

VOLKSWAGEN

AKTIENGESELLSCHAFT

KONZERNFORSCHUNG

EXTENSIONS OF THE XML FILE FOR THE HIERARCHICAL SYSTEM EDITOR

STATE OF THE ART: OPENPASS GUI

The screenshot displays the openPASS GUI interface, which is a complex system editor. It features several panels with adjustable parameters and algorithm configurations. The panels include:

- EgoSensor**: Parameters for ego vehicle sensing, including position, velocity, and acceleration.
- Init_Agent**: Parameters for the initial agent, such as trajectory, weight, and wheelbase.
- Sensor_Collision**: Parameters for collision sensing, including penetration time and collision occurrence.
- Algorithm_TrajectoryFollower**: Parameters for trajectory following, including tire coefficients, wheel angle limits, and brake/gas pedal gains.
- Algorithm_Selector**: A list of algorithms to be used, such as Driver Throttle Pedal, Driver Brake Pedal, Driver Steering Angle, Brake Assist, Lane Assist, and Evasive Steering Assist.
- Dynamics_TwoTrack**: Parameters for two-track dynamics, including tire radius, static and dynamic forces, slip, and engine torque.
- Dynamics_Collision**: Parameters for collision dynamics, including collision occurrence.

The GUI uses a hierarchical tree structure to organize these parameters and algorithms, with expandable sections and numerical input fields for each parameter.

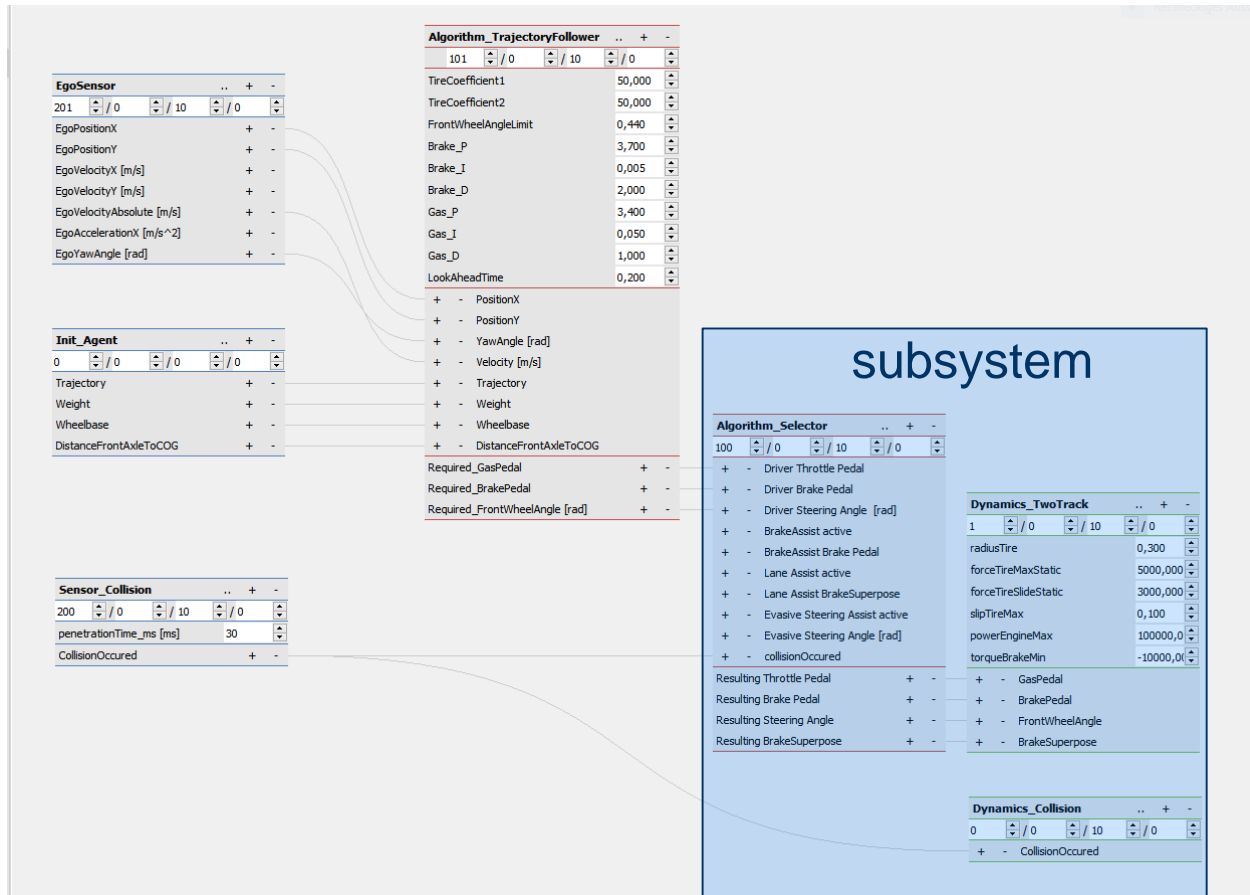
existing system editor allows for building several independent systems

