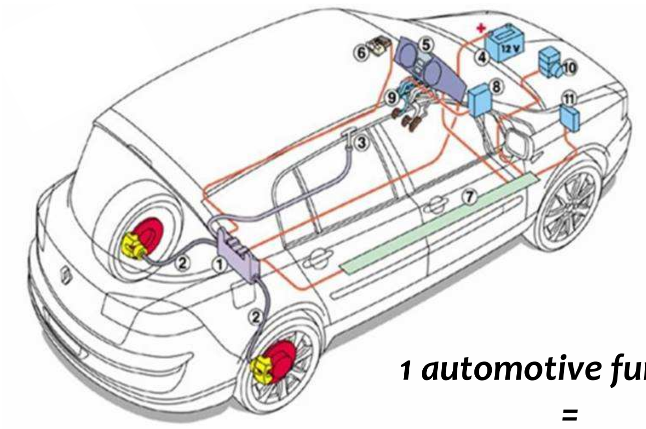
# An approach for improving Fault-Tolerance in Automotive Modular Embedded Software

**Caroline Lu,** **RENAULT SAS,** Guyancourt, France
**Jean-Charles Fabre, Marc-Olivier Killijian,** **LAAS-CNRS**, Toulouse, France

**RTNS'09**
**Paris, October 26-27, 2009**

# Safety requirements on Automotive applications




*1 automotive function*
*=*
*1 mechatronic system*

**Fault model on Embedded SW in 1 Electronic Control Unit (ECU)**

- **Physical Faults** : HW, Environment
- **Software Faults** : Design, Coding

- **ISO26262 Standard**
  - « Road Vehicle, Functional Safety »
  - Criticality levels : ASIL A-D
  - Requirements and Recommandations of safety mechanisms

- **Car-maker constraints**
  - New safety-critical functions

    Ex: active safety systems
  - USE: Unwanted System Events

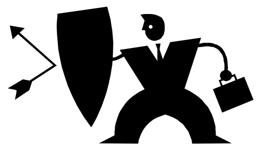
    Ex: wrong sequence order for switching on or off vehicle lamps

