CLOSING THE DESIGN CYCLE LOOP WITH EXECUTABLE REQUIREMENTS AND OSLC

Hubertus Tummescheit, Modelon
Bob Sherman, Procter & Gamble
Juan Llorens, The Reuse Company

INCOSE IW 2017
MBSE Workshop

AGENDA

- Motivation: Systems Engineering and Modeling and Simulation need to converge
- Open Standards we build on: Modelica, FMI, OSLC, SySML
- An Ideal Process to Integrate Systems Engineering with Model Based Design
- Continuous Integration to Close the Loop for Rapid Design Iterations
- First Steps to Automate Requirements Formalization
- Call to Action

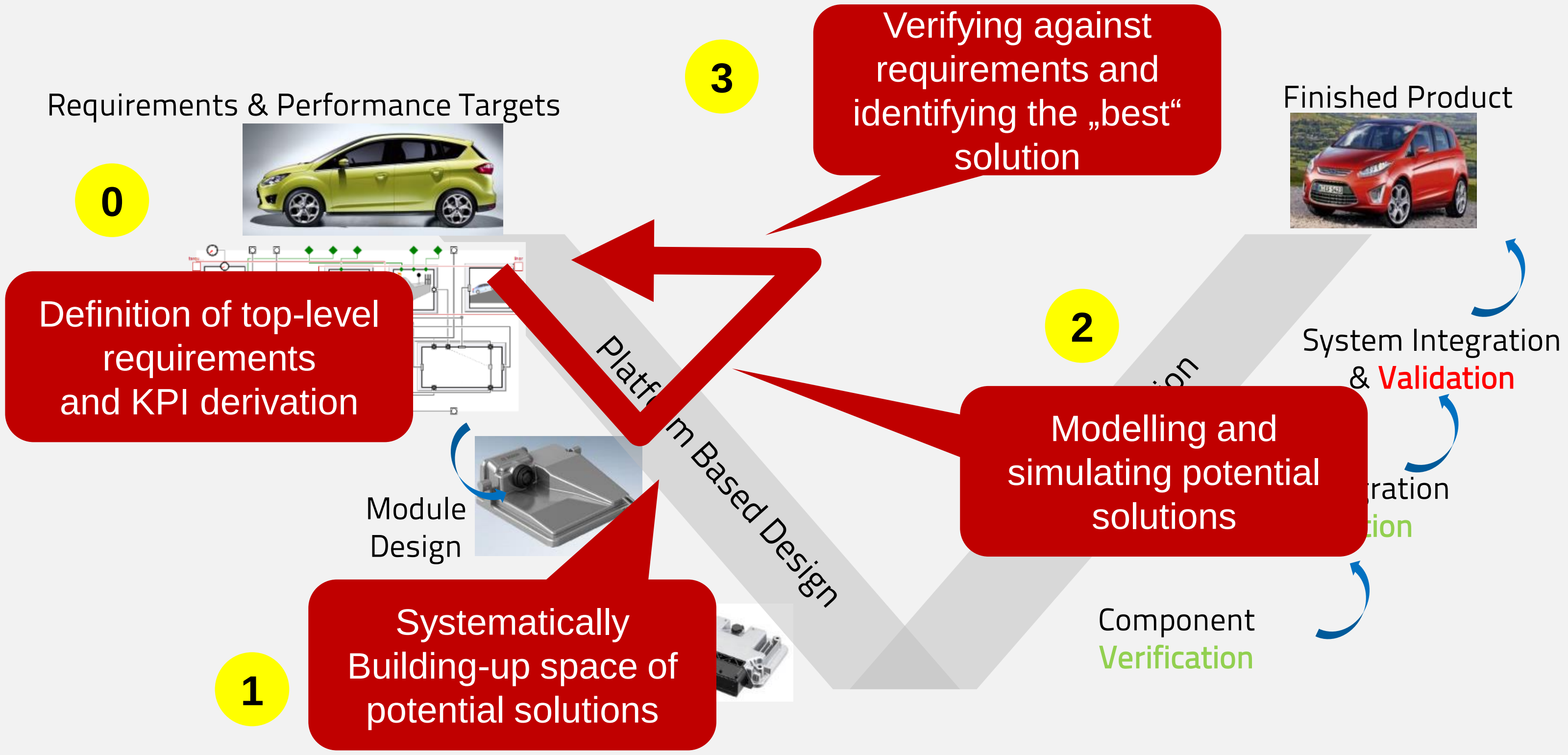
SYSTEMS ENGINEERING AND MODEL BASED DESIGN