problems:

1. user loses overview if many components are included
2. repeating constructions have to be copied



STATE OF THE ART: OPENPASS GUI



existing system editor allows for building several independent systems

problems:

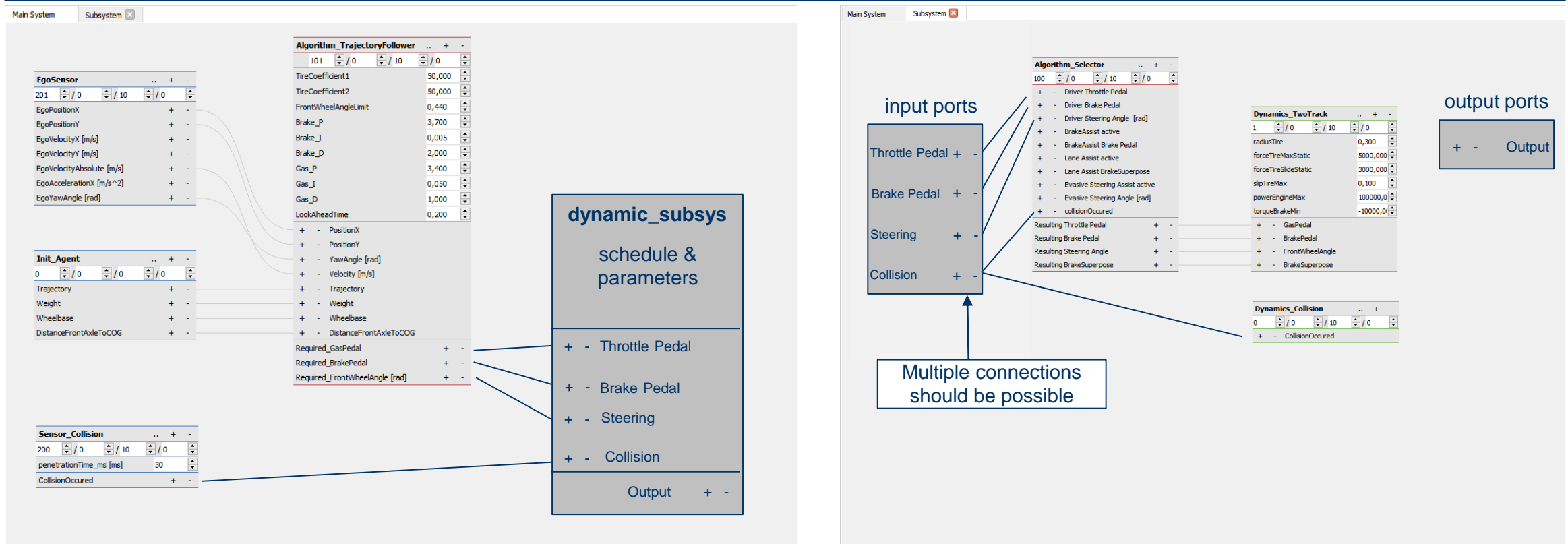
1. user loses overview if many components are included
2. repeating constructions have to be copied

idea: hierarchical system editor

grouping components and their connections to subsystems (e.g. as in Simulink)

