can legally drive and that the hope was to **field a fully autonomous car by the end of the decade**.

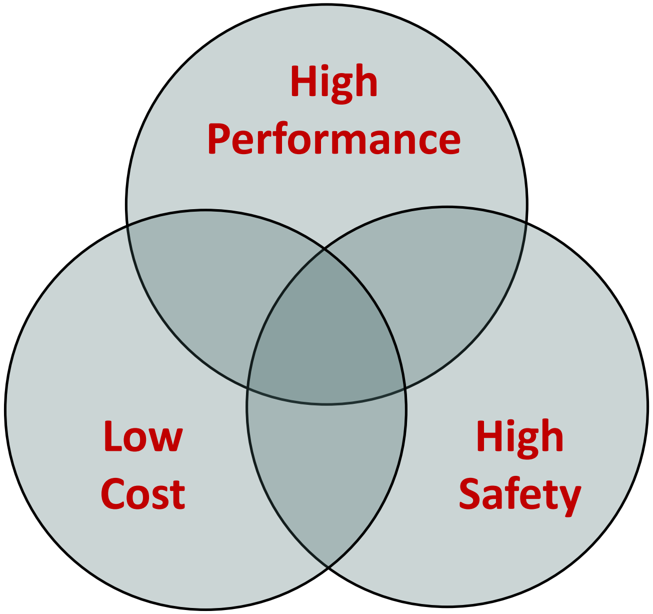
# Google now says

"How quickly can we get this into people's hands? If you read the papers, you see maybe it's three years, **maybe it's thirty years**. And I am here to tell you that honestly, it's a bit of both"

Chris Urmson, Google

# Grand challenge

- Assuring the needed **high safety**
- Using **low cost** solutions
- And achieving **high (functional) performance**
  - Possibly by employing:
    - **Complex software** solutions
    - **Vehicular cooperation**



# Challenges: Fail-operational

- There will be no driver in the loop
- Autonomous driver will have to handle all situations
- Extremely demanding requirements to sensors, actuators and computing
- No fail-safe state – system must be designed to be fail-operational for at least a limited amount of time

# Challenges: Security

- Need to prevent unauthorized access or software change
- Need to deal with an increasing amount of threats…
- …and increasing complexity of vehicle software systems, potentially introducing more vulnerabilities
- Need to manage potentially conflicting goals between security and safety

# Challenges: Big data collection

- More than **1 Gb/s** stream of produced data
- Need to **collect** and **process** a lot of data
  - Lots of sensors
  - Road maps and conditions
  - Traffic conditions
  - Weather conditions
  - Traffic signs
  - Other vehicles around the car
  - Pedestrians
  - …
- Bring cloud computing to the car: fog computing

# Challenges: Sensor fusion

- Need for accurate data

- How to classify objects?
- How to avoid false detections?
- How to avoid missed detections?

- Dependent on context/situation:
  - Amount of surrounding objects and object types
  - Lighting conditions
  - Weather
  - …

# More challenges

- **Validation**
  - Has the Google car been sufficiently validated?
  - Is it sufficient to use synthetic data and simulation?
- **SW cost integration**
  - Platforms allowing modularity, reuse, independent V&V, etc.
- **Driver interaction**
  - Before we get to driverless, drivers may still take control
  - HMI interfaces: who is driving now?
- **Legal**
  - Who is responsible when a car crashes?
- **Ethical**
  - A driver has ethics, but an “intelligent” vehicle does not…

# Cooperative vehicles challenges

- **No existing business model yet** for carmakers to incorporate cooperative functions in new vehicles
  - Who will pay for the benefit of having such cooperative functions (based on cooperative sensing)?
- New **safety risks when using external data** for decision making in safety-critical functions
  - How to ensure that received data is trustworthy and will not compromise safety?
- Strong **interoperability** is required
  - New standards must still be developed
- **Even more data being collected** through remote sensors (cooperative sensing)
  - How to manage such huge amount of information?

# Further ahead: the cooperation dimension

# FP7 KARYON project

Kernel-based ARchitecture for safetY-critical cONtrol
(2011-2014)

Provide system solutions for predictable and safe coordination of smart vehicles that autonomously cooperate and interact in an open and inherently uncertain environment

# Application domains

- Automotive domain
  - Adaptive Cruise Control Systems
  - Coordinated lane change manoeuvres
  - Coordinated intersection crossing
- Avionics domain
  - UAS/Aircraft manoeuvres in shared air space

https://youtu.be/bIKPs53eWzo

https://youtu.be/FEj2qn7XrDU

**Promo videos available on** You Tube

**http://www.youtube.com/user/KaryonProject**

# Cooperation

## And related terminology

- **Cooperation**: explicit exchange of data, allowing all participants achieving their own goals and eventually coordinate
- **Coordination**: all participants achieve their own goals with or without explicit interaction (e.g., using pre-defined rules)
- **Collaboration**: interaction towards a common goal
  - Not very appropriate for autonomous vehicles, where each vehicle has its own goals, own view of environment, etc.

