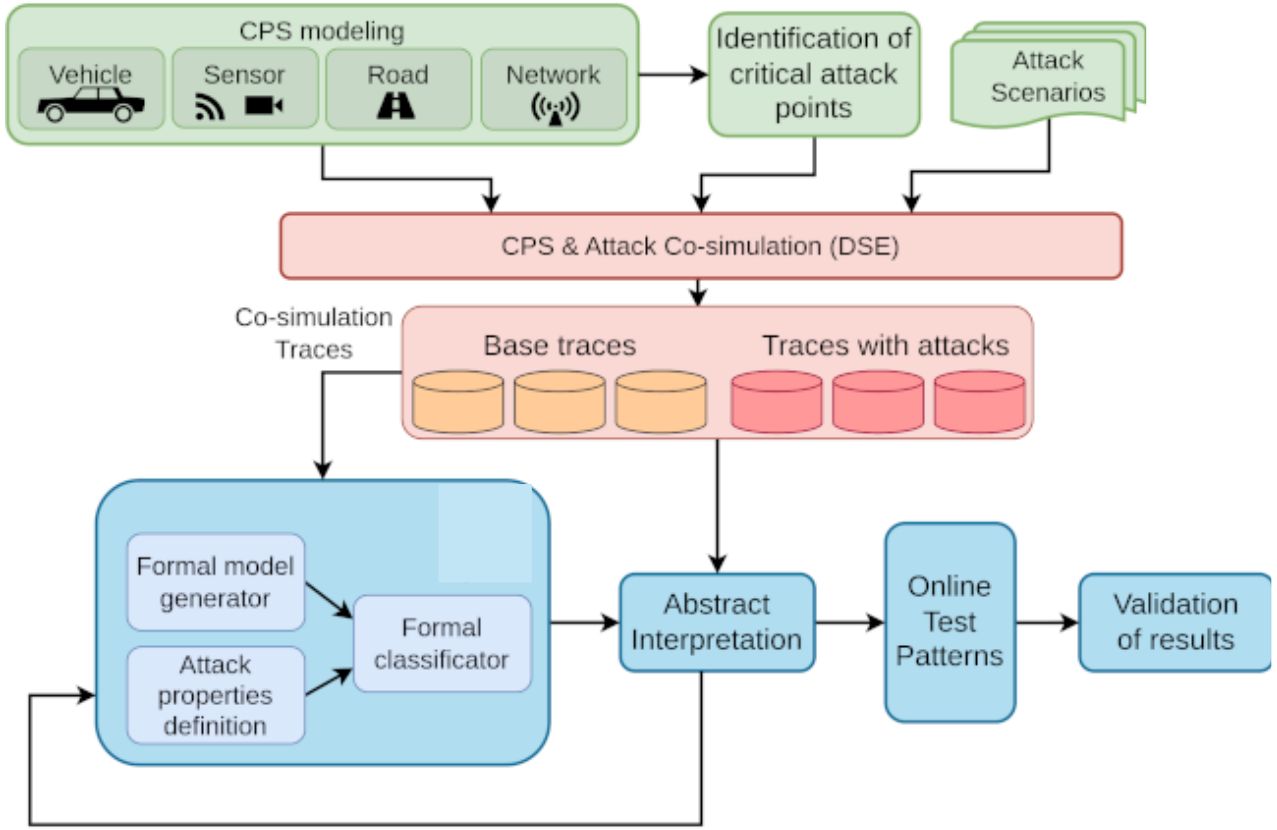
Project FORESEEN: FORmal mEthodS for attack dEtEction in autonomous driviNg systems, PRIN- PNRR 2022, Nov 2023 - Nov 2025.

Development of a methodology with supported tools for the detection of attacks in autonomous driving systems. The platoon case study.

