Embedding Temporal Requirements in Co-Simulation

A Collaborative Platform for Formalizing Requirements Authoring and Exchange in Co-Simulation Using Temporal Logic

OSP-Conference

Aalto-yliopisto
Aalto-universitetet
Aalto University

Aalto University
DigiTwin Lab

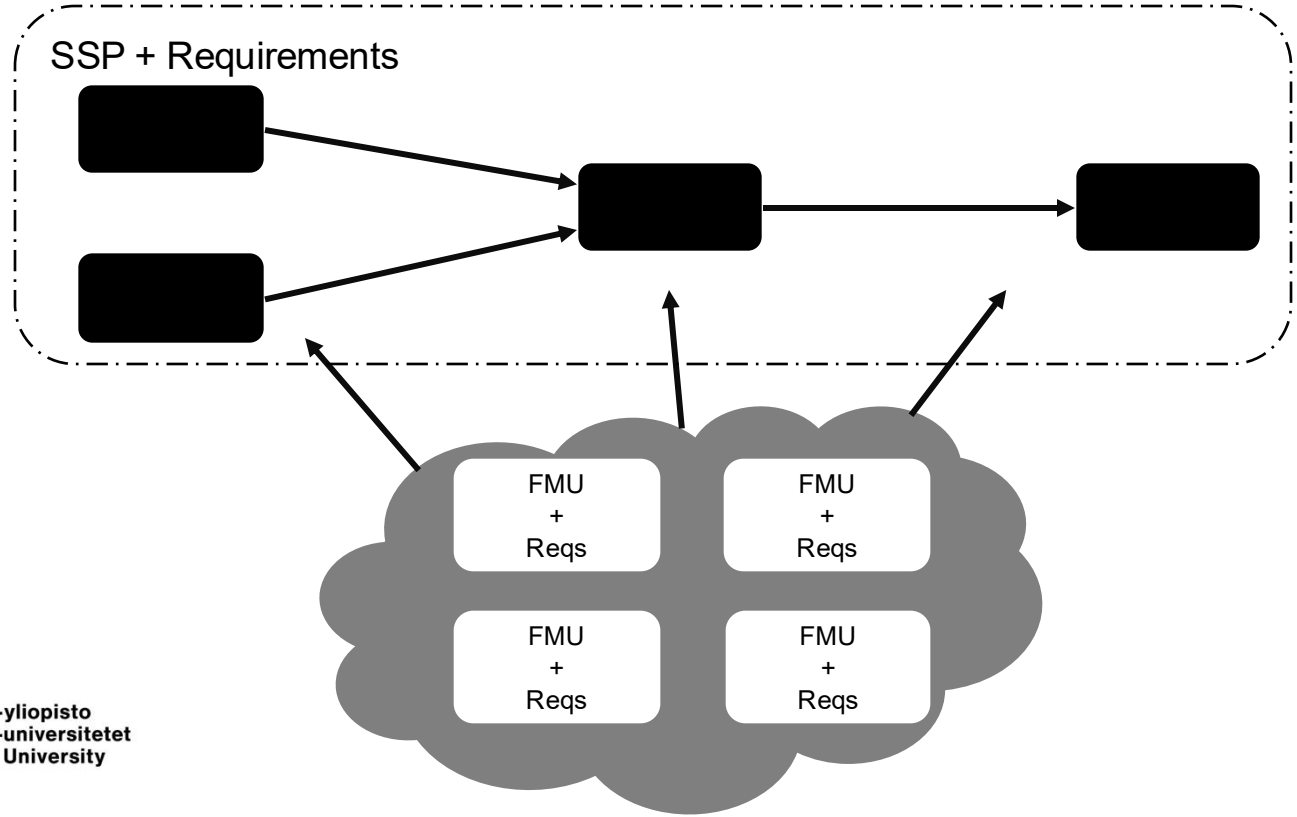
Haitham Al-Shami

12.11.2025

Funded by Business Finland