# Cooperation scope

- Challenge: **How do cooperative vehicles find out the vehicles with which they need to cooperate?**
- Possible approaches
  - **Distributed solutions**: e.g. protocols for agreement on a certain group view or membership
  - **Centralized solutions**: e.g. road-side unit or cloud service that is aware of all vehicles in some area
  - **Pre-defined groups**: e.g., in platooning all vehicles know their peers

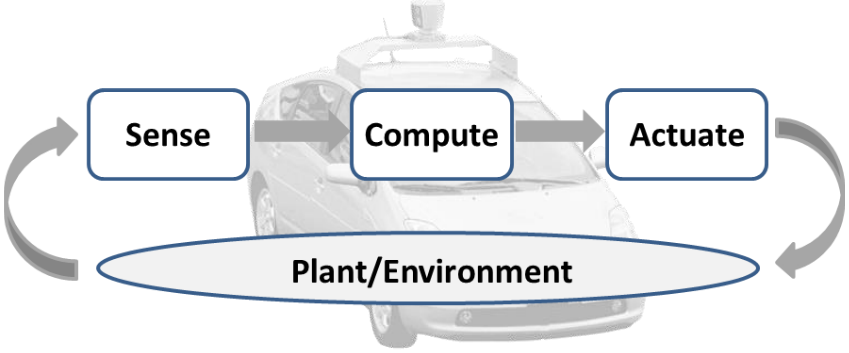
# KARYON problem statement

- **Improve functional performance** and keep safety by using more accurate context information

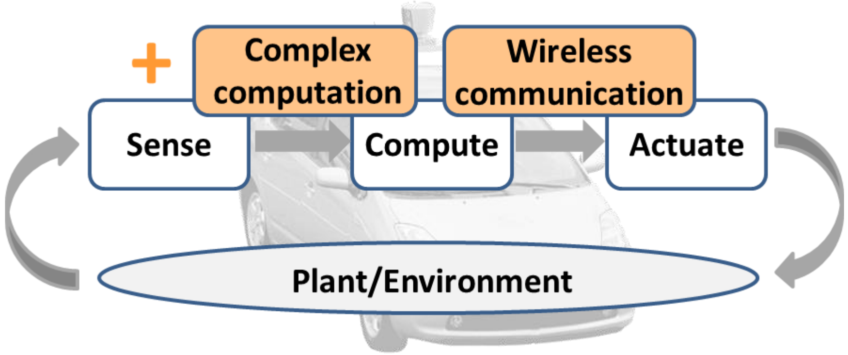
1. Exploit **cooperation** (e.g., exchange of information with nearby vehicles)
2. Exploit **complex software** solutions (e.g., environment recognition through video processing)

- Address the **temporal uncertainties** inherent to
  - Wireless communication
  - Complex processing

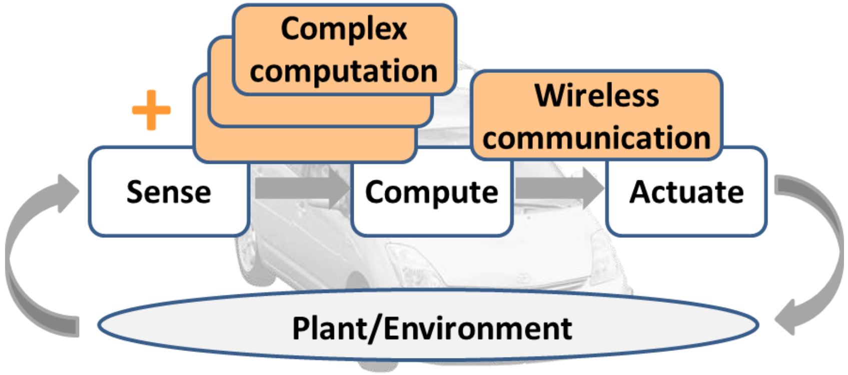
# Approach: design time



- **Level of Service 0**
- Functions are performed safely (by design, hazardous situations are excluded)

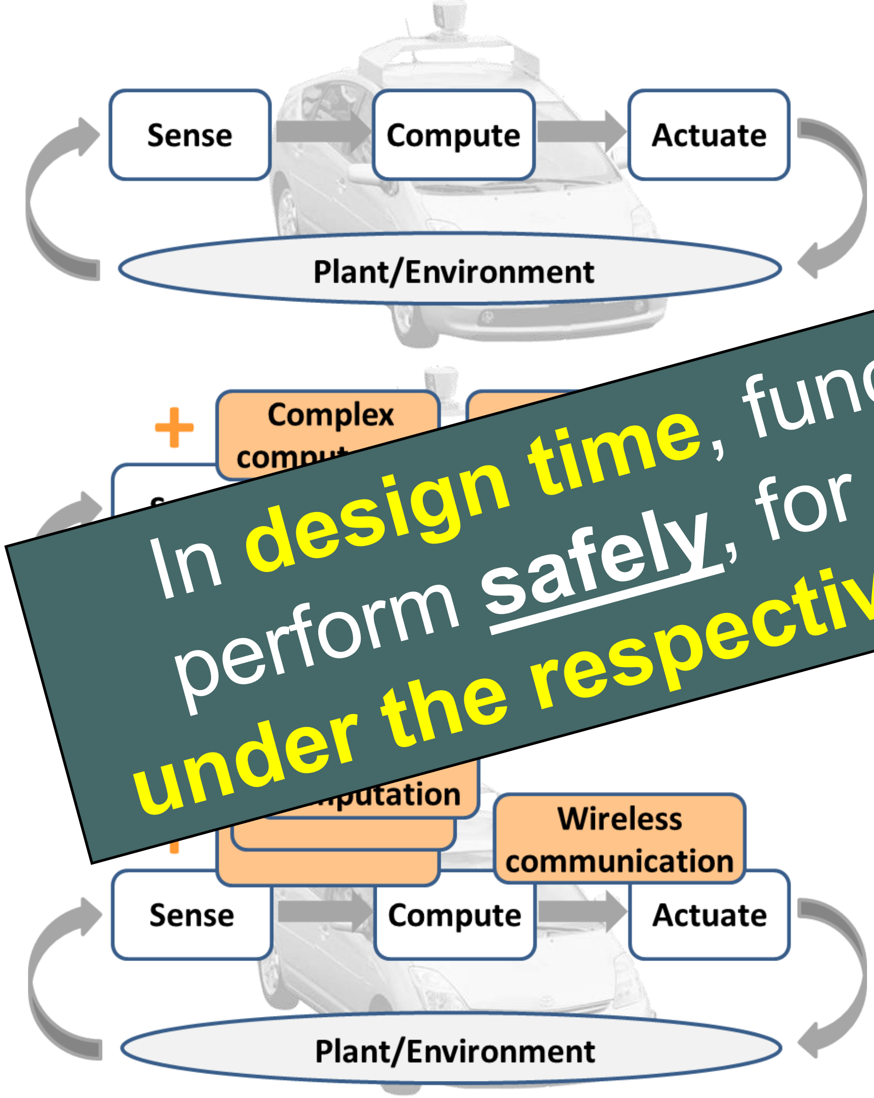


- **Level of Service 1**
- Functions are performed safely as long as some assumptions (**safety rules for LoS1**) are satisfied