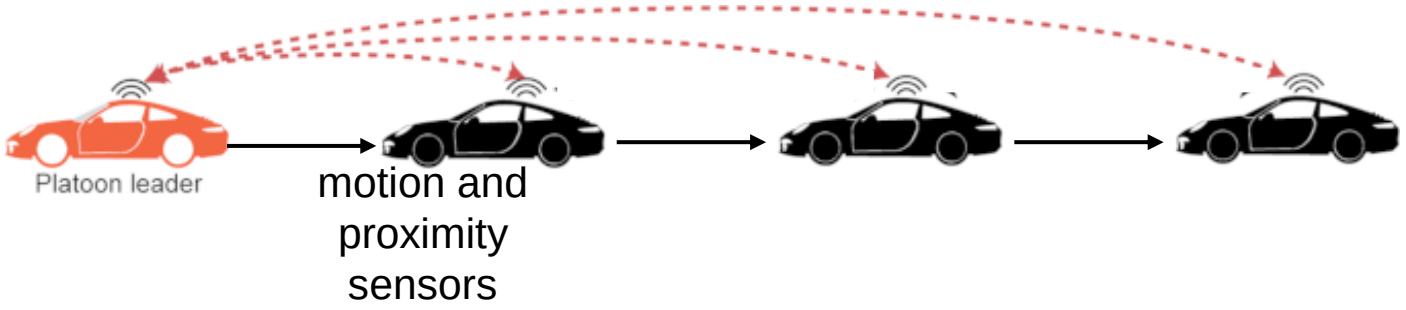


A basic platoon system

The system consists of a platoon of vehicles with one leader car and three following cars, these reach a steady state in which they mantain a constant spacing of 15 meters and the same speed.

Distributed control: **Cooperative Adaptive Cruise Control** (**CACC**) algorithm

Ideal communications

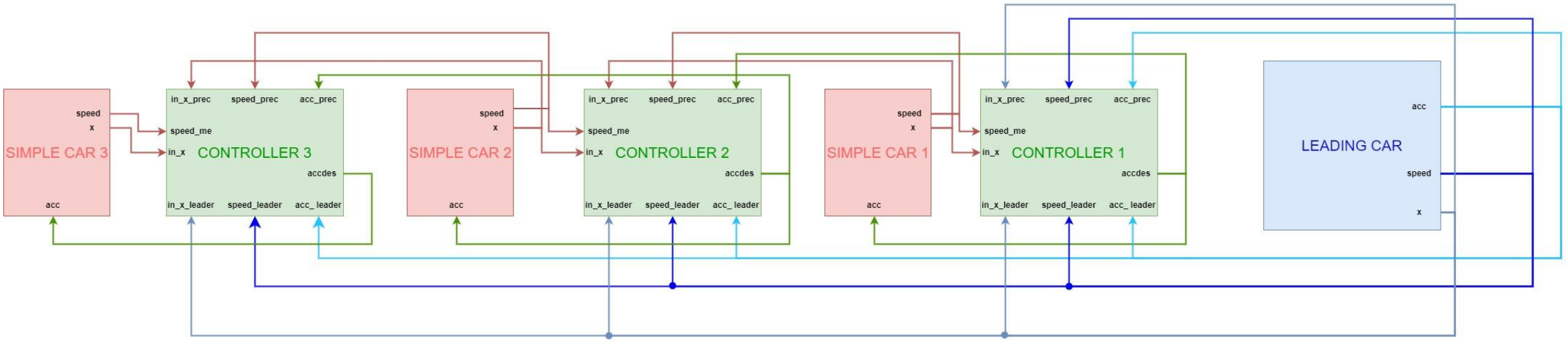


Leader car: modelled by its accelleration function

Following car: modelled by plant + local controller

- reads **acc.**, **speed** and **position** of the preceeding car by on-board sensors
- receives the **acc., speed** and **position of the leader** from the leader car

FMI-cosimulation schema



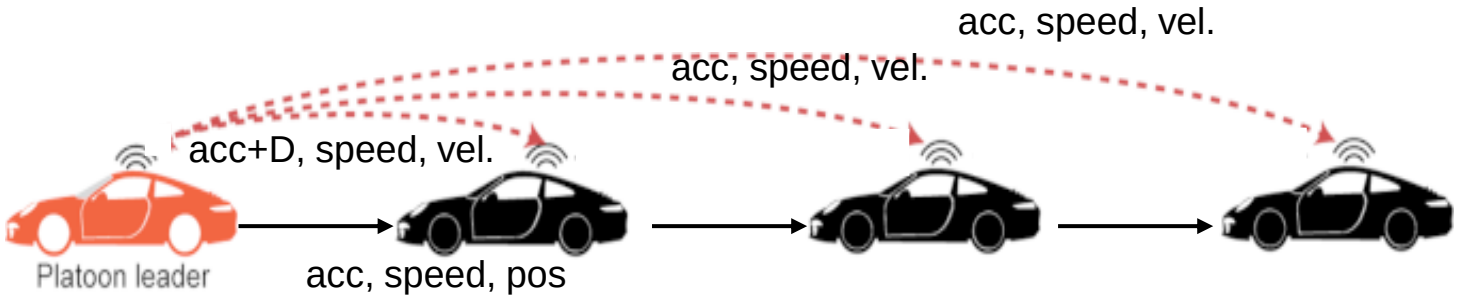
ego: following car
front: preceeding car

The **Adaptive Cruise Control algorithm** uses the formula below to calculate the acceleration needed to lead the system to steady state.

$$a_{ego} = C_1 a_{leader} + (1 - C_1)\, a_{front} - K_1\left(V_{ego} - V_{leader}\right) - K_2(x_{ego} - x_{front} + L)$$

C_1, K_1, K_2 are parameters of the system, while L is the lenght of cars

Platoon under attack: fmiSwap



Simple case: Leader has been compromised.