Two worlds that need to converge

Modeling & Simulation IN THE V-MODEL is necessary Today

But SE tools and Simulation tools Typically don’t Work together

Simulation-in-the-loop along the Design Flow of the Systems Engineering V

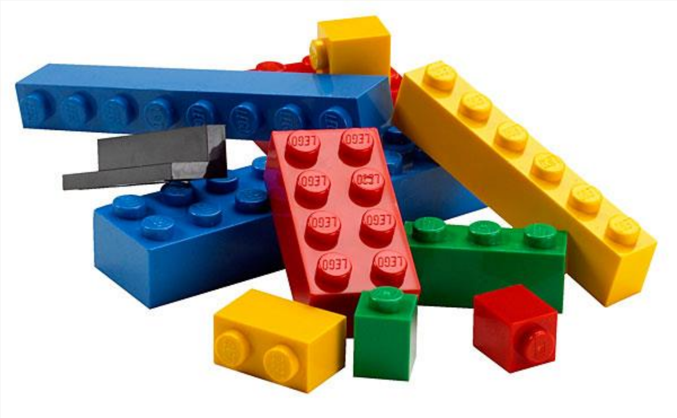


Many industries do this all the time, but the tools are not integrated !

MODELICA: THE OPEN STANDARDS SYSTEM LANGUAGE

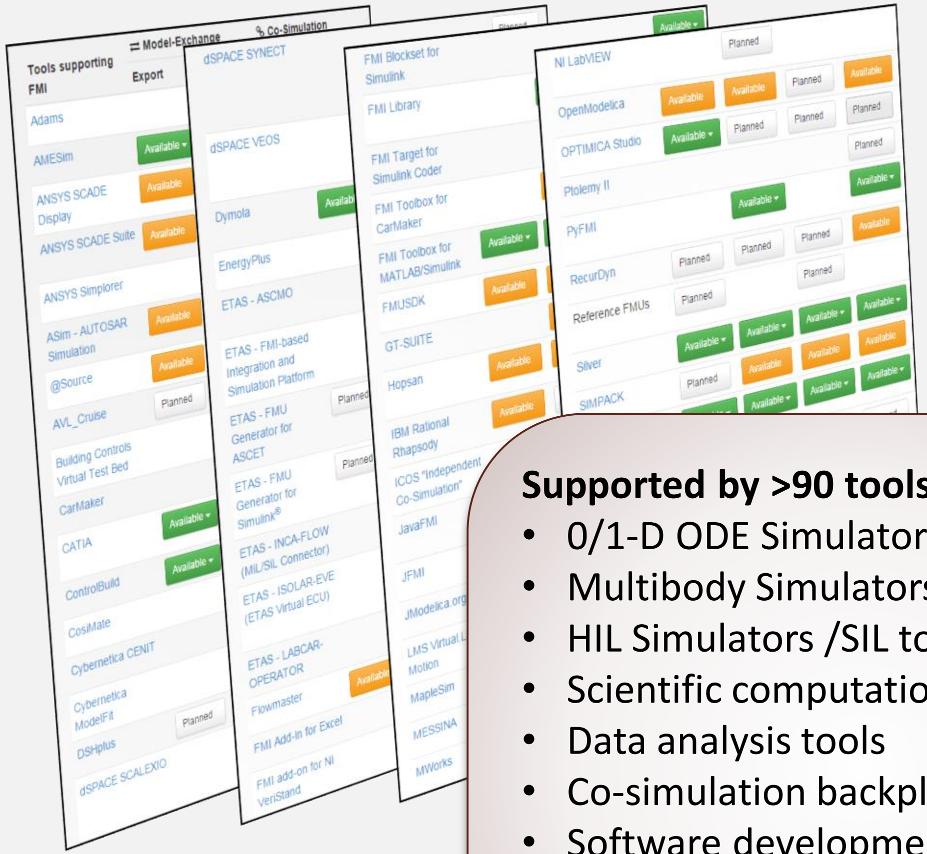
Modelica® is a non-proprietary, object-oriented, equation based language to conveniently model complex physical systems containing, e.g., mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents

- Object oriented modeling language
- Non-causal and equation based
- First principles (mass, energy, momentum balances)
- Supports multi-domain modeling
- Available in more than 10 different tools



FMI IN A NUTSHELL

- What is FMI?
  - an application programming interface and its semantics
  - an xml schema that describes the model structure and capabilities
  - the structure of a zip file that is used to package the model, its resources and documentation.
- > 90 tools support FMI in 10 different categories.