- **Level of Service n**
- Functions are performed safely as long as **safety rules for LoS n** are satisfied

# Approach: design time



- **Level of Service 0**
- Functions are perf
  (by desi

ormed safely as
some assumptions (**safety rules for LoS1**) are satisfied

In **design time**, functions are proved to perform **safely**, for **each configuration under the respective set of assumptions**

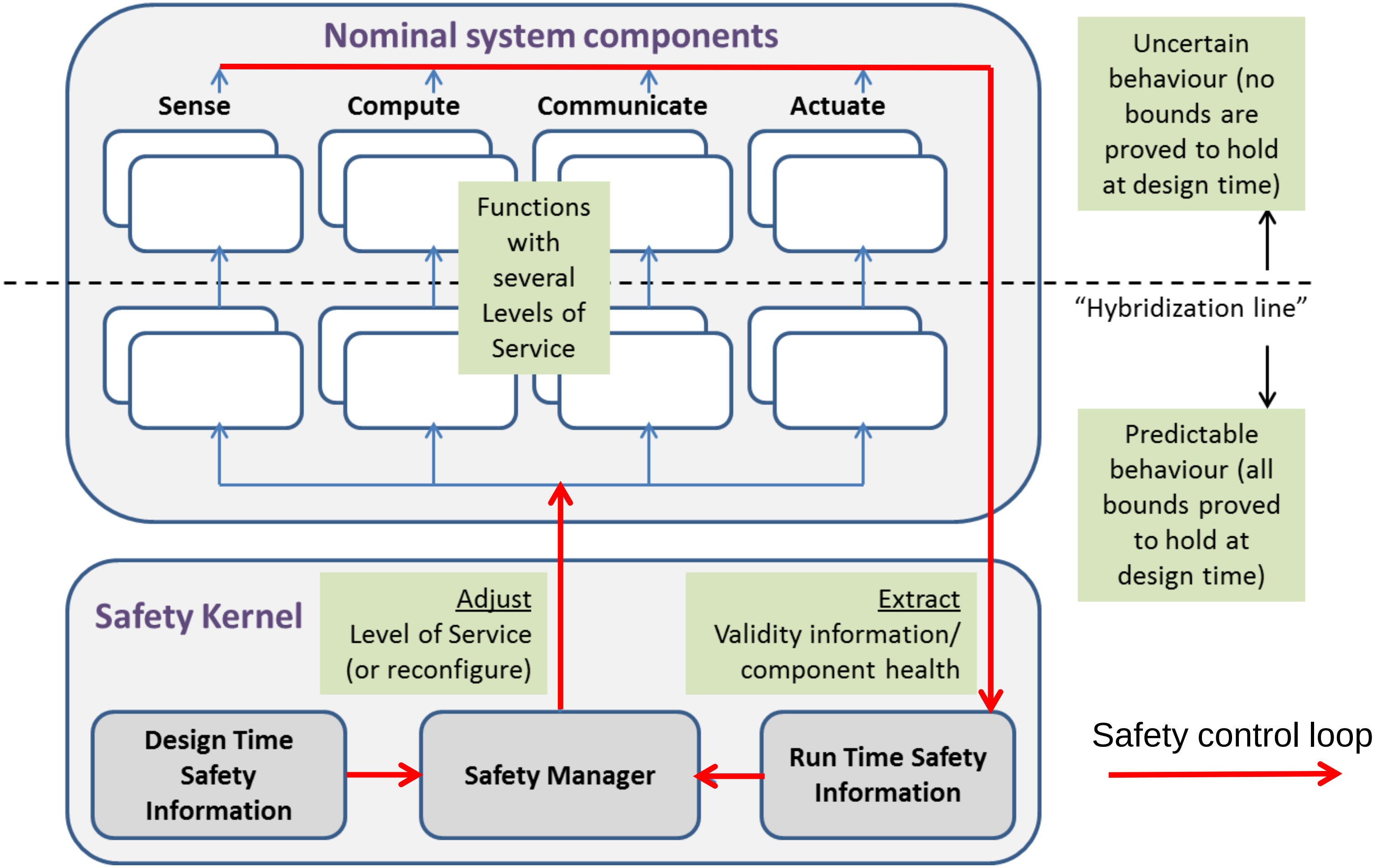
- **Level of Service n**
- Functions are performed safely as long as **safety rules for LoS n** are satisfied