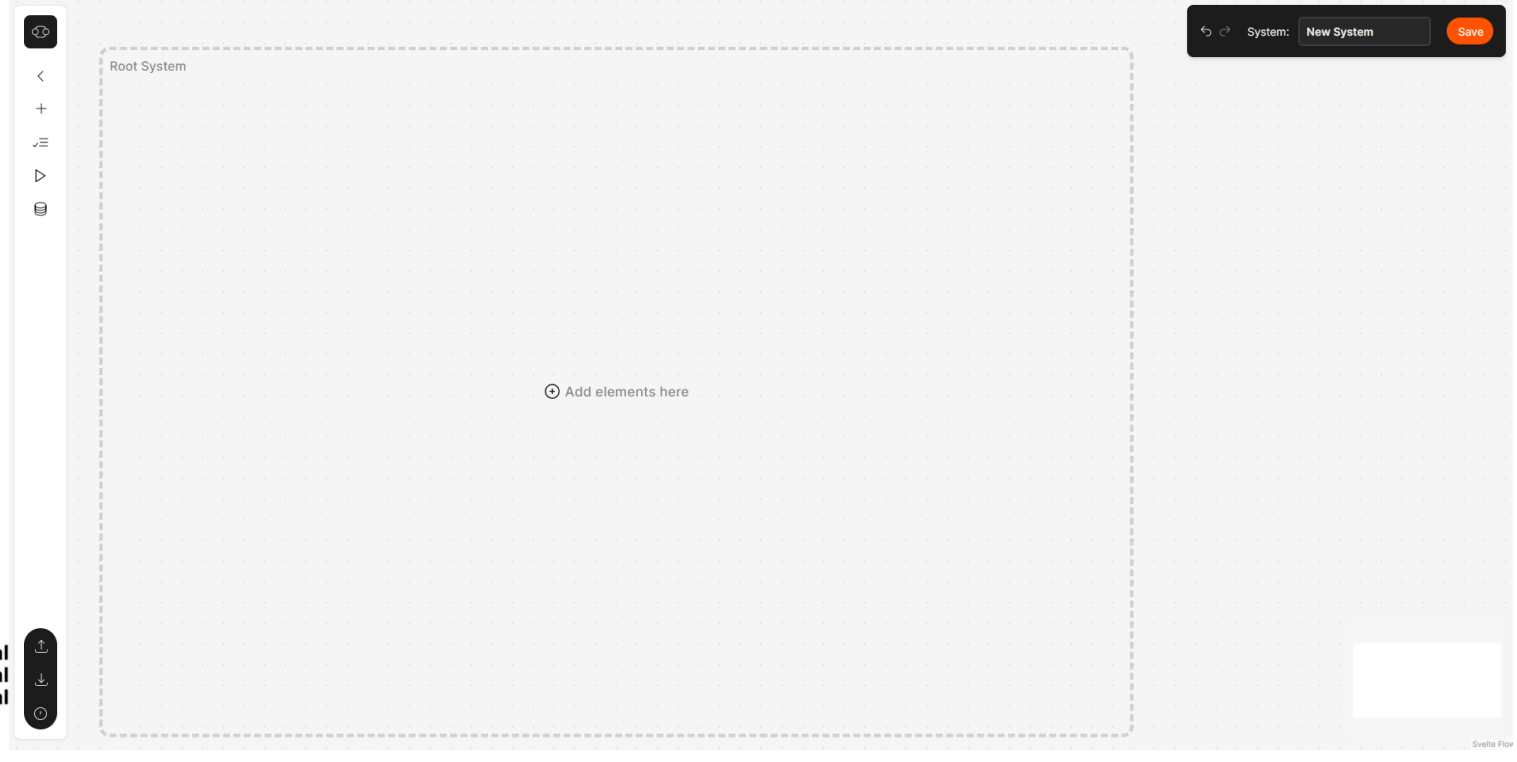
Introduction & Motivation

- Co-Simulation can enable collaborative, simulation-based engineering.
- The requirements of a system or its components are still encoded in natural language or tool specific way.
- In collaborative environments between system integrators and OEMs, can requirements be encoded alongside the simulation files?