PERSPECTIVE: HIERARCHICAL SYSTEM EDITOR

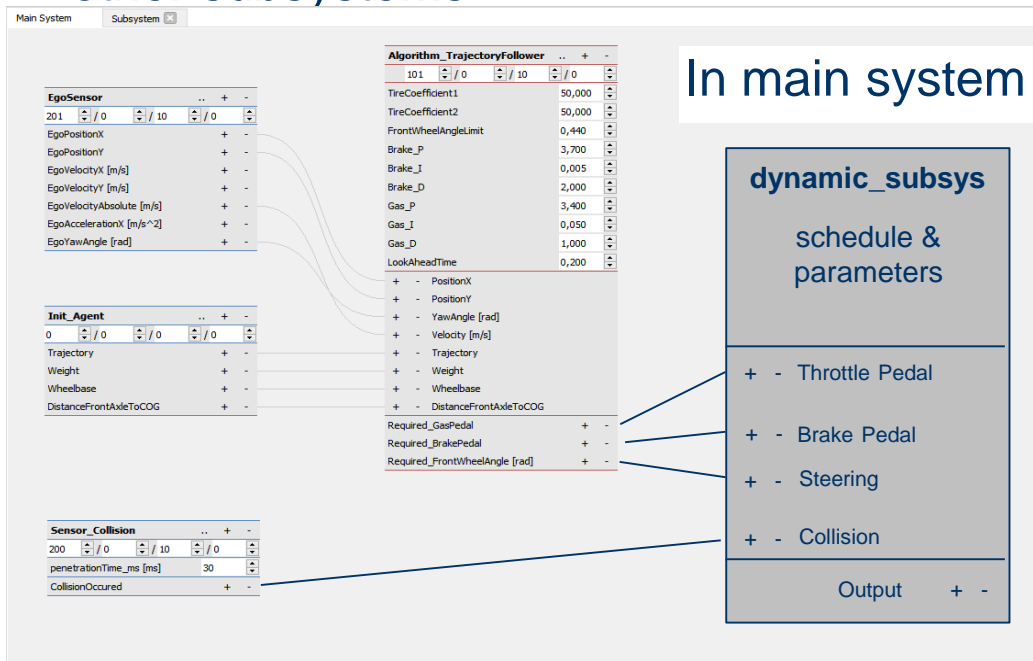
system XML file



NEW TYPE SUBSYSTEM: COMBINATION OF SYSTEM AND COMPONENT PROPERTIES

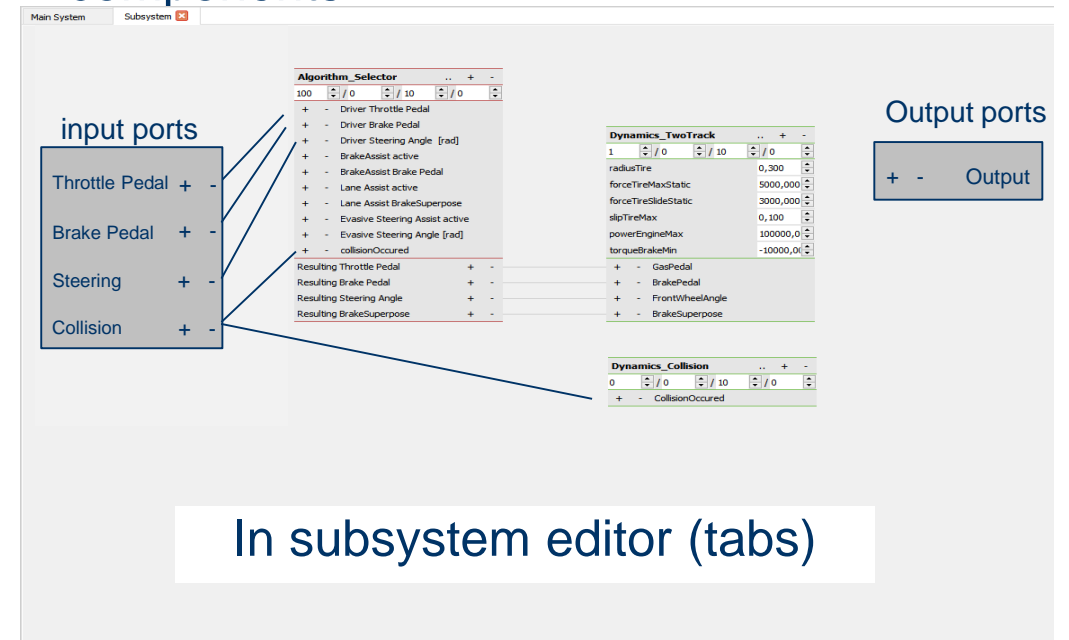
Subsystems behave as components:

- inputs, outputs, parameters and schedule
- they can be connected to components or other subsystems



Subsystems behave as systems:

- they consist of multiple components
- they consist of connections between components



NECESSARY EXTENSIONS OF THE XML FILE STRUCTURE

1. Embedding into the existing XML file

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<systems>  
  <system>  
  ...  
  </system>  
</systems>
```

today's system XML file is
contained and **not modified!**

```
<subsystems>  
  <subsystem>  
  ...  
  </subsystem>  
</subsystems>
```

global definition of
subsystems

2. Basic XML skeleton of a subsystem