# Approach: run-time

In **run-time**, safety management is performed by a **Safety Kernel**

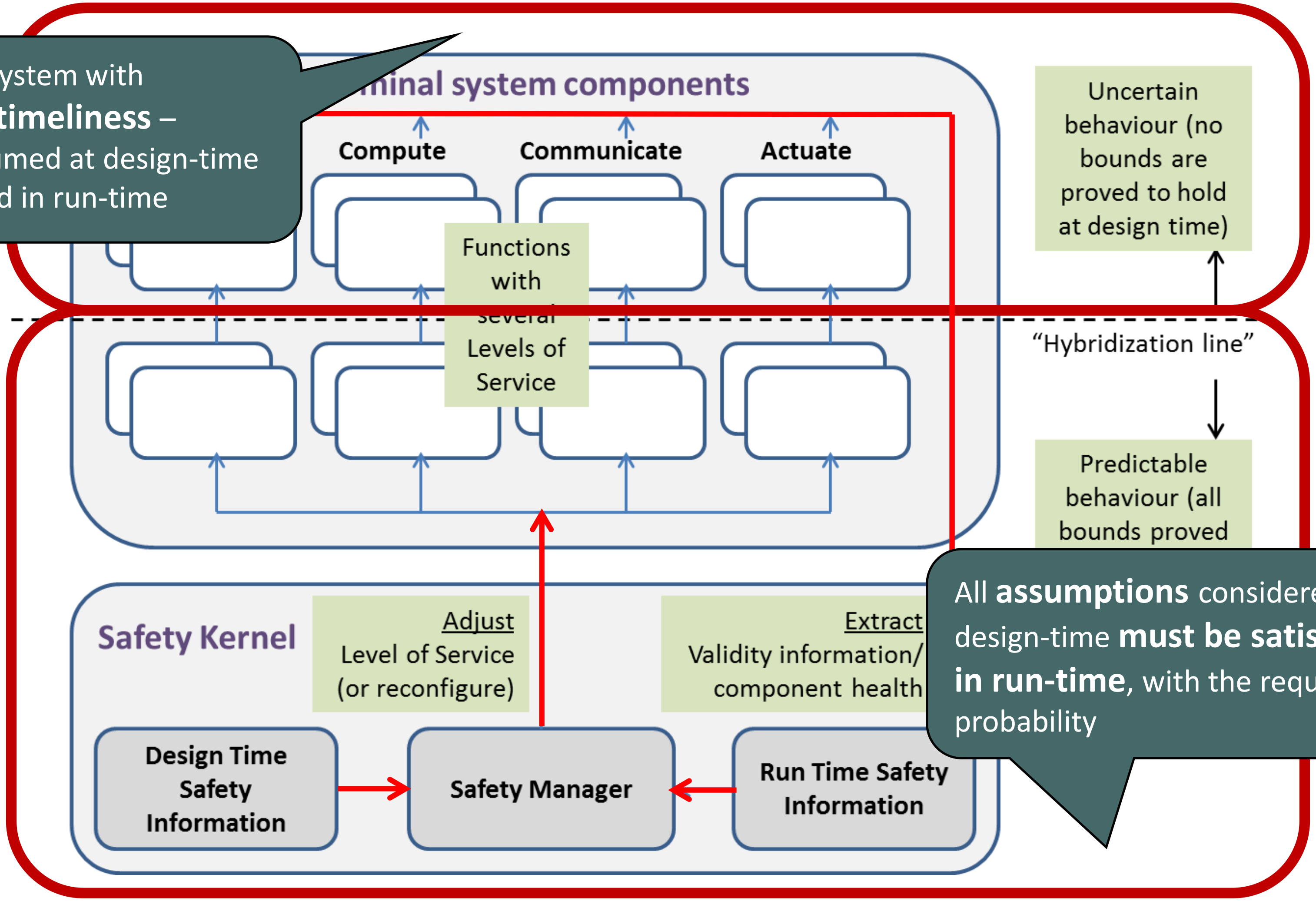
- The **Safety Kernel** **is continuously checking if safety rules are satisfied** and **determines the highest possible Level of Service (LoS)**
- For that, it collects system health data, namely:
  - The **validity** of sensor data
  - The **timeliness** of components' execution

# The KARYON architectural pattern



Nominal system components
Sense
Compute
Communicate
Actuate
Functions with several Levels of Service
Uncertain behaviour (no bounds are proved to hold at design time)
"Hybridization line"
Predictable behaviour (all bounds proved to hold at design time)
Safety Kernel
Adjust
Level of Service (or reconfigure)
Extract
Validity information/ component health
Design Time Safety Information
Safety Manager
Run Time Safety Information
Safety control loop