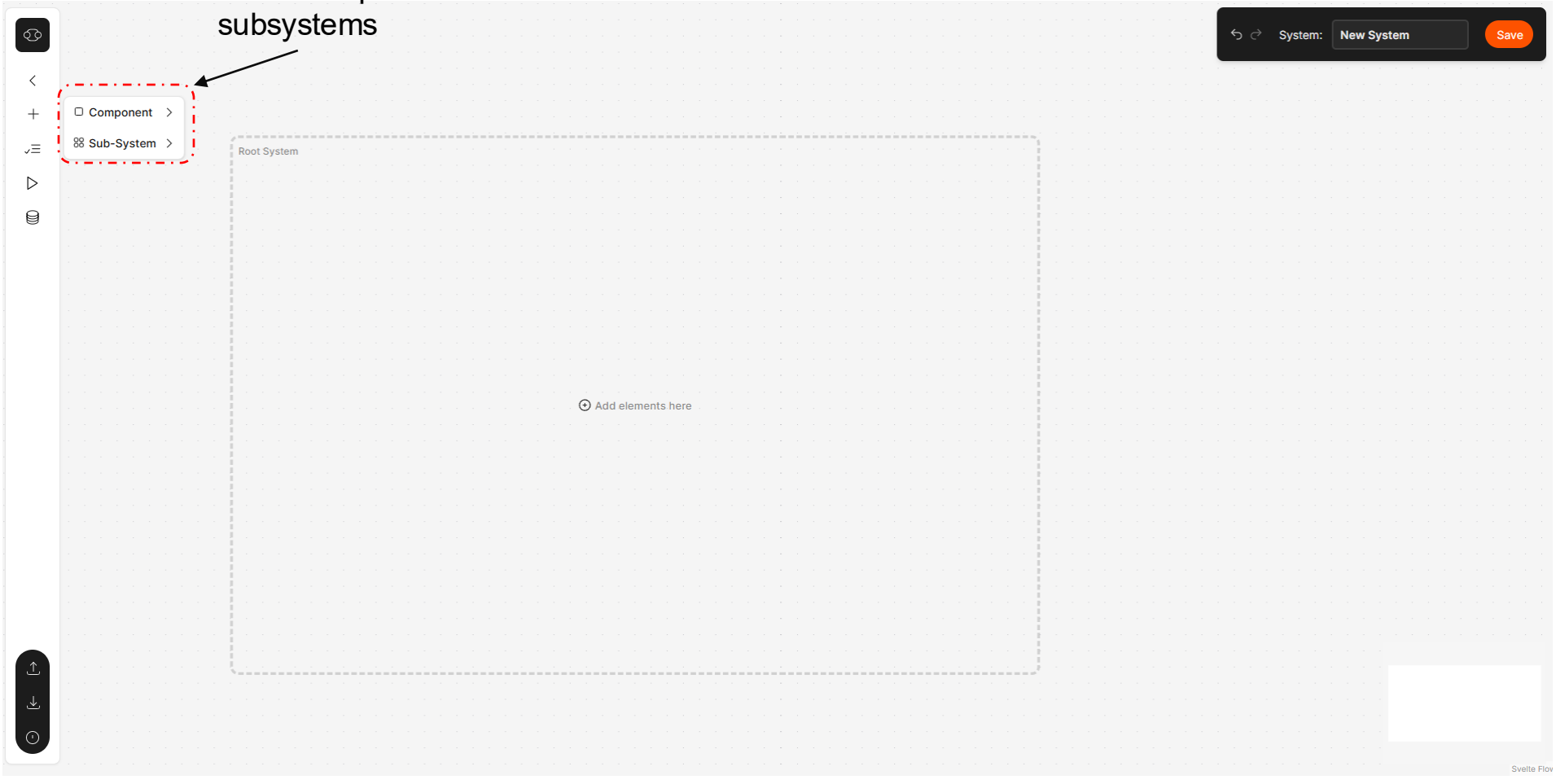
Objectives

- A way to write requirements in:
  1. Standard form.
  2. Quarriable and shareable.
  3. Suitable for runtime-verification during simulation.
  4. Usable by engineers.

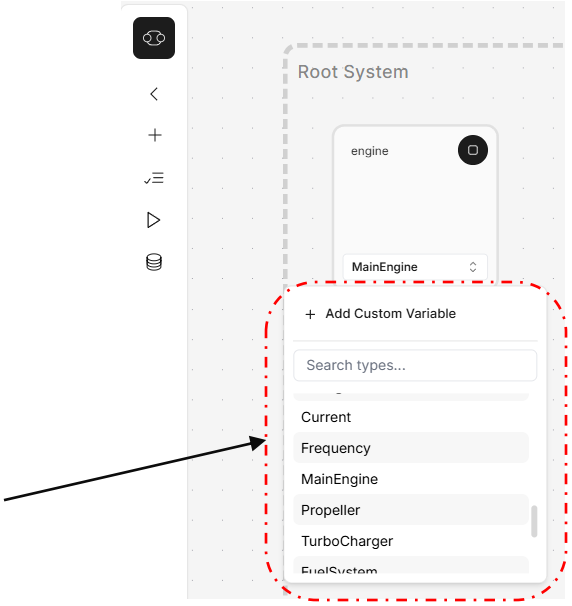


Aal
Aal
Aal

You can define components or subsystems

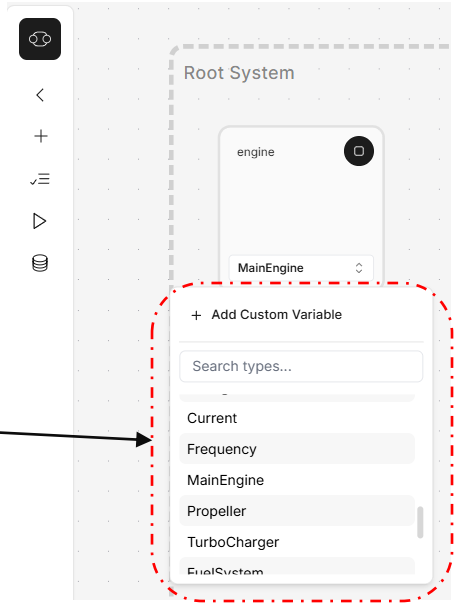


Each component follows a pre-defined standard terminology from a defined ontology, in this case the VSSo: Vehicle Signal Specification Ontology