  → improve fault tolerance

# An approach for improving Fault-Tolerance in Automotive Modular Embedded Software

- **Automotive software context**

- **Framework overview**
- **Defense software**



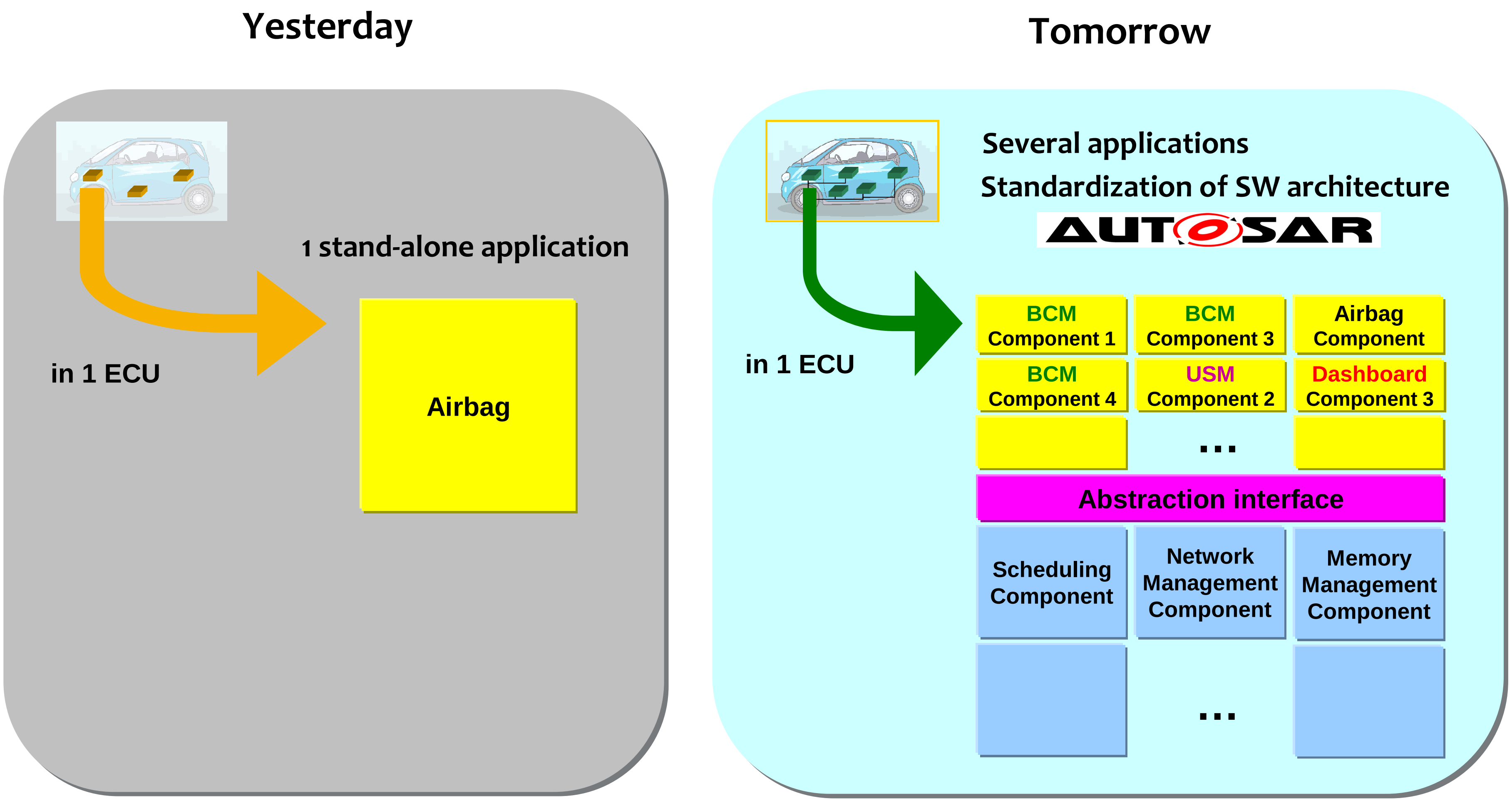
- **Instrumentation**

# Automotive Software Context

RTNS'09
Paris, October 26-27, 2009