```
<subsystem>
```

```
  <id/>
```

```
  <title/>
```

```
  <schedule/>
```

```
  <parameters/>
```

```
  <inputs/>
```

```
  <outputs/>
```

```
  <components/>
```

```
  <connections/>
```

```
</subsystem>
```

Inherited from type **component**

Inherited from type **system**



NECESSARY EXTENSIONS OF THE XML FILE STRUCTURE

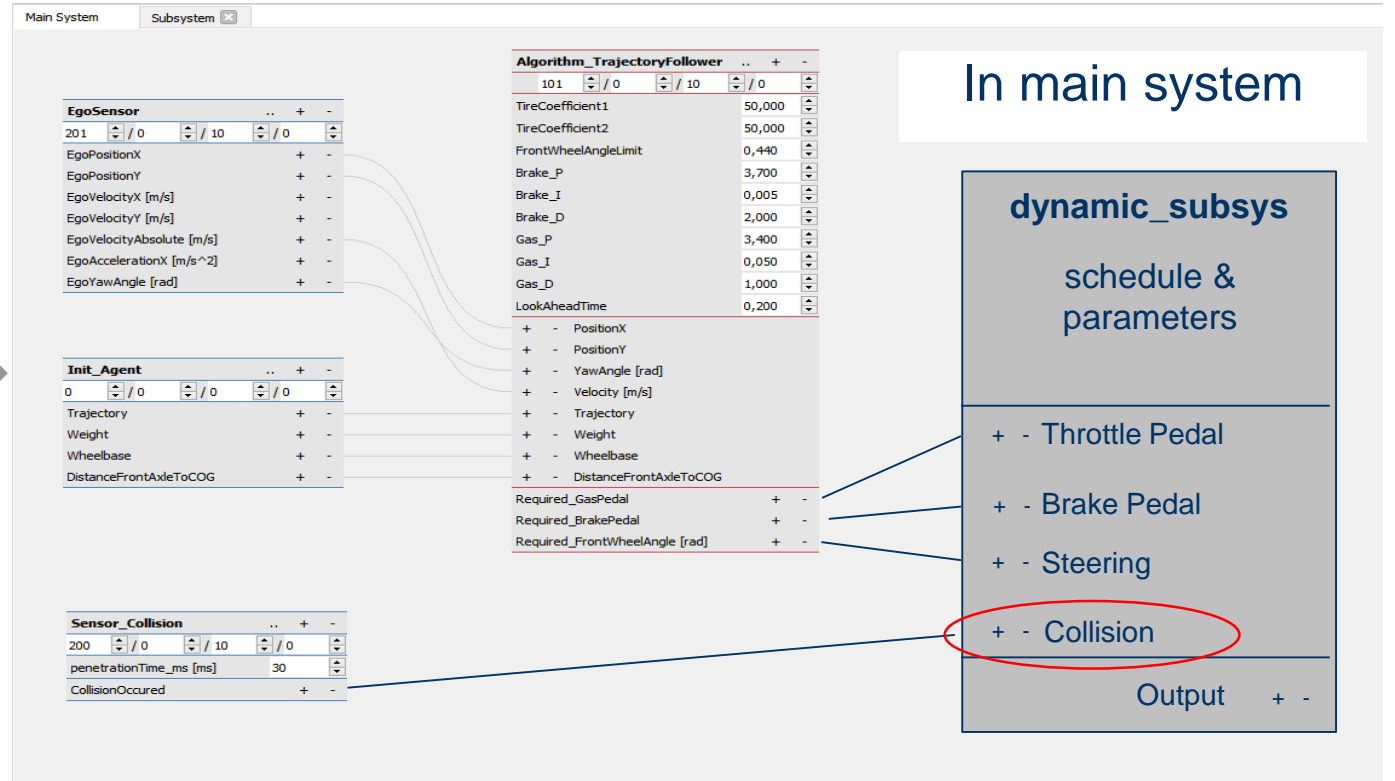
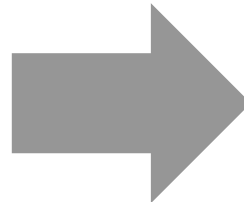
3. Definition of the subsystem as a component in the main system

```

<systems>
  <system>
    <components>
      ....
      <component>
        <id> 4 </id>
        <library> $ subsystem:0$ </library>
        <title> dynamic_subsys </title>
        <schedule>
          ...
        </schedule>
        <parameters/>
      </component>
    </components>
  </system>
</systems>

```

definition of an subsystem as a component



NECESSARY EXTENSIONS OF THE XML FILE STRUCTURE

3. Definition of input, output and parameter items as in component XML file

```
<subsystems>
```

```
  <subsystem>
```

```
    <inputs>
```

```
      ....
```

```
      <input>
```

```
        <id> 3</id>
```

```
        <title> Collision </title>
```

```
        <type> bool </type>
```

```
        <unit> </unit>
```

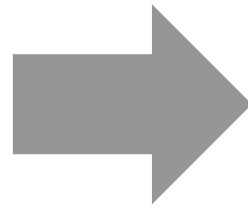
```
      </input>
```