# The KARYON architectural pattern

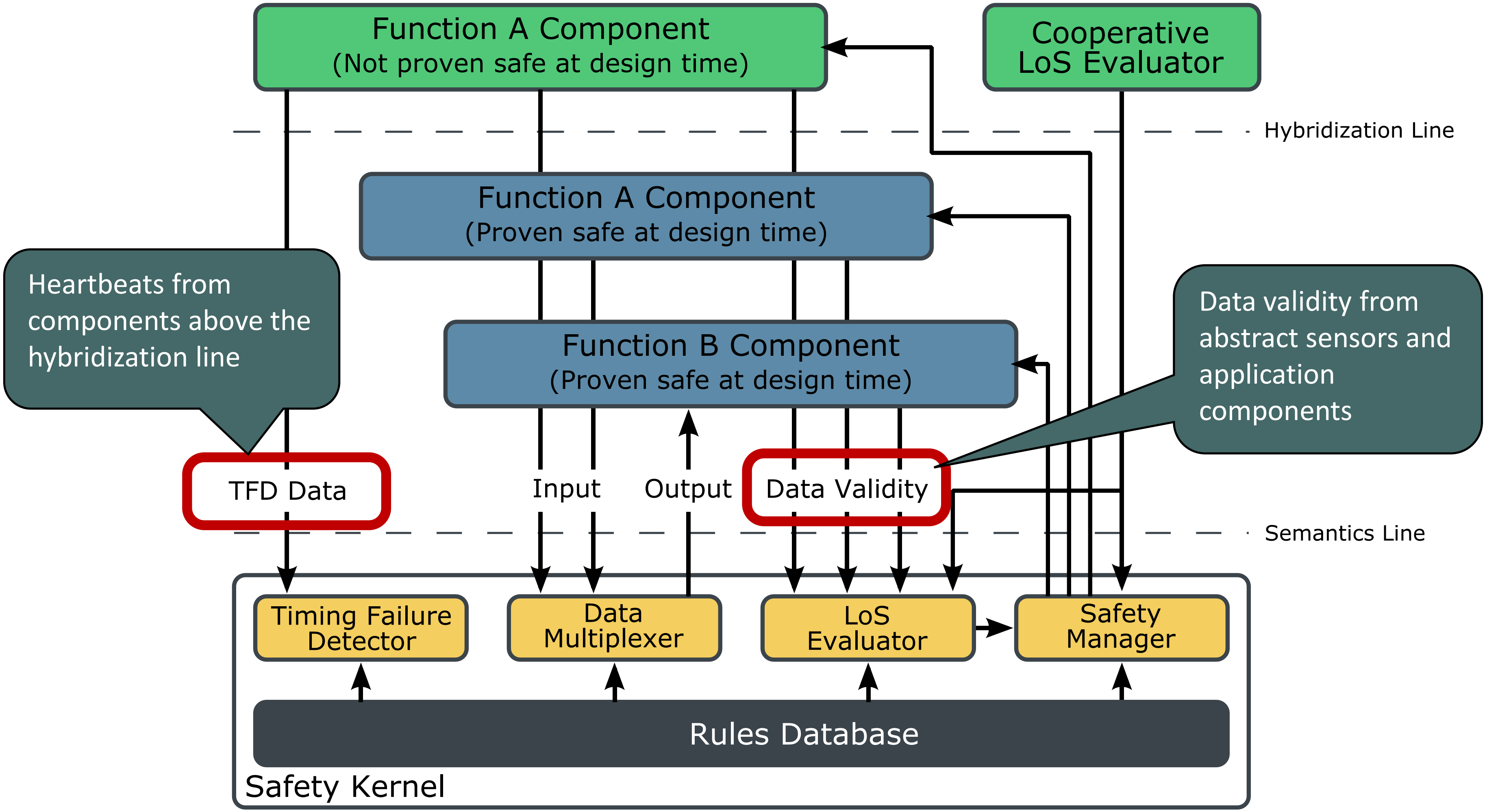


minal system components
Compute
Communicate
Actuate
Functions with several Levels of Service
Uncertain behaviour (no bounds are proved to hold at design time)
"Hybridization line"
Predictable behaviour (all bounds proved
Safety Kernel
Adjust Level of Service (or reconfigure)
Extract Validity information/ component health
Design Time Safety Information
Safety Manager
Run Time Safety Information

Part of the system with **uncertain timeliness** – bounds assumed at design-time may not hold in run-time

All **assumptions** considered in design-time **must be satisfied in run-time**, with the required probability