Leader sends acc. incremented by a positive constant D to simple car1, after the interval time T1

Leader sends correct acc. to the other cars

- We used file .xml to inject the attack in the simple system.

- We define **Controller1_mitigation** module.
  After the interval of time T1+T2 , we swap **Controller1** with **Controller1_mitigation**

Controller1_mitigation module. For CAR 1, the acc. received by the leader must be equal to the acc. read from sensors. If they differ, we trust local sensors, and the controller of car1 substitutes the acc. received from the leader with the acc. read from local sensors in the formula.

$$a_{car1} = C_1 a_{front} + (1 - C_1)\, a_{front} - K_1\left(V_{ego} - V_{lead}\right) - K_2(x_{ego} - x_{front} + L)$$

Questions

- Is it possible to quickly find the identifier of a variable inside of a multi model, in order to properly use the .xml fault injection file?

```
<events>
    <event id="id-A" when="t&gt;=12.0" other="(var_17&gt;1.6) &amp; (var_16=0.0)" vars="var_17,var_16">
        <variable valRef="16" type="real" newVal="1.0-var_16" vars="var_16," />
    </event>
    <!-- <event id="id-A" when="t&gt;=0.5" other="(var_17&gt;1.5) &amp; (var_16=0.0)" vars="var_17,var_16" > -->
    <!--     <variable valRef="17" type="real" newVal="-0.001*var_17+1.5" vars="var_17," /> -->
    <!-- </event> -->
</events>
```

```
▼<ScalarVariable name="maxlevel" valueReference="0" description="the max tank level" causality="parameter" variability="fixed" initial="exact">
    <Real start="2.0"/>
  </ScalarVariable>
▼<ScalarVariable name="minlevel" valueReference="1" description="the min tank level" causality="parameter" variability="fixed" initial="exact">
    <Real start="1.0"/>
  </ScalarVariable>
▼<ScalarVariable name="level" valueReference="3" description="the tank level" causality="input" variability="continuous">
    <Real start="1"/>
  </ScalarVariable>
▼<ScalarVariable name="valve" valueReference="4" description="the tank valve state" causality="output" variability="discrete" initial="calculated">
    <Boolean/>
  </ScalarVariable>
</ModelVariables>
▼<ModelStructure>
  ▼<Outputs>
    <Unknown index="4" dependencies=""/>
  </Outputs>
</ModelStructure>
```

Questions

- Is it possible to quickly find the identifier of a variable inside of a multi model, in order to properly use the .xml fault injection file?

In the previous slide, the first image refers to the wt_fault.xml file, where the variable var_17 for example refers to the water level detected in the tank controller, instead var_16 refers to the valve being open or closed.

The second image instead refers to the ModelDescription.xml of the water tank controller, where it can be seen that those two variables have completely different indexes.

Each FMU, within the model description file, will assign an index to determine the specific value inside of the FMU, but Maestro will assign, to each variable of each FMU, a new index.

We would like to know how to quickly fin such index in a multi model that has 7 or more FMUs, or if there is any other way in which we can name said variables univocally without using an index number.

Questions

• How is it possible to specify the swap condition using variables such as time or the output of some FMU?

Here the swap condition and the step condition are set to (true).

How we should define a swap condition like, for example that is set to true as soon as the leak_detector detects a leak in the water tank?

Would it be like the following?

({X3}.leak_detector.leak > 0)

```
"modelSwaps": {
  "controller": {
    "swapInstance": "leak_controller",
    "stepCondition": "(true)",
    "swapCondition": "(true)",
    "swapConnections": {
      "{x4}.leak_controller.valve": [
        "{x2}.tank.valvecontrol",
        "{x3}.leak_detector.valve"
      ],
      "{x2}.tank.level": [
        "{x4}.leak_controller.level"
      ],
      "{x3}.leak_detector.leak": [
        "{x4}.leak_controller.leak"
      ]
    }
  },
  "modelTransfers": {
    "controller": "controller",
    "tank": "tank"
  }
}
```

Questions

• How to properly define the object ModelTranfers?

We can’t understand to what the first and second “controller” refers, and the same goes for both the “tank” occurencies Withing the **modelTransfers**.