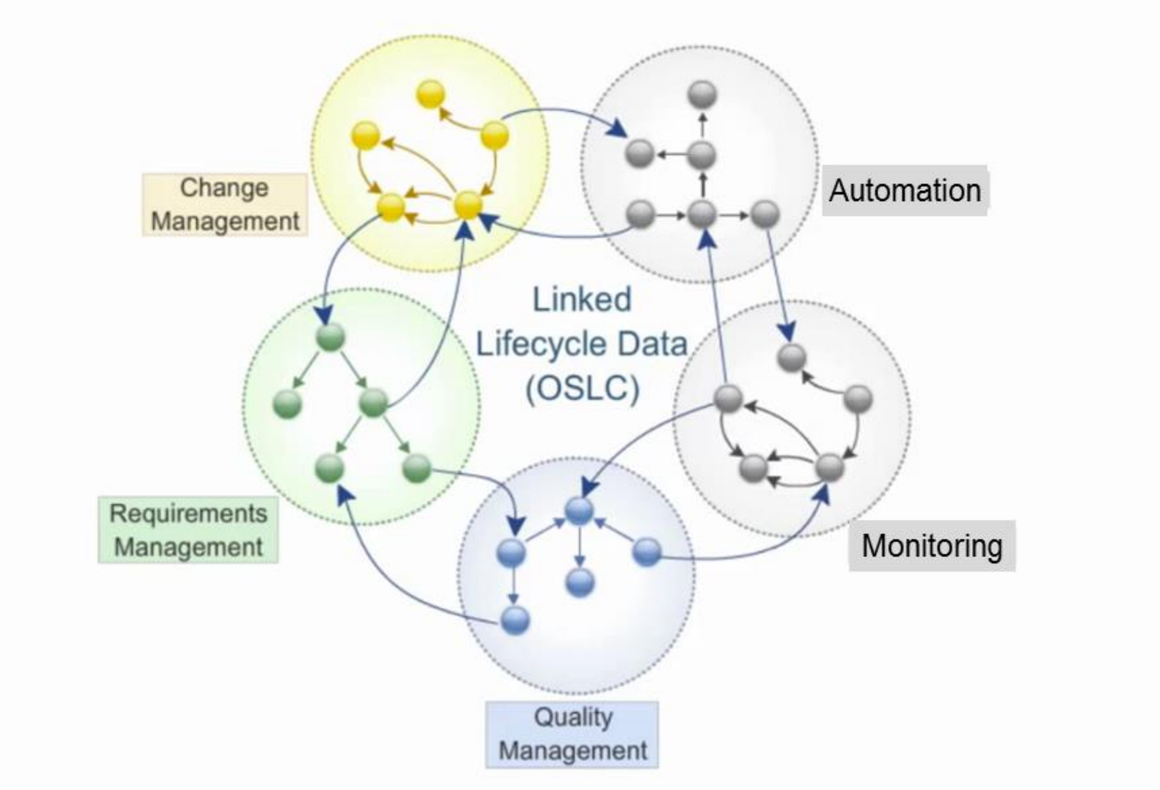


Supported by >90 tools:

- 0/1-D ODE Simulators
- Multibody Simulators
- HIL Simulators /SIL tool chains
- Scientific computation tools
- Data analysis tools
- Co-simulation backplanes
- Software development tools
- Systems engineering tools
- Process integration and optimization tools
- SDKs

OPEN SERVICES FOR LIFECYCLE COLLABORATION (OSLC)