```
    </inputs>
```

```
  </subsystem>
```

```
</subsystems>
```

definition of an input



In main system

dynamic_subsys

schedule & parameters

- + - Throttle Pedal
- + - Brake Pedal
- + - Steering
- + - Collision

Output + -

The screenshot shows a 'Main System' window with several subsystems: 'EgoSensor', 'Init_Agent', 'Sensor_Collision', and 'Algorithm_TrajectoryFollower'. Each subsystem has a list of parameters with numerical values and units. Lines connect these parameters to the 'dynamic_subsys' box on the right, indicating data flow. The 'Collision' output is highlighted with a red circle.



NECESSARY EXTENSIONS OF THE XML FILE STRUCTURE

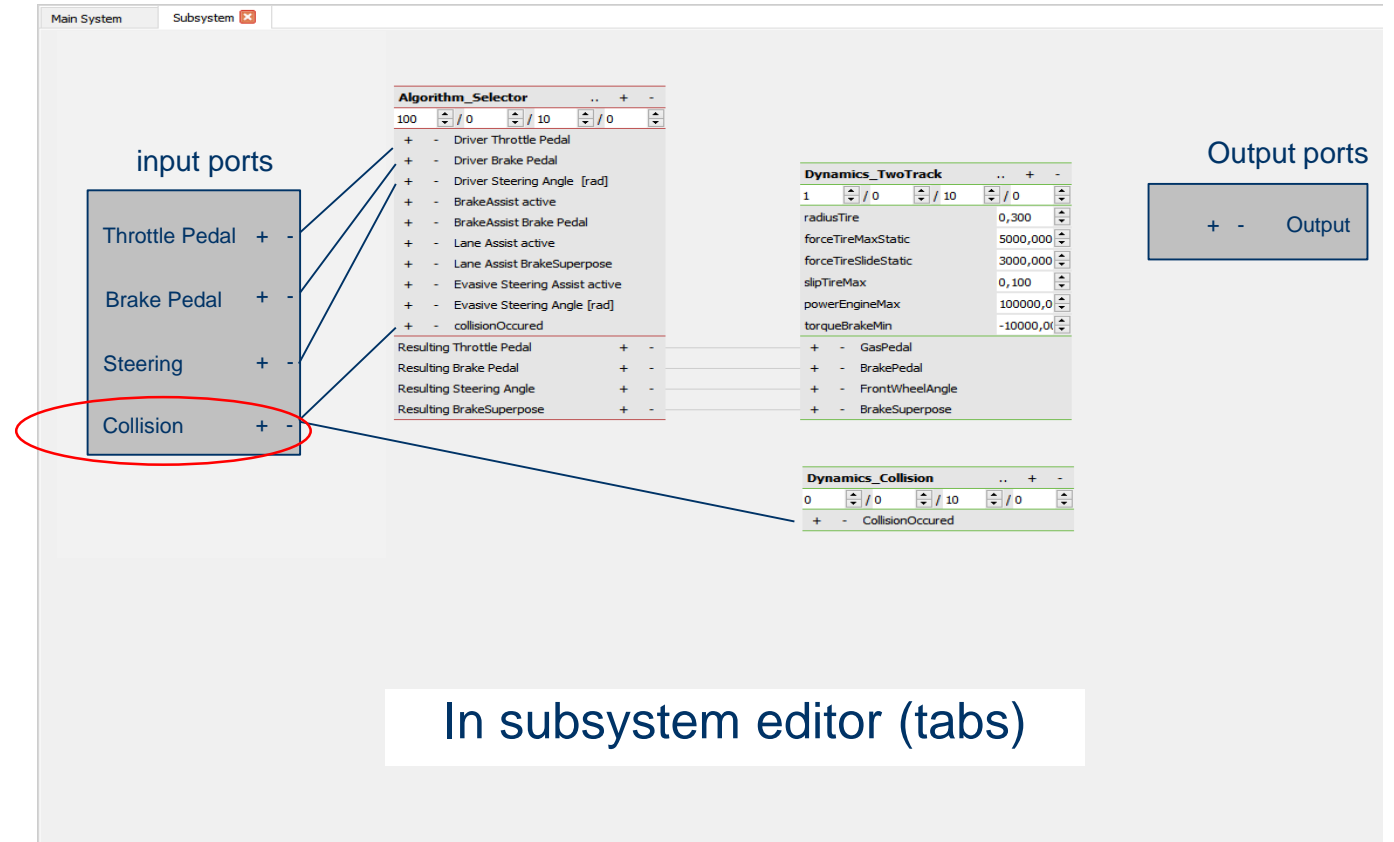
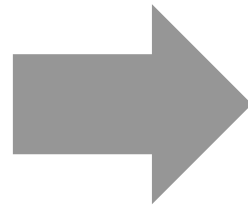
4. Definition of input, output and parameter items as in component XML file