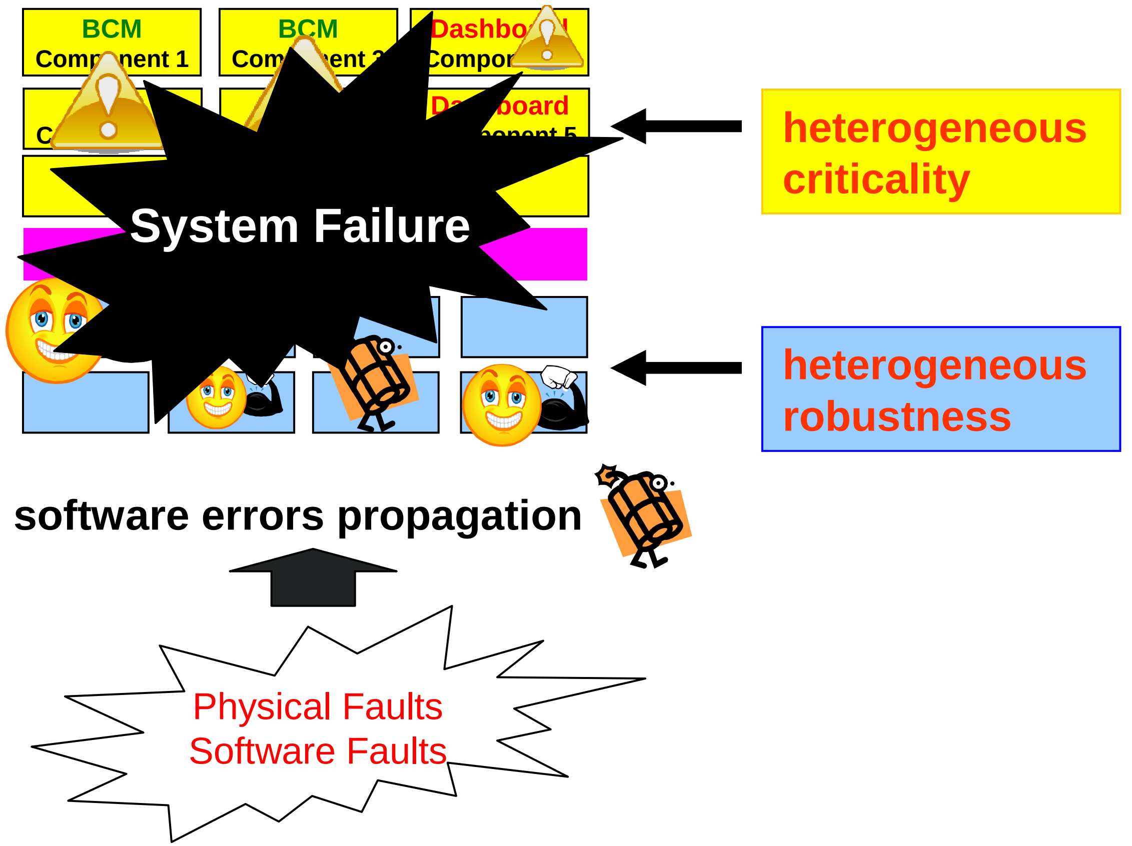
RENAULT

# Automotive SW architecture evolution



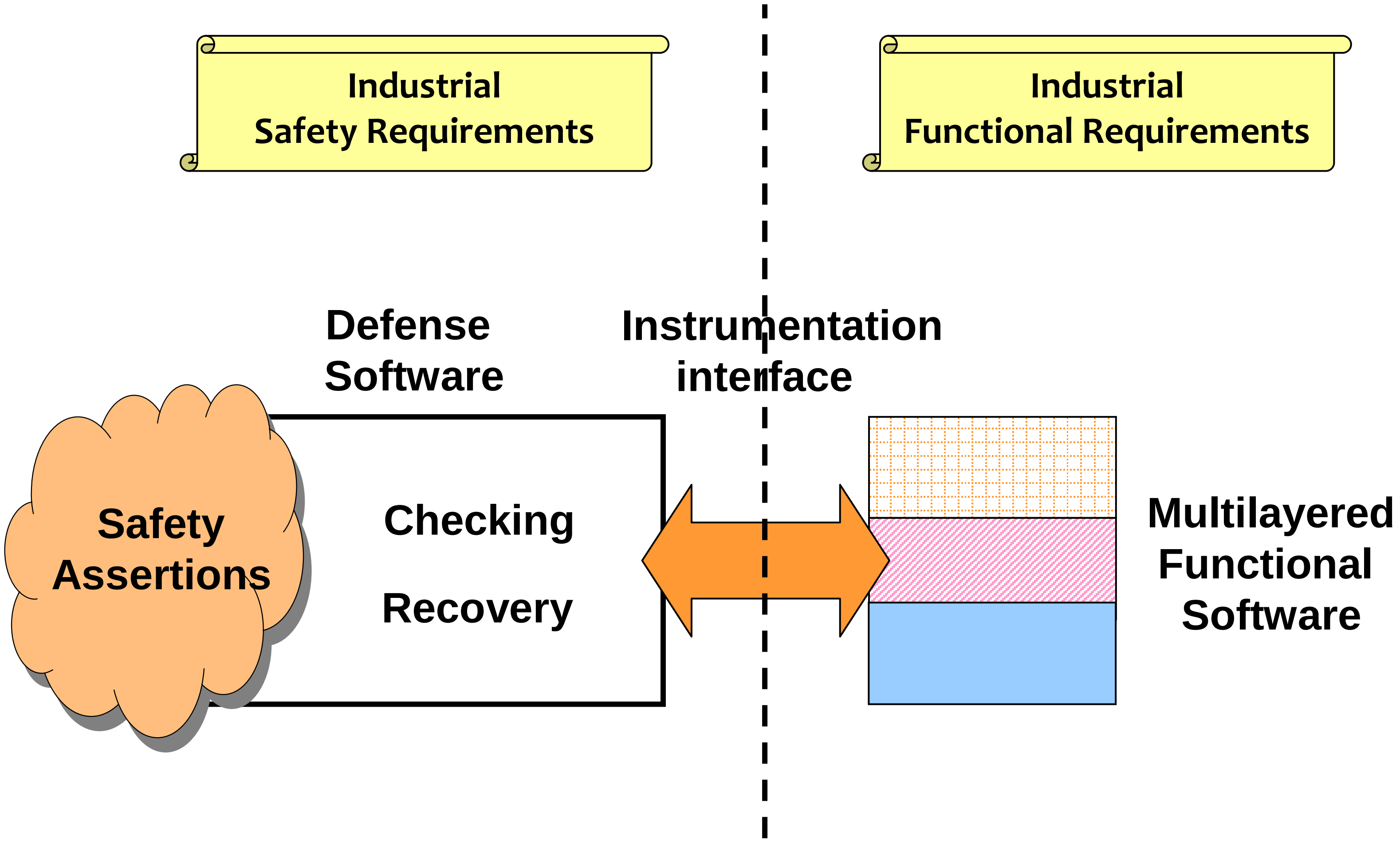
# Problem statement

➔ **Control error propagation between components through layers**

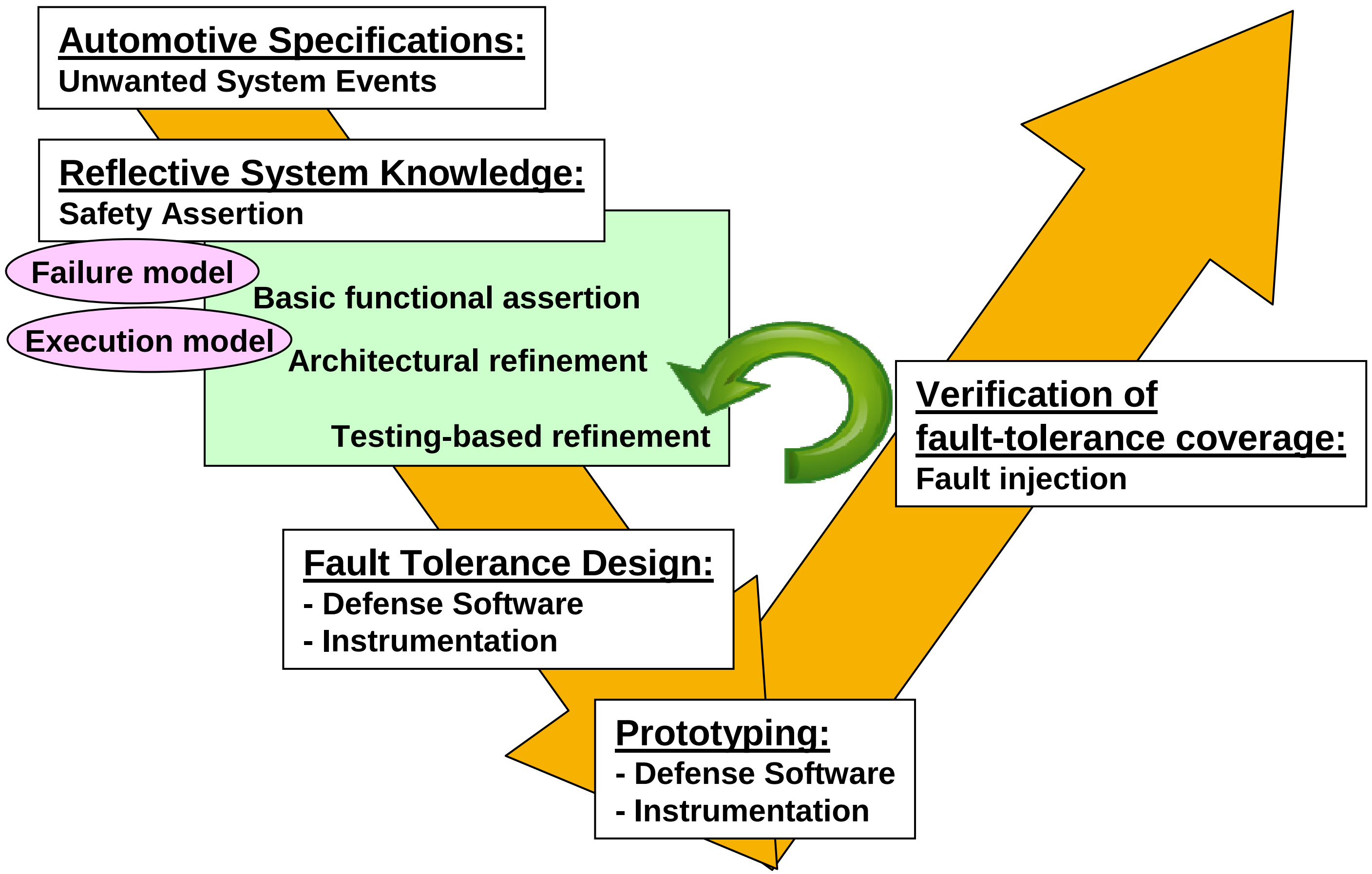


# Framework Overview

# Reflective Principle



# Development process