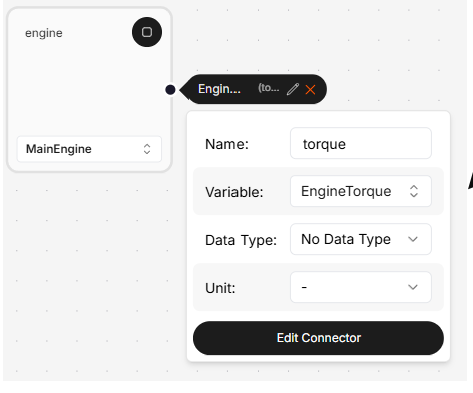
This is to match naming in the SSP and its requirements with the FMUs and their requirements

Each component follows a pre-defined standard terminology from a defined ontology, in this case the VSSo: Vehicle Signal Specification Ontology



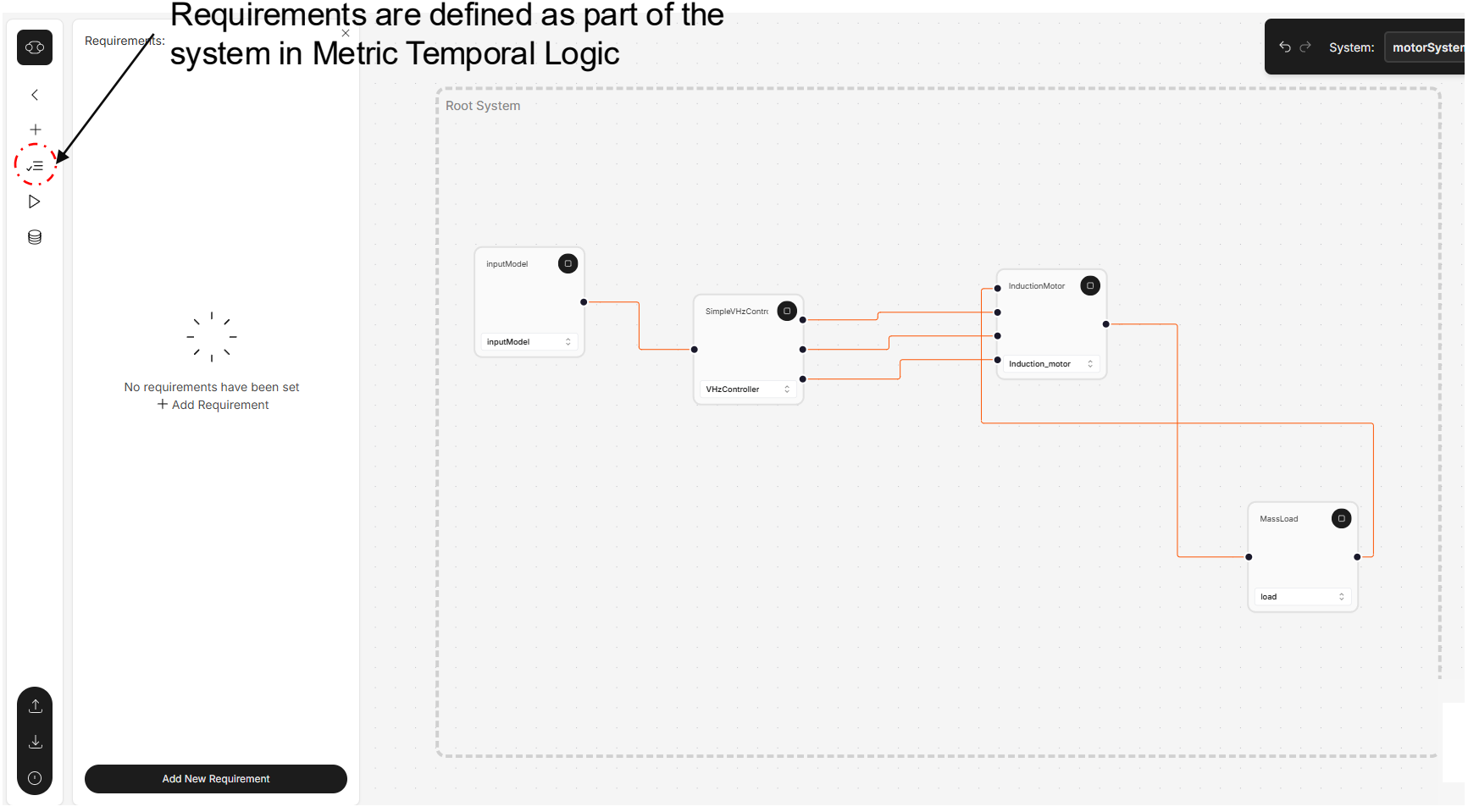
This is to match naming in the SSP and its requirements with the FMUs and their requirements