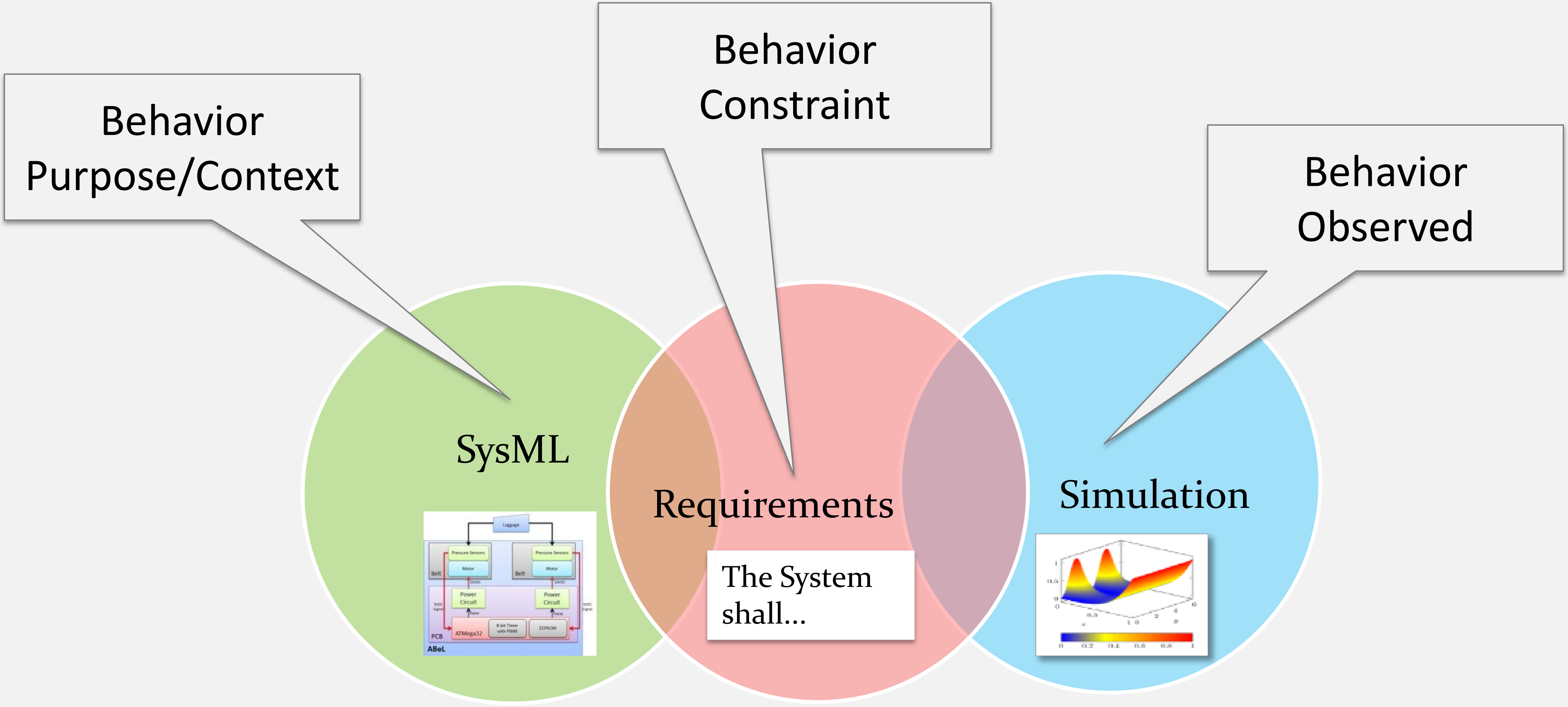
- OSLC = reusing web standards for tool integration
- Based on Web standards **linked data** and **RESTful Web services**
- Create specifications for interactions between tools
- Initiated by IBM, now managed by OASIS
- Focus on software-and systems engineering
- Not much traction (yet) with M&S tools