of Defense mechanisms



# Failure model

- **Critical Control Flow Failures**
  - Control events of activation/termination of a treatment
  - Execution sequence of treatments
  - Execution time of a treatment

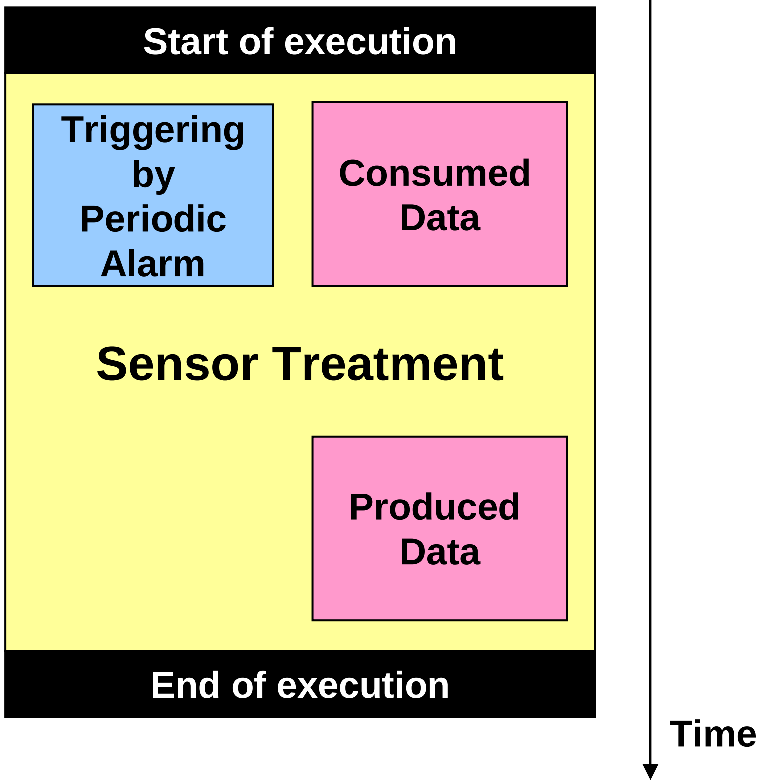
- **Critical Data Flow Failures**
  - Value of data
  - Time of data exchange