The resulting connection variable is MainEngine.EngineTorque

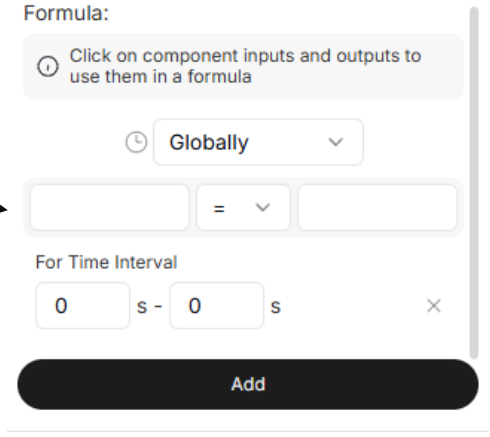


Aalto-yliopisto
Aalto-universitetet
Aalto University

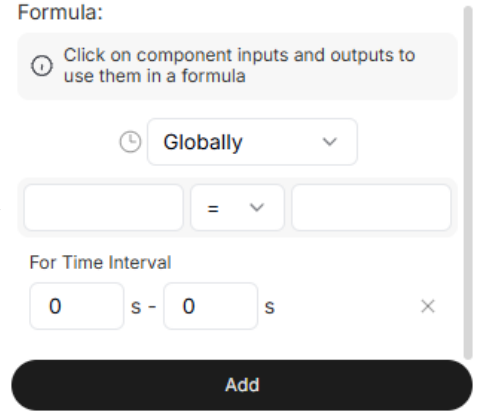
Requirements are defined as part of the system in Metric Temporal Logic



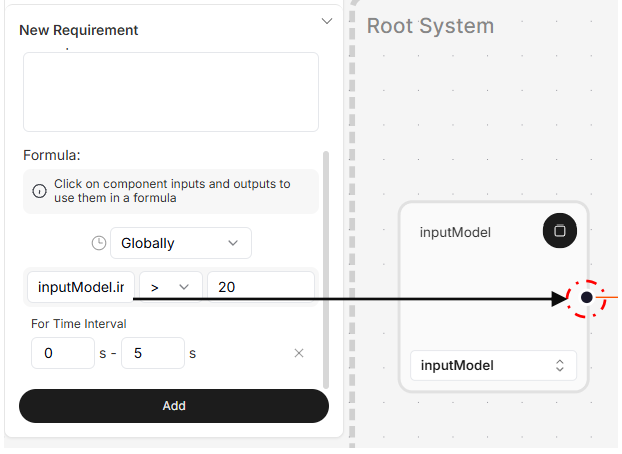
Requirements are defined as part of the system in Metric Temporal Logic



Requirements are defined as part of the system in Metric Temporal Logic



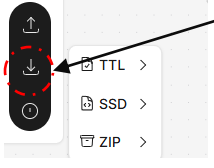
The full path is used in the requirement