Moreover, if we tried to add an entry to modelTransers, such as for example:
“leak_detector”: “leak_detector”
it makes everything crash, so we clearly don’t understand how it should be used.

```
"modelSwaps": {
  "controller": {
    "swapInstance": "leak_controller",
    "stepCondition": "(true)",
    "swapCondition": "(true)",
    "swapConnections": {
      "{x4}.leak_controller.valve": [
        "{x2}.tank.valvecontrol",
        "{x3}.leak_detector.valve"
      ],
      "{x2}.tank.level": [
        "{x4}.leak_controller.level"
      ],
      "{x3}.leak_detector.leak": [
        "{x4}.leak_controller.leak"
      ]
    }
  },
  "modelTransfers": {
    "controller": "controller",
    "tank": "tank"
  }
}
```

Questions

- Is the Maestro used for the artifact (2.3.0) the same that is used with the INTO-CPS application? Could it be possible to use the .xml file for a fault injection with the above mentioned Maestro version?

```
<events>
    <event id="id-A" when="t&gt;=12.0" other="(var_17&gt;1.6) &amp; (var_16=0.0)" vars="var_17,var_16">
        <variable valRef="16" type="real" newVal="1.0-var_16" vars="var_16," />
    </event>
    <!-- <event id="id-A" when="t&gt;=0.5" other="(var_17&gt;1.5) &amp; (var_16=0.0)" vars="var_17,var_16" > -->
    <!--     <variable valRef="17" type="real" newVal="-0.001*var_17+1.5" vars="var_17," /> -->
    <!-- </event> -->
</events>
```

We would like to use the same xml file, that you used for the fault injection, to inject attacks on our system, using the INTO-CPS Application, ***thus being able to run a large set of simulations with the DSE.***

This is crucial for the data gathering process in our project.

Questions

- Why does the default values for each parameter, set in the ModelDescription.xml within the FMU, have the priority with respect to the values specified in the mm.json configuration file?

```
"parameters": {
  "{SimpleCar1}.SimpleCar1.v0": 5,
  "{SimpleCar1}.SimpleCar1.x0": 25,
  "{Controller1}.Controller1.targetDistance": 15,
  "{Controller2}.Controller2.targetDistance": 15,
  "{SimpleCar2}.SimpleCar2.x0": 15,
  "{SimpleCar2}.SimpleCar2.v0": 5,
  "{Controller3}.Controller3.targetDistance": 15,
  "{SimpleCar3}.SimpleCar3.x0": 5,
  "{SimpleCar3}.SimpleCar3.v0": 5,
  "{LeadCar}.LeadCar.simple_car.x0": 40,
  "{LeadCar}.LeadCar.simple_car.v0": 10
```

```
<!--  Index of variable = "8"  -->
▼<ScalarVariable name="v0" valueReference="7" variability="fixed" causality="parameter">
  <Real start="10"/>
</ScalarVariable>
<!--  Index of variable = "9"  -->
▼<ScalarVariable name="x0" valueReference="8" variability="fixed" causality="parameter">
  <Real start="30"/>
</ScalarVariable>
```

In this example, the image on the left is taken from mm1.json, the one on the right is taken from the ModelDescription.xml file of the SimpleCar1. We would like its speed and position to be 5 and 25 but they are instead 10 and 30.

Questions

- When the transition from a multi model to another has been made, and when the swap occurs, how does the previously obtained state for each FMU get preserved? Does it get serialized and then unserialized inside the upcoming FMUs?

For example, taking a look at the water tank controller, after 22 simulation seconds there is a transition from mm1 to mm2, moreover, after the transition the swap and step conditions in mm2 are set to (true) right away, thus starting the swap as soon as the transition ends.

After the swap and transition, the new controller that has been swapped in (leak_controller) has an internal state that is coherent with the state of the basic controller that was running during the first stage of the overall cosimulation.

This is achieved by the use of fmi2Serialize and fmi2Deserialize functions?
Does this mean that when doing a transition and a swap one has to use FMUs that have the same internal structure expect for some new variables?

Questions

Moreover, during some testing, this error showed up doing the docker run. Any idea on how to solve it?

16:40:52.828 [main] ERROR org.intocps.maestro.MablSpecificationGenerator - Internal error in plug-in 'JacobianStepBuilder' at fixedStepSizeTransfer.
Message: Internal error: java.lang.RuntimeException: Expansion not possible type errors:
Error 0999: Internal error in plug-in 'Initializer' at initialize_transfer.
Message: Internal error: 'initialize_transfer' at 129:21
Error 0997: Unfold failure in plugin Initializer for initialize_transfer null
Error 0000: java.lang.NullPointerException null
Error 0999: Internal error in plug-in 'JacobianStepBuilder' at fixedStepSizeTransfer. Message: Internal error: 'fixedStepSizeTransfer' at 131:29
Error 0997: Unfold failure in plugin JacobianStepBuilder for fixedStepSizeTransfer null