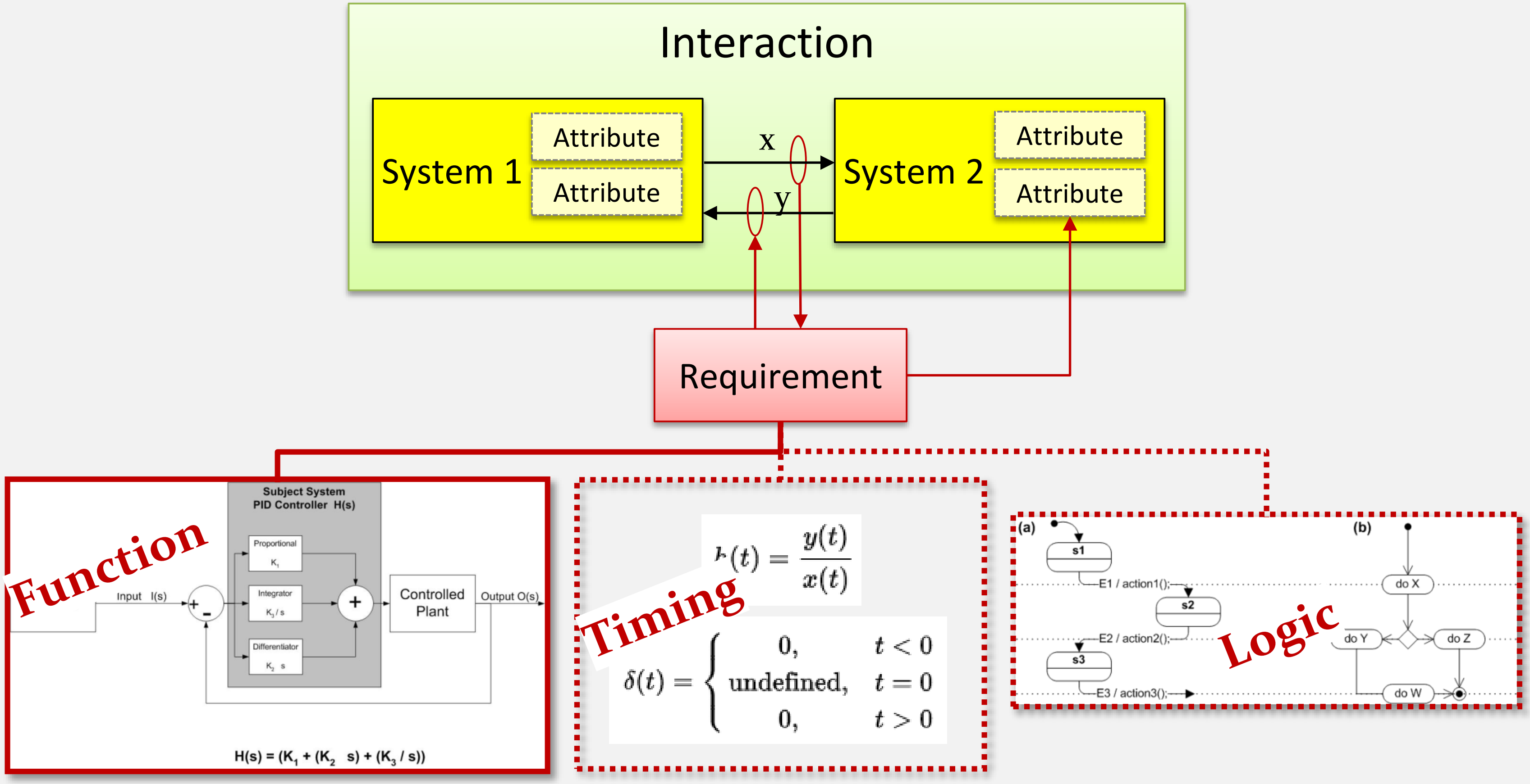
We built an open-source OSLC-to-FMI connector to link simulation results and parameters to life cycle tools