All defined requirements

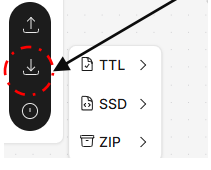
All defined requirements

Requirements can be downloaded in different formats



All defined requirements

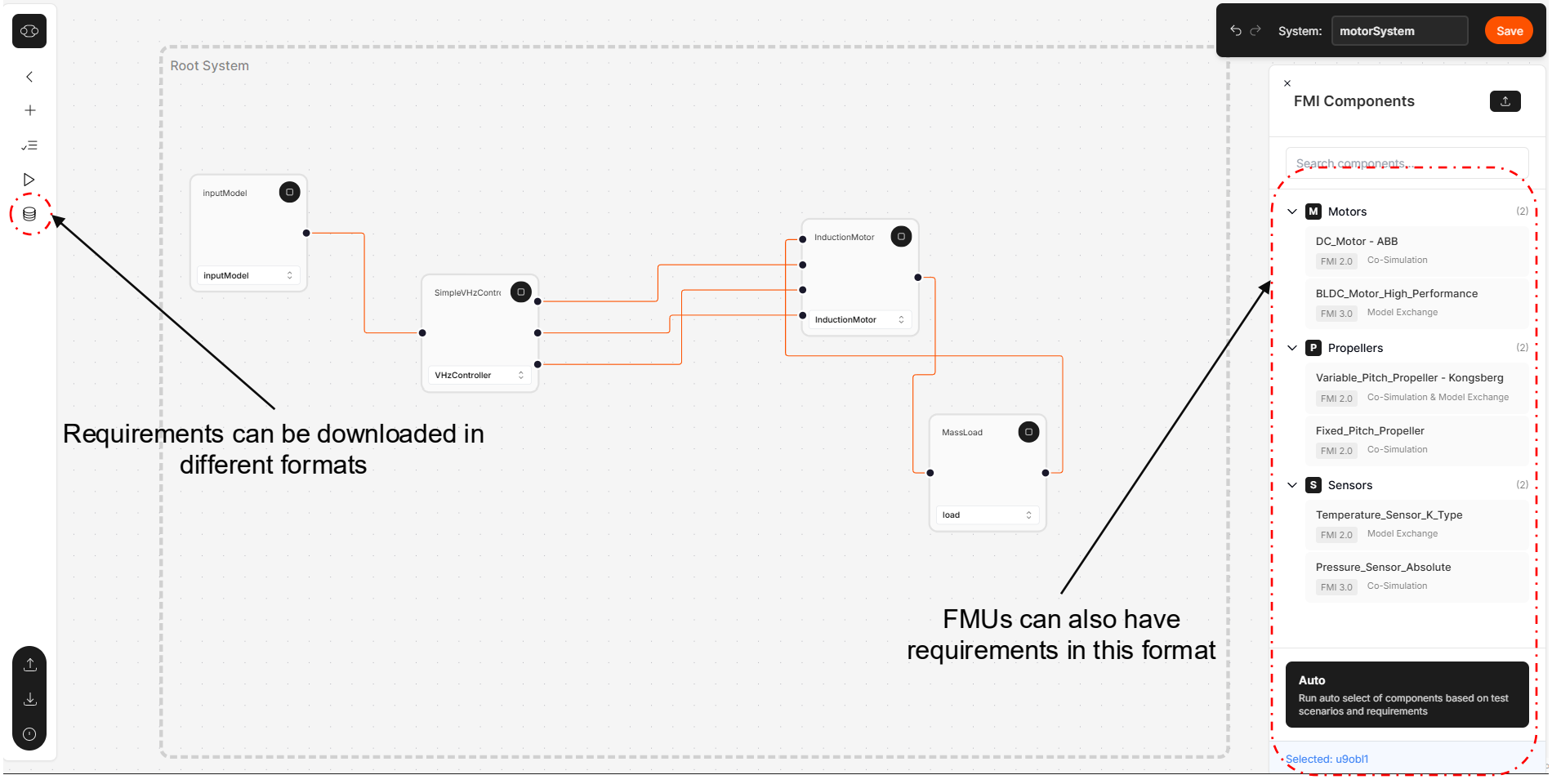
Requirements can be downloaded in different formats

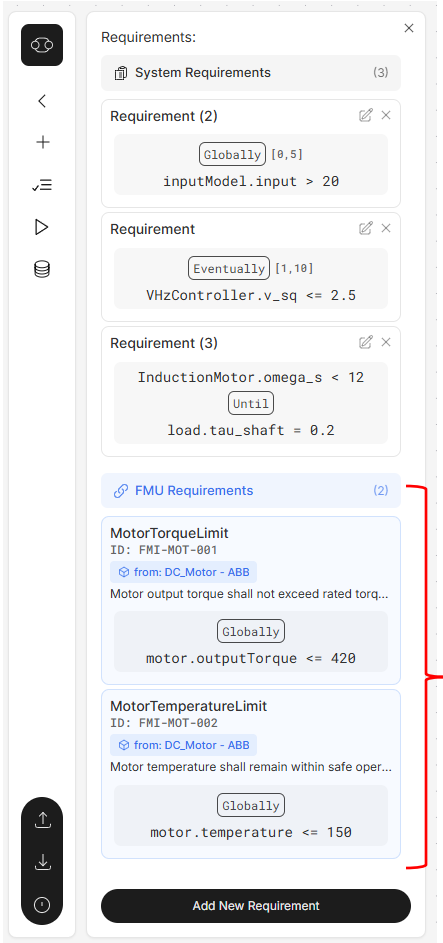



```
req:Requirement__2_ a mtl:MTLFormula ;
    mtl:hasOperator mtl:Globally ;
    mtl:hasTimeInterval [
        a time:Interval ;
        time:hasBeginning [ a time:Instant ; time:inTimePosition
        [ time:numericPosition "0.0"^^xsd:double ; time:hasTRS time:unitSecond ] ] ;
        time:hasEnd [ a time:Instant ; time:inTimePosition
        [ time:numericPosition "5.0"^^xsd:double ; time:hasTRS time:unitSecond ] ]
    ] ;
    mtl:hasRightOperand [
        a mtl:MTLFormula ;
        mtl:hasOperator mtl:GreaterThan ;
        mtl:hasLeftOperand "inputModel.input" ;
        mtl:hasRightOperand "20.0"^^xsd:double
    ] .

req:Requirement a mtl:MTLFormula ;
    mtl:hasOperator mtl:Eventually ;
    mtl:hasTimeInterval [
        a time:Interval ;
        time:hasBeginning [ a time:Instant ; time:inTimePosition
        [ time:numericPosition "1.0"^^xsd:double ; time:hasTRS time:unitSecond ] ] ;
        time:hasEnd [ a time:Instant ; time:inTimePosition
        [ time:numericPosition "10.0"^^xsd:double ; time:hasTRS time:unitSecond ] ]
    ] ;
    mtl:hasRightOperand [
        a mtl:MTLFormula ;
        mtl:hasOperator mtl:LessThanOrEqual ;
        mtl:hasLeftOperand "VHzController.v_sq" ;
        mtl:hasRightOperand "2.5"^^xsd:double
    ] .
```