```

<subsystems>
  <subsystem>
    <inputs>
      ....
      <input>
        <id> 3</id>
        <title> Collision </title>
        <type> bool </type>
        <unit> </unit>
      </input>
    </inputs>
  </subsystem>
</subsystems>

```

definition of an input



NECESSARY EXTENSIONS OF THE XML FILE STRUCTURE

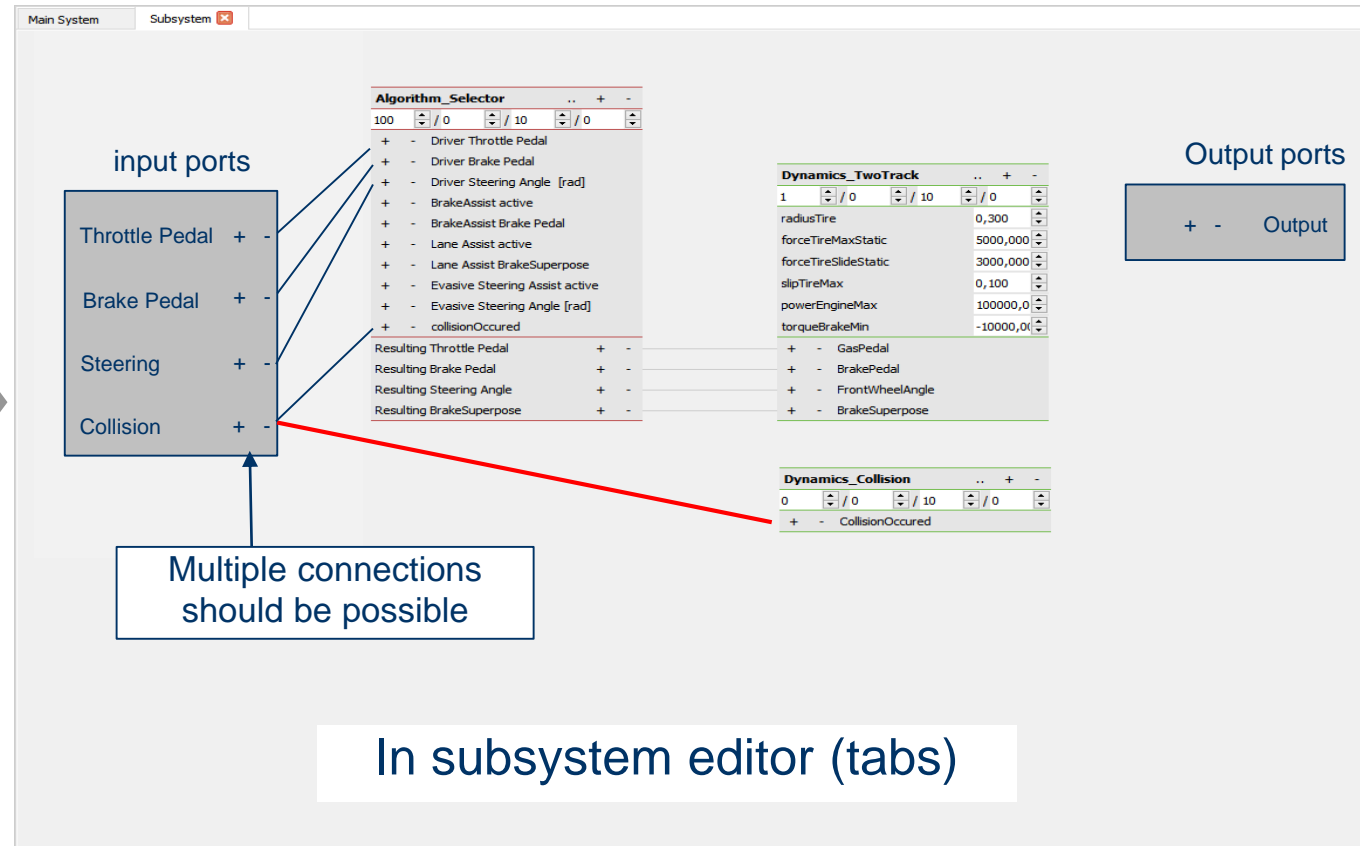
5. Connections between inputs/outputs and components similar to system XML file