AN IDEAL PROCESS TO INTEGRATE SYSTEMS ENGINEERING WITH MODEL BASED DESIGN

Semantic Integration

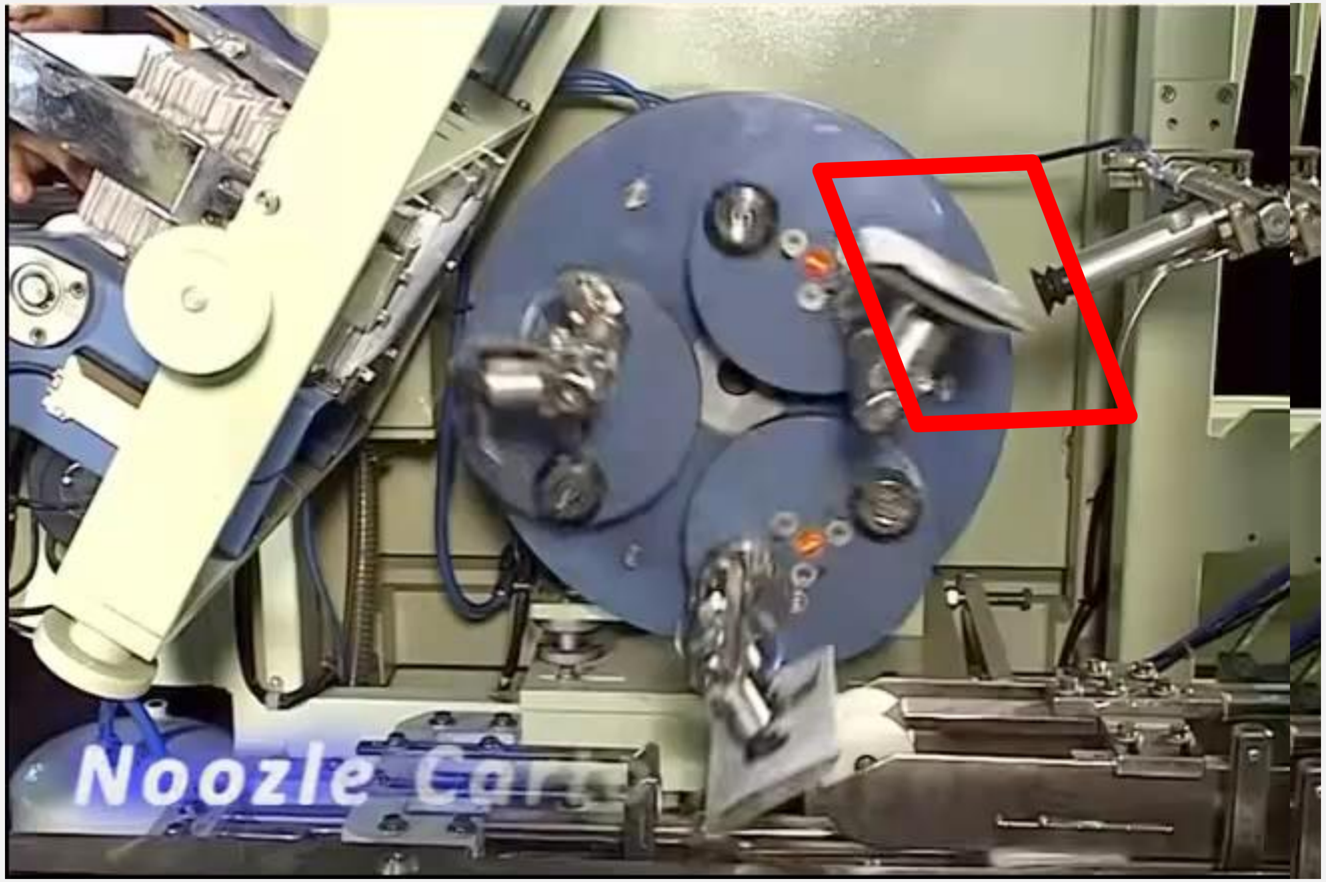


Purpose, Context & Anatomy of a Requirement

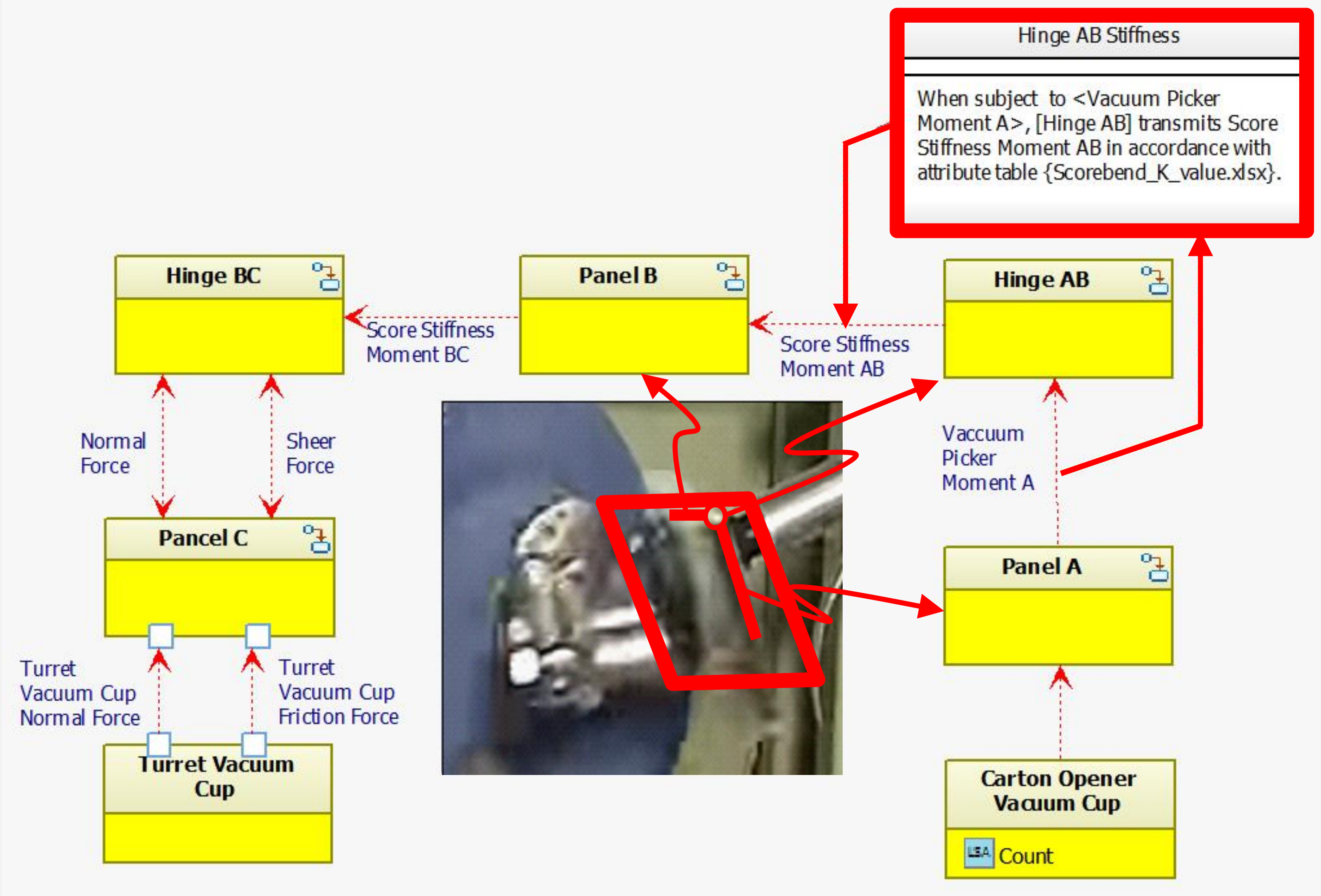


Bill Schindel (of ICTT):
"Requirements are Transfer Functions"

Example System



Example Requirement
(Transfer Function)



Example Requirement