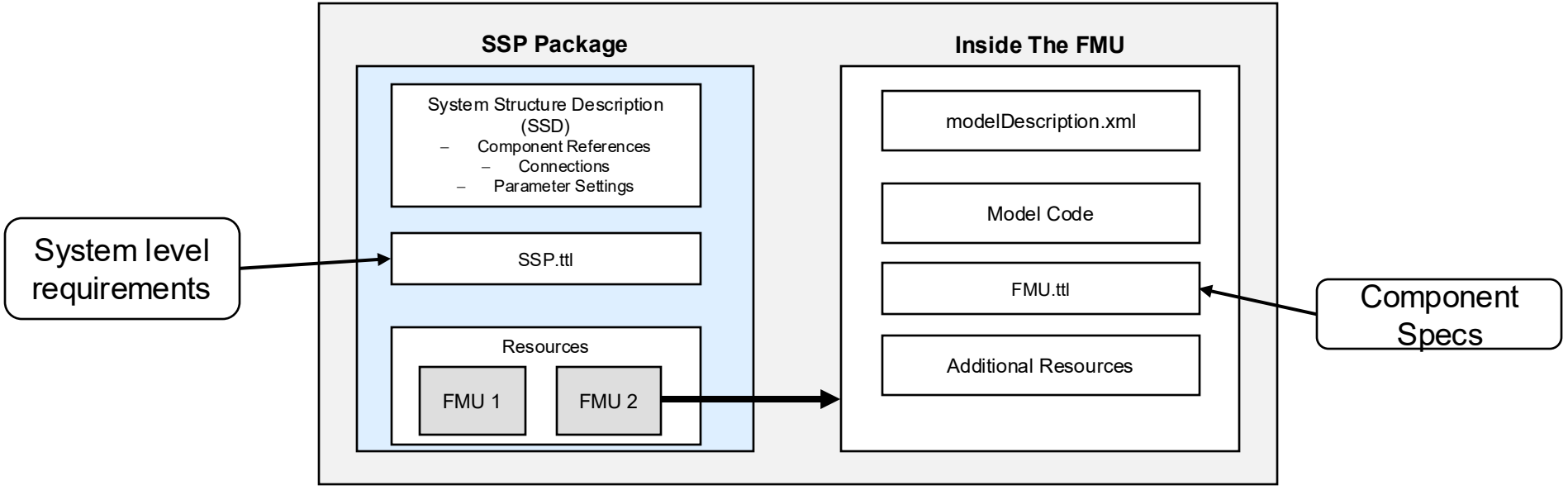
The requirements are written in the .TTL in RDF and OWL standard forms