```

<subsystems> <subsystem>
  <inputconnections>
    <inputconnection>
      <id>0</id>
      <source> 3 </source> <!-- input ID-->
      <target>
        <component> 3 </component>
        <input> 0 </input>
      </target>
    </inputconnection>
    ...
  </inputconnections>
</subsystem> </subsystems>

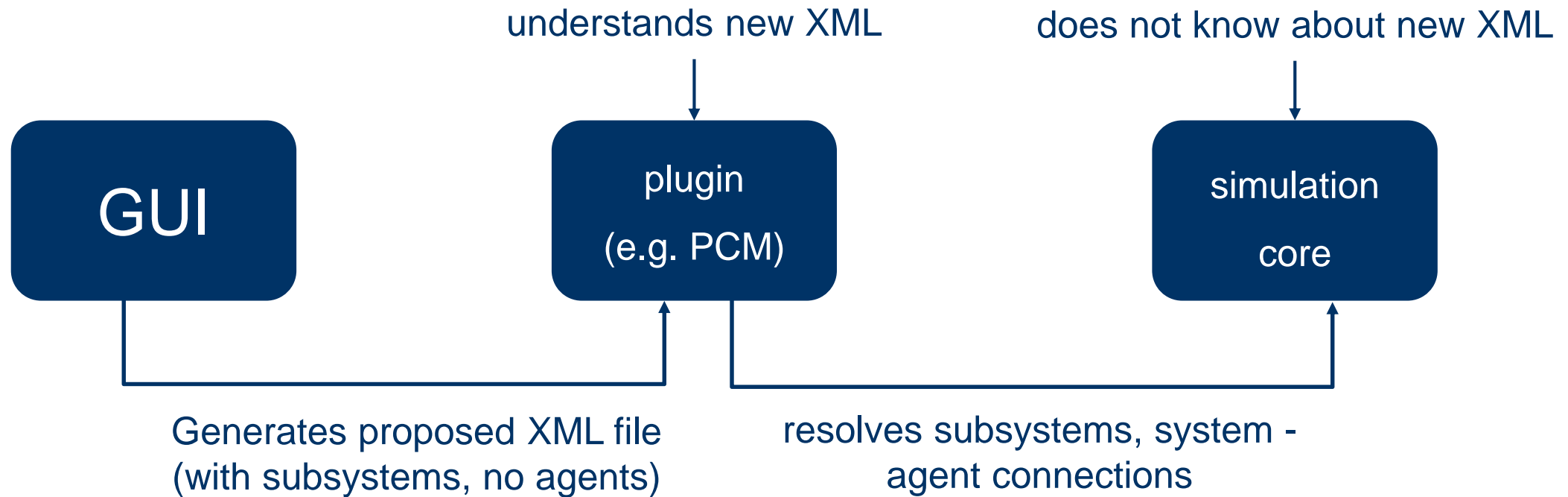
```

connection between
component and input



INCORPORATING THE NEW XML STRUCTURE INTO THE FRAMEWORK

Option 1: plugins have to resolve the subsystems (i.e. translate to old XML file structure)



CONCLUSIONS

Hierarchical system editor

- Definition of a new object type subsystem
- Subsystems act as both systems and components
- Creating and editing subsystems yields a benefit in usability
- **XML file structure has to be extended for defining subsystems**

Points to be decided on

- Confirmation of the new XML file structure
- How to incorporate the new XML file structure into the framework (option 1 or 2)