**(The classification is not orthogonal)**

# Execution model

**Scheduled Entity**

Ex: a sensor treatment




- **Sequence order of execution**
- **Timing characteristics**
  - Start/End of execution
  - Execution time
- **Exchanged Data**
  (Ex: global variable, message)
- **Control Events for activation/termination**
  (Ex: alarm, explicit activation service)

# From Unwanted System Event to assertions

**USE for hybrid transmission module:**

***The system is blocked (more than 1 second) in mode A, while the engine status is equal to 2, whereas it should switch to mode B***

**Assertion (checked periodically every second):**

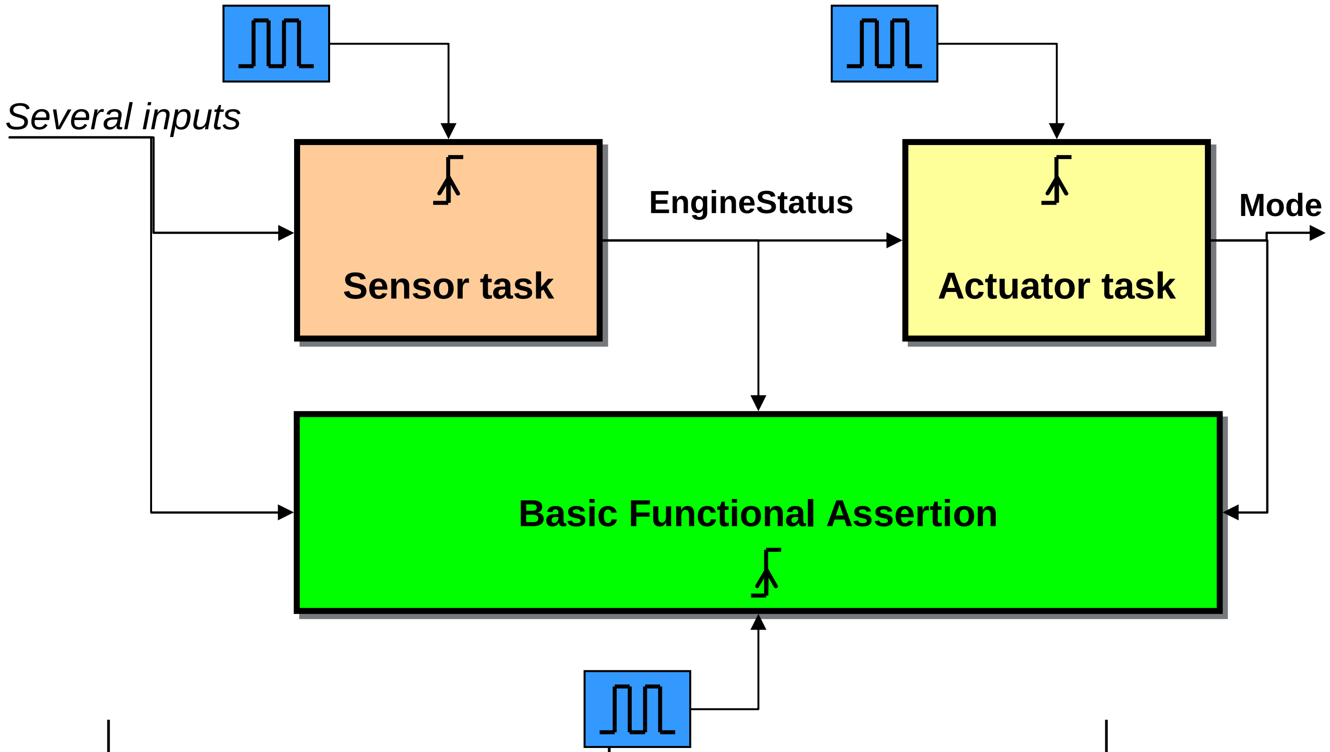
***The "Mode" variable output of the actuator task is consistent with the value of the sensor task inputs, while the "EngineStatus" variable is equal to 2***