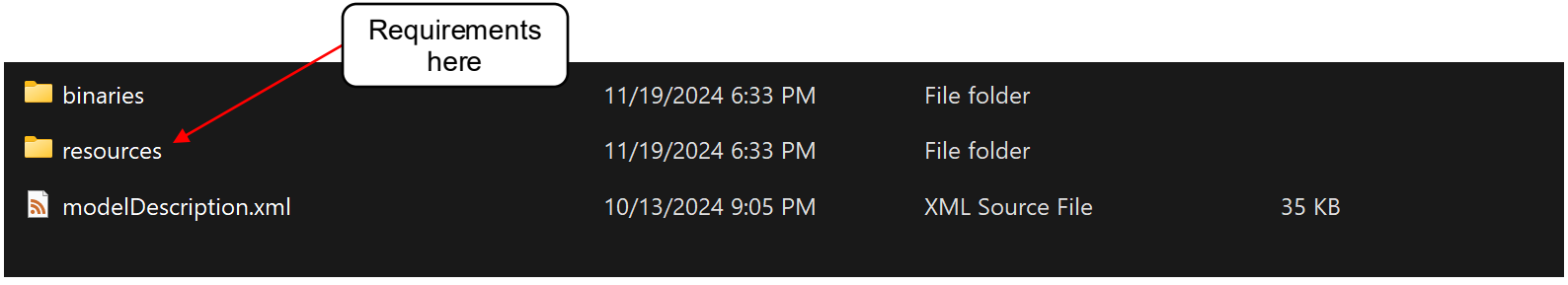

System:
motorSystem
Save
Root System
inputModel
inputModel
SimpleVHzContr
VHzController
InductionMotor
InductionMotor
MassLoad
load
FMI Components
Search components...
Motors (2)
DC_Motor - ABB
FMI 2.0 Co-Simulation
BLDC_Motor_High_Performance
FMI 3.0 Model Exchange
Propellers (2)
Variable_Pitch_Propeller - Kongsberg
FMI 2.0 Co-Simulation & Model Exchange
Fixed_Pitch_Propeller
FMI 2.0 Co-Simulation
Sensors (2)
Temperature_Sensor_K_Type
FMI 2.0 Model Exchange
Pressure_Sensor_Absolute
FMI 3.0 Co-Simulation
Auto
Run auto select of components based on test scenarios and requirements
Selected: u9obl1
Requirements can be downloaded in different formats
FMUs can also have requirements in this format



FMU requirements

SSP and FMU Integration





```
monitors = syncWrite (\() -> [
        testingStationWriteViolationMonitor "hull1" "finally (0, 600.0) (mtlConditionHolds(mtlGreaterThan(mtlGetVariable(\"ship_design.Root.HULL_001.V_knots\")
        testingStationWriteViolationMonitor "consumption1" "globally (290, 330) (mtlConditionHolds(mtlLessThan(mtlGetVariable(\"ship_design.Root.CONSUMPTION_00
        testingStationWriteViolationMonitor "eng1" "mtlNowAndForever(mtlConditionHolds(mtlLessThan(mtlGetVariable(\"ship_design.Root.ENGINE_001.T_ME\")::Double
        testingStationWriteViolationMonitor "gearbox1" "mtlNowAndForever(mtlConditionHolds(mtlLessThan(mtlGetVariable(\"ship_design.Root.GEARBOX_001.n_ME_scale
        testingStationWriteViolationMonitor "prop1" "((mtlNowAndForever(mtlConditionHolds(mtlGreaterThan(mtlGetVariable(\"ship_design.Root.PROPELLER_001.T\")::
        testingStationWriteViolationMonitor "sys1" "mtlNowAndForever(mtlConditionHolds(mtlLessThan(mtlGetVariable(\"ship_design.Root.HULL_001.V_knots\")::Doubl
        testingStationWriteViolationMonitor "sys2" "mtlCausesWithinInterval(mtlConditionHolds(mtlGreaterThan(mtlGetVariable(\"ship_design.Root.HULL_001.V_knots
        testingStationWriteViolationMonitor "sys3" "mtlNowAndForever(((mtlConditionHolds(mtlGreaterThan(mtlGetVariable(\"ship_design.Root.HULL_001.V_knots\")::
        ]
) :: [Resource]
```

8 Requirements are tested, drive cycle was 18 hours

```
Test monitor: hull1, verdict: Violated
Test monitor: consumption1, verdict: Violated
Test monitor: eng1, verdict: Not Violated
Test monitor: gearbox1, verdict: Not Violated
Test monitor: prop1, verdict: Violated
Test monitor: sys1, verdict: Not Violated
Test monitor: sys2, verdict: Violated
Test monitor: sys3, verdict: Violated
Simulation time: 65000.0
```

Verdict showing which requirements were violated

Future work & Conclusions

- The platform can define requirements along side co-simulation methods in a standardized, shareable and quarriable way.
- This way, sharing models can carry more meaningful data that is relevant to OEMs and designers.
- The platform is a way to abstract away complexity of working with ontologies

- Integrate the simulator with runtime verification to the platform
- Add a way to manage and organize requirements, SPARQL gives a path towards to this.

Thank You!

Website at: ddtlab.aalto.fi

Aalto-yliopisto
Aalto-universitetet
Aalto University

Haitham Al-Shami

12.11.2025