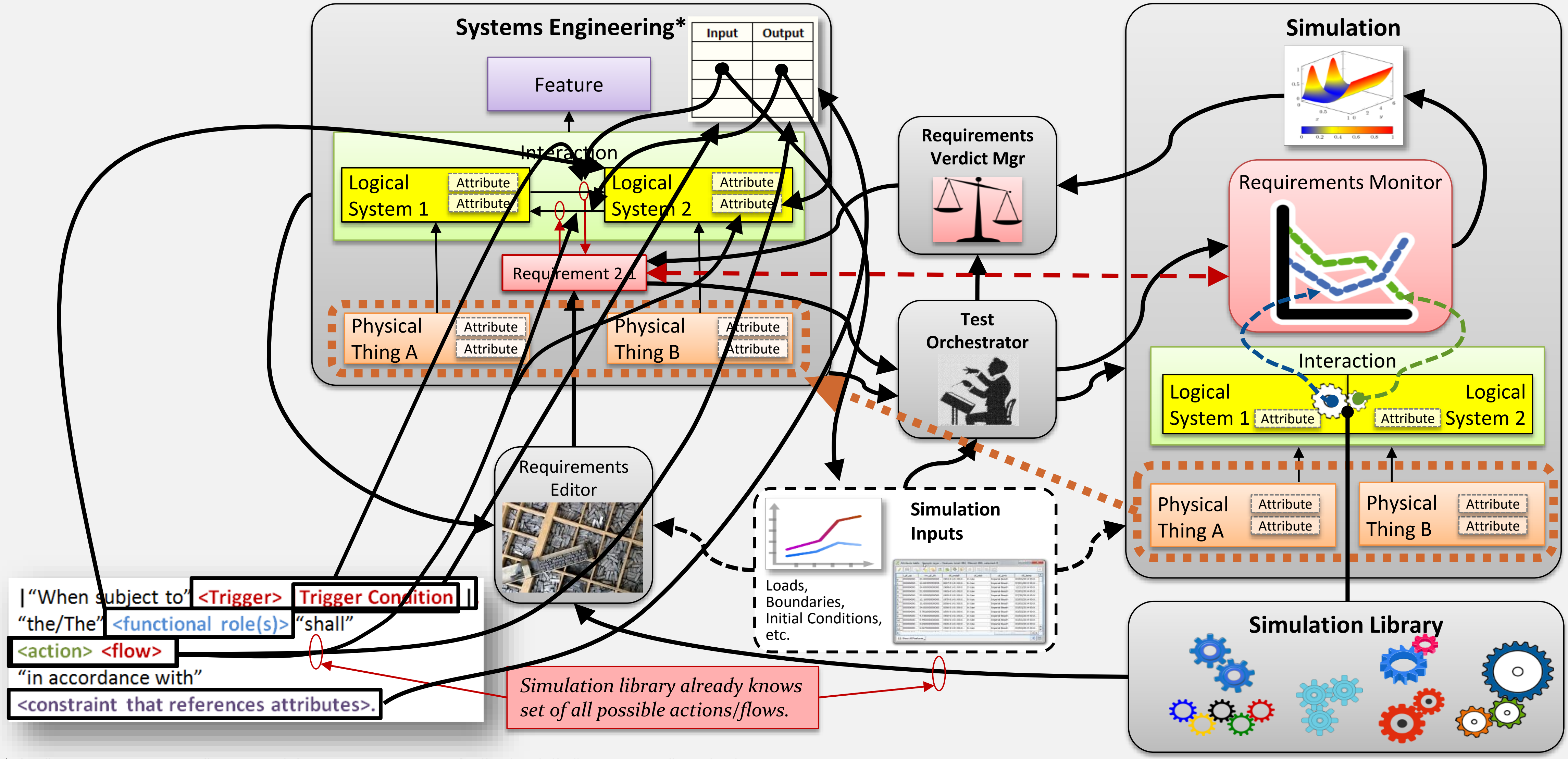
(Transfer Function)

When subject to Vacuum Picker Moment A, Hinge AB shall transmit Score Stiffness Moment AB
in accordance with attribute table: Scorebend_K_Value_Table.

|"When subject to" <Trigger> |Trigger Condition||, "the/The" <functional role(s)> "shall"
<action> <flow>
"in accordance with" <constraint that references attributes>.