**Implementation:**

Scheduled entities?
- Sensor task
- Actuator task

Exchanged data?
- Several inputs
- EngineStatus
- Mode

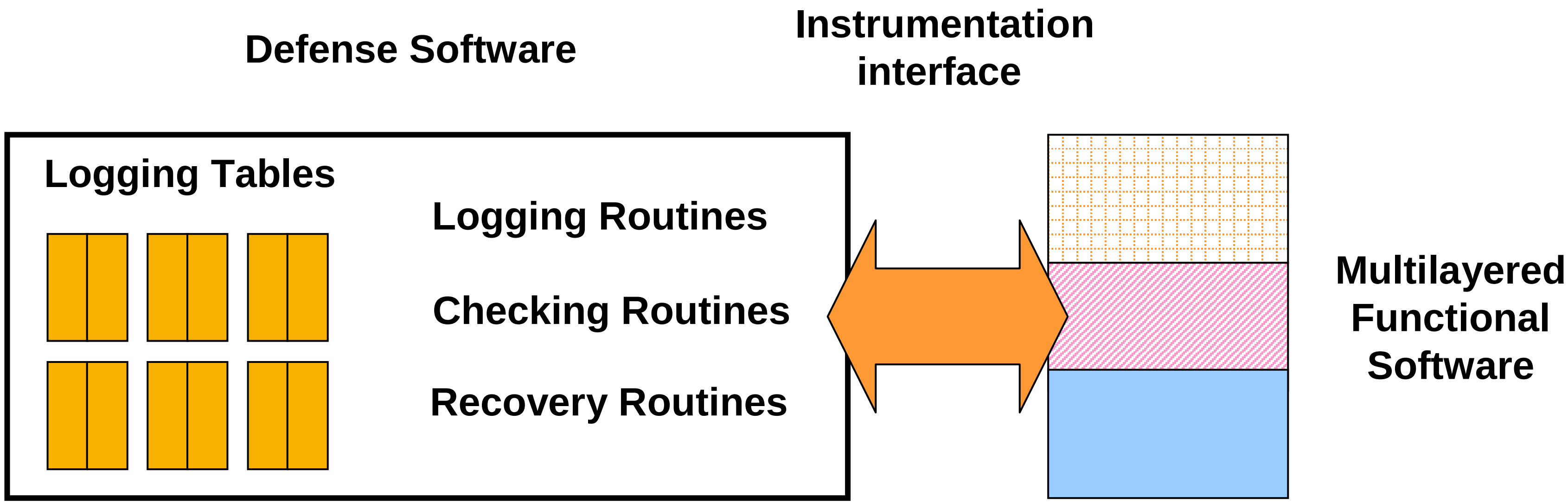
Control events?
none



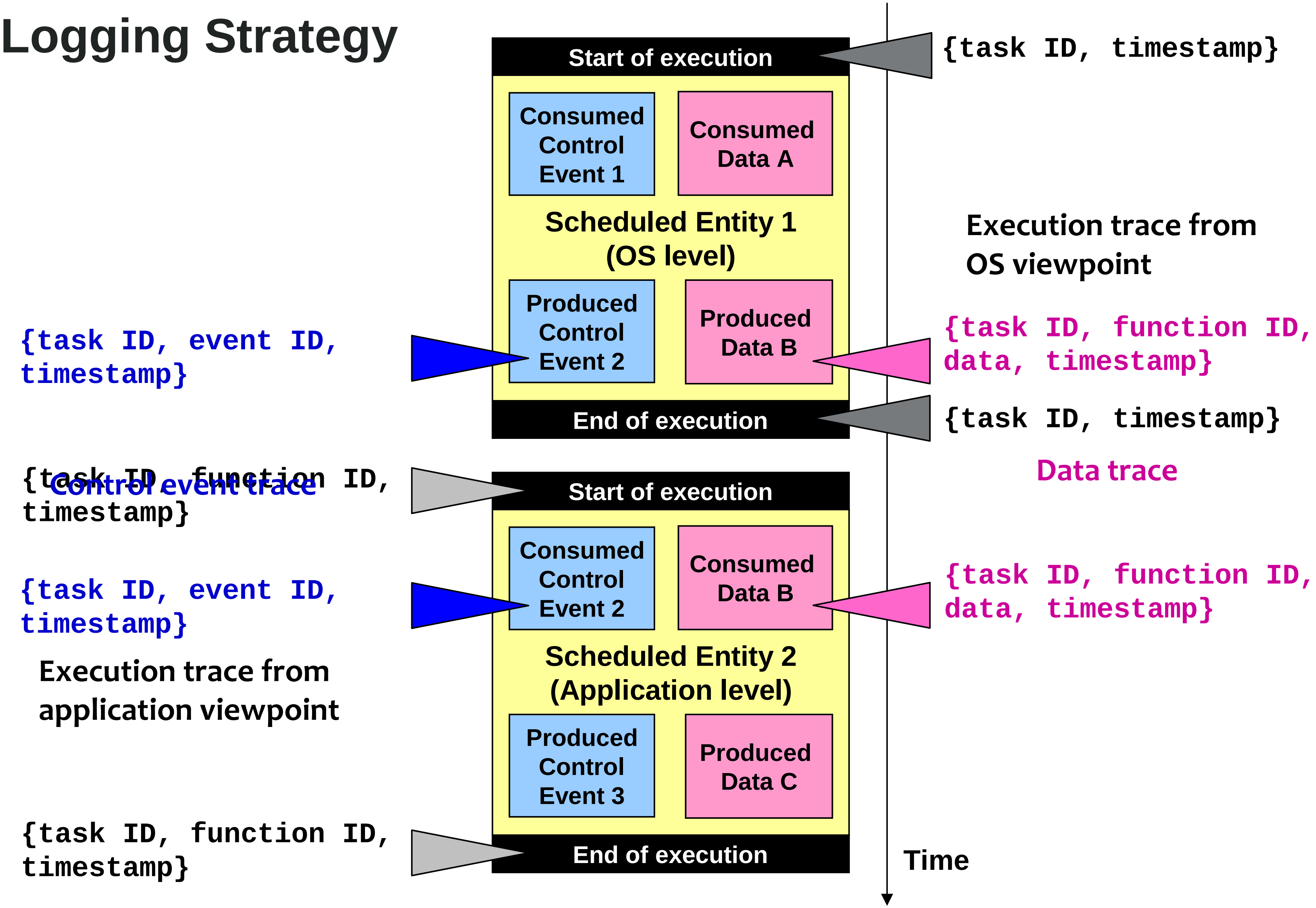
# Defense Software

RENAULT eco²

# Defense Software Organization



# Logging Strategy



# Detection Strategy

- **Logging tables as reference**

| Assertion with: | Logging tables |