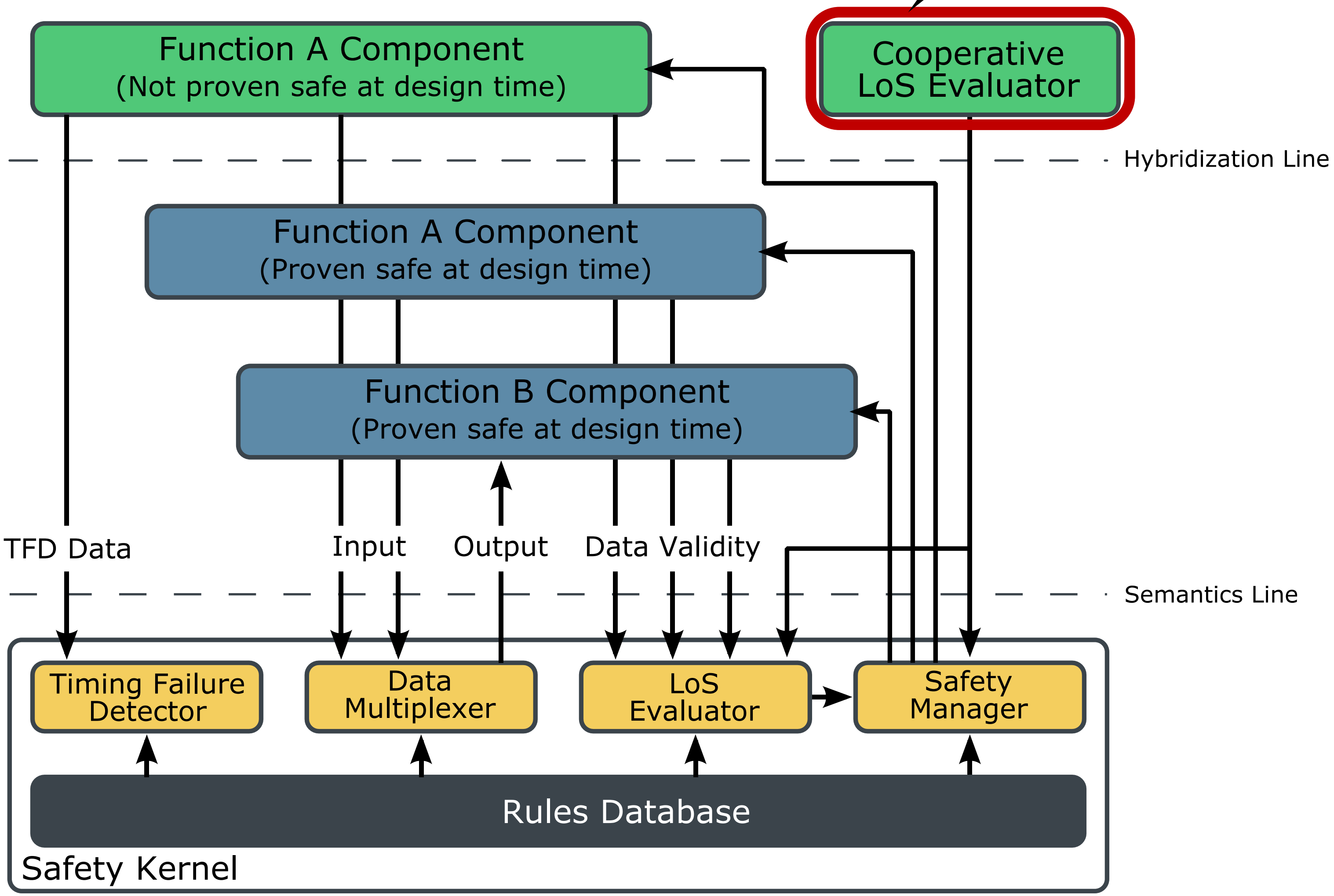
# A closer look into the SK


Function A Component
(Not proven safe at design time)
Cooperative LoS Evaluator
Hybridization Line
Function A Component
(Proven safe at design time)
Heartbeats from components above the hybridization line
Function B Component
(Proven safe at design time)
Data validity from abstract sensors and application components
TFD Data
Input
Output
Data Validity
Semantics Line
Timing Failure Detector
Data Multiplexer
LoS Evaluator
Safety Manager
Rules Database
Safety Kernel

# Cooperation with a Safety Kernel

- **The Safety Kernel is a local component**
  - It ensures that functions will be performed by the (local) system (e.g. a vehicle) at the highest possible Level of Service (LoS), given the observed timeliness of components and data validity
- Is it possible to cooperate if each peer has a different perception of the LoS under which some function should be performed?
  - Yes, but with some trade-offs
  - It is harder to predict how a peer will behave

# Agreement on the LoS



Agreement on the LoS for cooperative functions

# Cooperative LoS evaluation

- **Why agreement on LoS?**
  - Cooperative driving function design assumes that all vehicles perform the function in the same LoS
  - Lower uncertainty implies better performance
- Allows vehicles to **agree on a common “Cooperative LoS”**
  - If agreement is reached in a **timely way**, then vehicles can rely on the cooperative LoS
  - Otherwise, all vehicles will implicitly agree to perform the function in the lowest LoS (without cooperating)

# Cooperative LoS evaluation

- The **Cooperative LoS** is evaluated based on all the Local LoS values proposed by vehicles
- A **fault tolerant consensus protocol** is executed to agree on the cooperative LoS
- The decision is taken as follows:

**Cooperative LoS = min (all received Local LoS)**

- The result is sent to the Safety Manager
- The result **must be sent periodically**

# (Local) LoS evaluation

- The **Local LoS** is evaluated (by the **LoS evaluator component**) only based on locally generated information (validity and timeliness), which is compared to **safety rules**
- Safety rule example:

```
Local LoS :=
          LoS_2 if  validity_received_timely &&
                    validity > validity_threshold &&
                    cooperative_LoS_received_timely
          LoS_1 otherwise
```

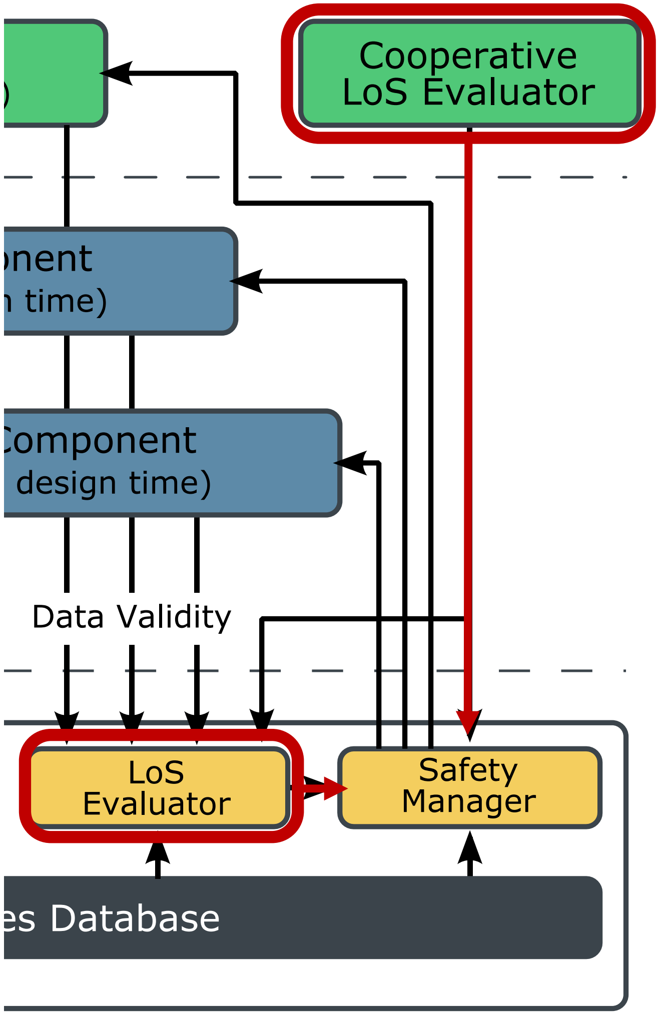
- The determined Local LoS is sent to the **cooperative LoS agreement** component and then forwarded to the other vehicles