|---|---|
| Control event | Control event trace |
| Sequence of execution | Execution trace |
| Timing constraints of execution | Execution trace |
| Value constraints on data | Data event trace |
| Timing constraints on data | Data event trace |

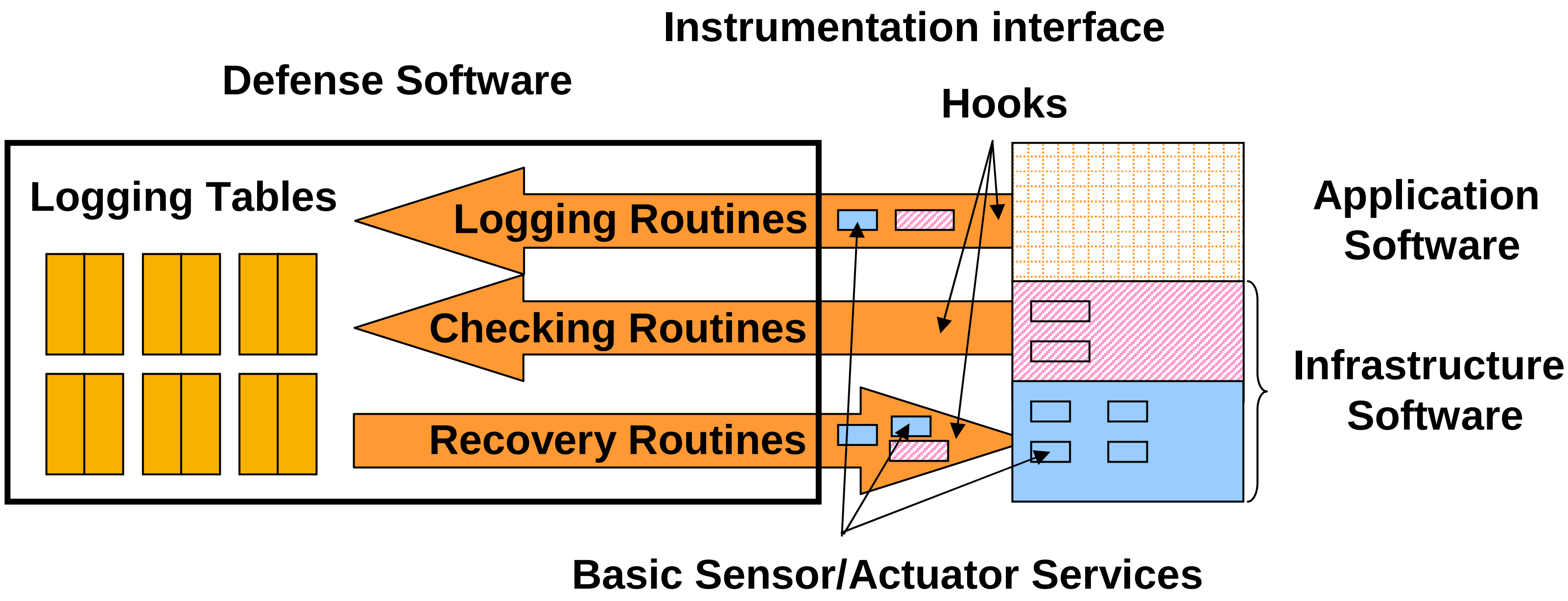
- **Detection routine**
  - Verification of assertions
  - Triggering of recovery treatments