# Effective LoS

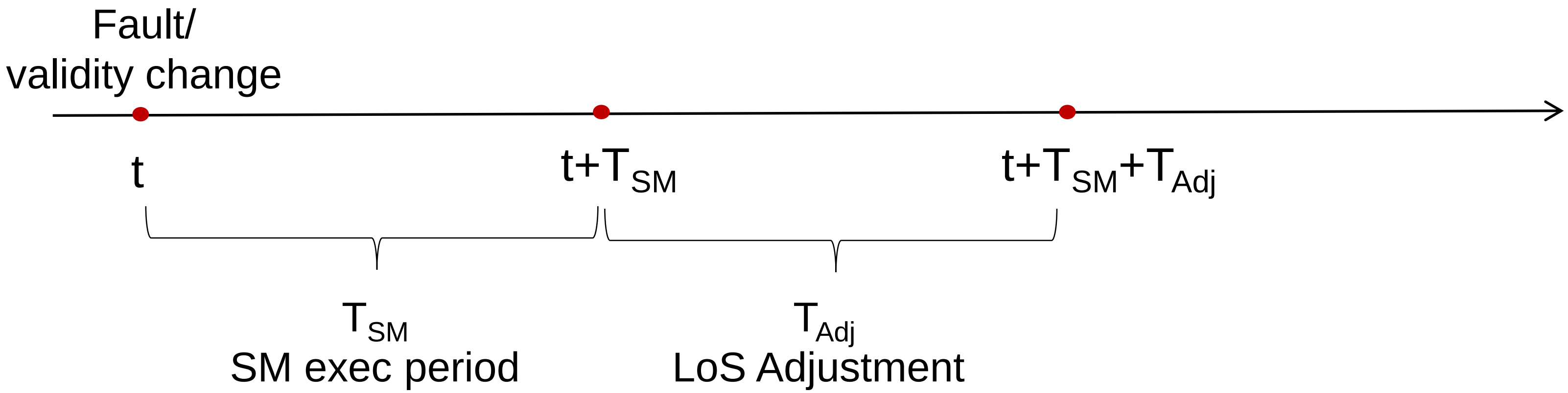
- Given the locally determined LoS, and the Cooperative LoS, it is possible to determine the LoS that must be effectively considered (for safety)
- It is calculated as:

**Effective LoS = min (Local LoS, Coop LoS)**

- The **Safety Manager** is in charge of informing functional components about the **Effective LoS**

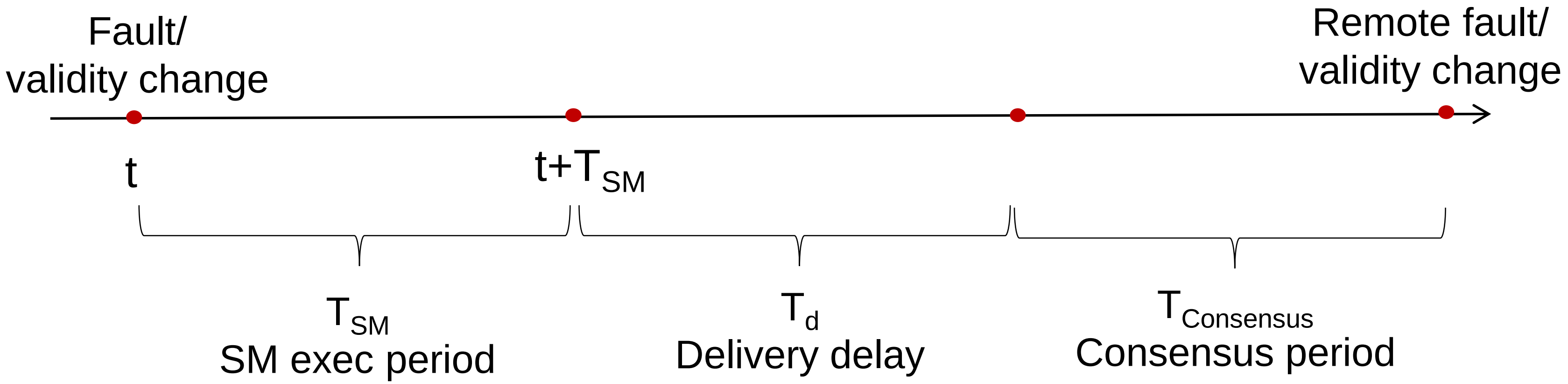


# Timeliness analysis (1)



- The Safety Manager executes periodically
- The LoS adjustment latency is always bounded, for all components below the hybridization line
- **LoS change due to local changes in integrity level** is always performed within **$t+T_{SM}+T_{Adj}$** from the fault occurrence

# Timeliness analysis (2)

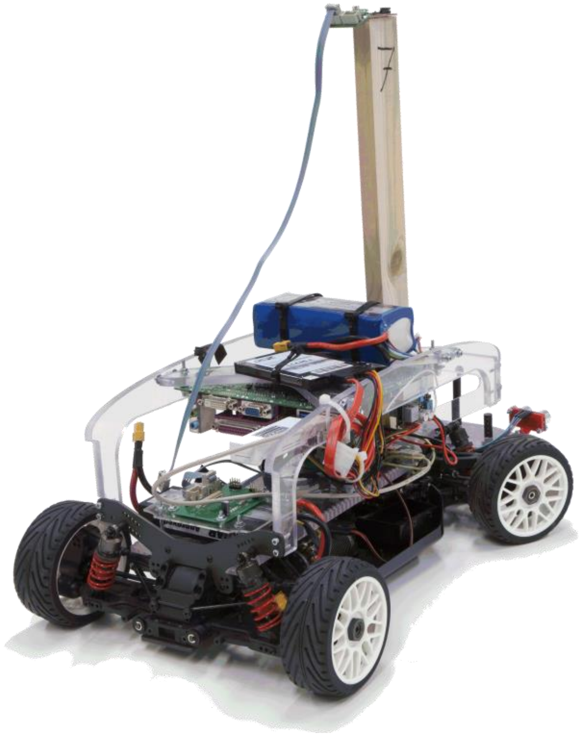


- It is possible to discard messages that take more that $T_d$ time units to be transmitted
- Every $T_{Consensus}$ time units, all Cooperative LoS Evaluators must compute a result
- The (lower) Local LoS **will be available at other vehicles at most by $t+T_{SM}+T_d+T_{Consensus}$**
- If the Cooperative LoS Evaluator is not timely, or if the message transmission is not timely, the other vehicles **will detect a timing failure of their Cooperative LoS Evaluator by time $t+T_{SM}+T_d+T_{Consensus}$**
- Therefore, on the other vehicles it will take an additional $T_{SM}+T_{Adj}$ to switch to the lower LoS

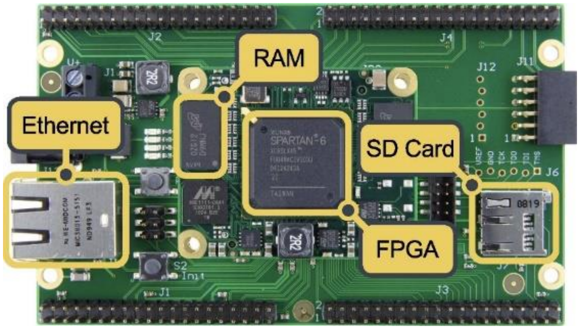
**LoS change bounded by time $2T_{SM}+T_d+T_{Consensus}+T_{Adj}$**

# Application

# KARYON (Gulliver) vehicle



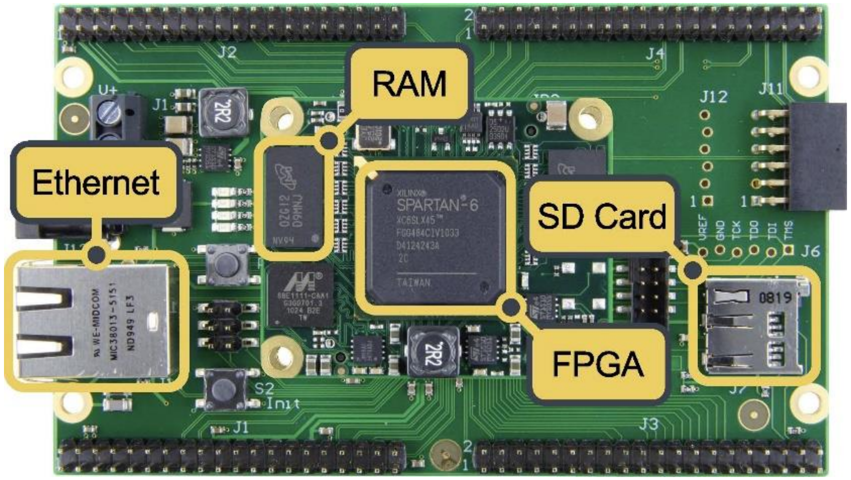
+



Autonomous and cooperative vehicle with KARYON architecture

# Safety kernel implementation

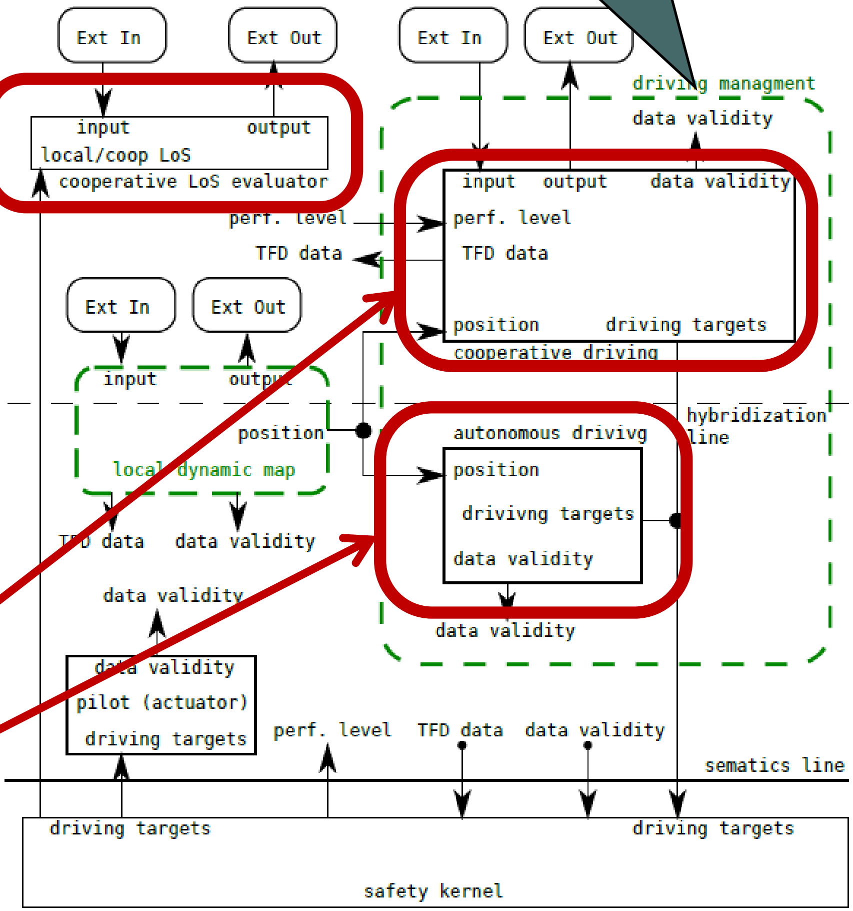
- FPGA-based development board
- Processing unit: LEON3 soft-processor (SPARC v8 arch)
- RTEMS executing on top
- Ethernet interface with payload system



# Vehicular application

- Gulliver test-bed

- **Two levels of service**:
  - Cooperative driving
  - Autonomous driving



Cooperative LoS evaluator

Driving Management:
- Cooperative
- Autonomous

# KARYON

## Automotive domain video

# Thank you for your attention!

To reach me: **casim@ciencias.ulisboa.pt**

Web page: **http://www.di.fc.ul.pt/~casim**

Ciências
ULisboa

LASIGE