# Recovery Strategy

- **Knowledge of application level**
  - Graceful degradation (*ex: if a critical sensor data is wrong then switch to another mode that computes other sensor data*)
- **Control of low-level services**
  - To correct control flow (*ex: kill a task and activate another one*)
  - To correct data flow (*ex: inhibit a wrong message and transmit the right one*)

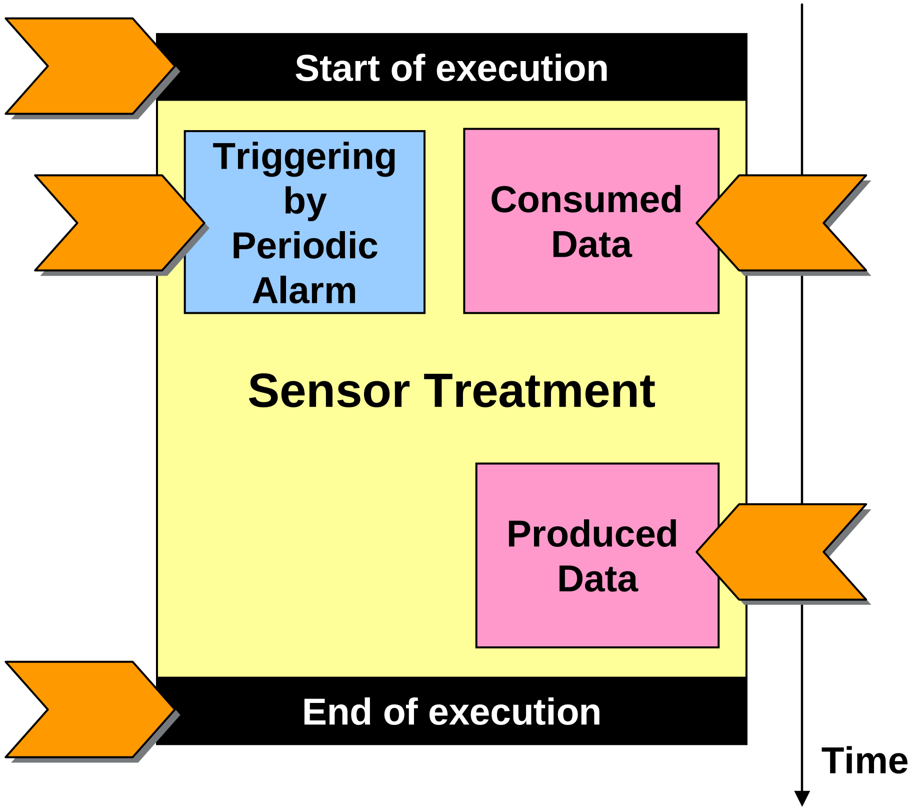
# Instrumentation

# Instrumentation



# Hooks

- **Where?**
  - On the control flow
  - On the data flow
- **When?**
  - Depending on information to logg
  - Depending on verifications to trigger
- **How?**
  - Existing hooks
  - Added hooks



# SW sensors/actuators

## Examples

- **OS services (OSEK OS) for Control flow**
  - Sensors: GetTaskID(), GetAlarm()…
  - Actuators: ActivateTask(), ChainTask(),…

- **Middleware services (AUTOSAR RTE) for Data flow**
  - Sensors: Rte_Read(),…
  - Actuators: Rte_Write(),…

# Conclusion

# Conclusions (1/2)

- **New trend in automotive systems:**
  - Multilayered software architecture
  - Use of Off-The-Shelf SW components
  - Emerging standards AUTOSAR, ISO26262

- **Dependability issues and safety concerns are of prime importance in this context**

- **Our approach:**
  - A reflective framework for fault-tolerance
  - A customizable defense software based on automotive safety requirements (USE)
  - A complete development methodology of the defense software

# Conclusions (2/2)

- **Early implementation and case studies on AUTOSAR SW platforms carried out for proof of concept**

- **Part of the demonstrator of the SCARLET project**

# Questions