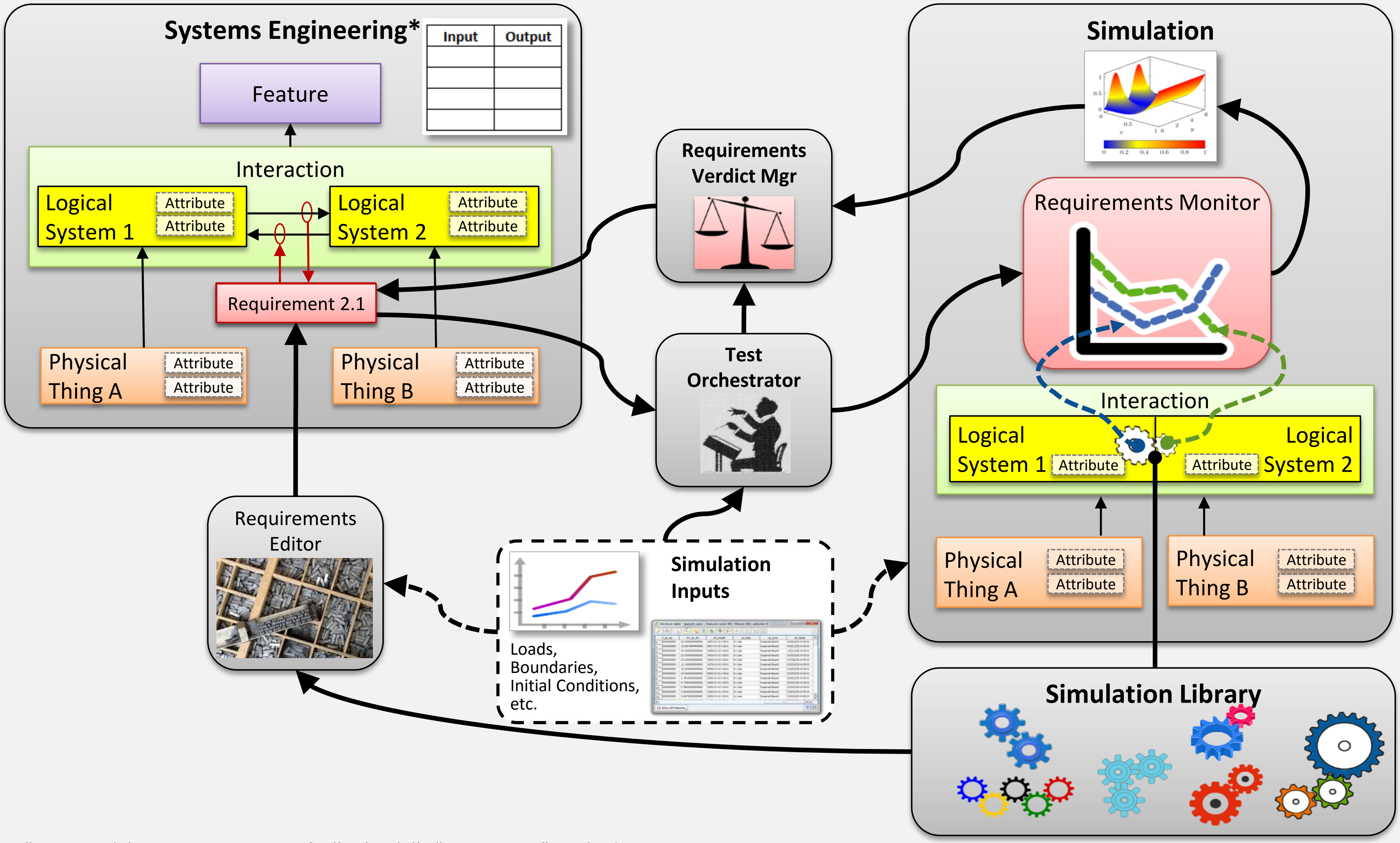
??? Machine readable requirements statement ???

Prospective SE and M&S Integration Strategy



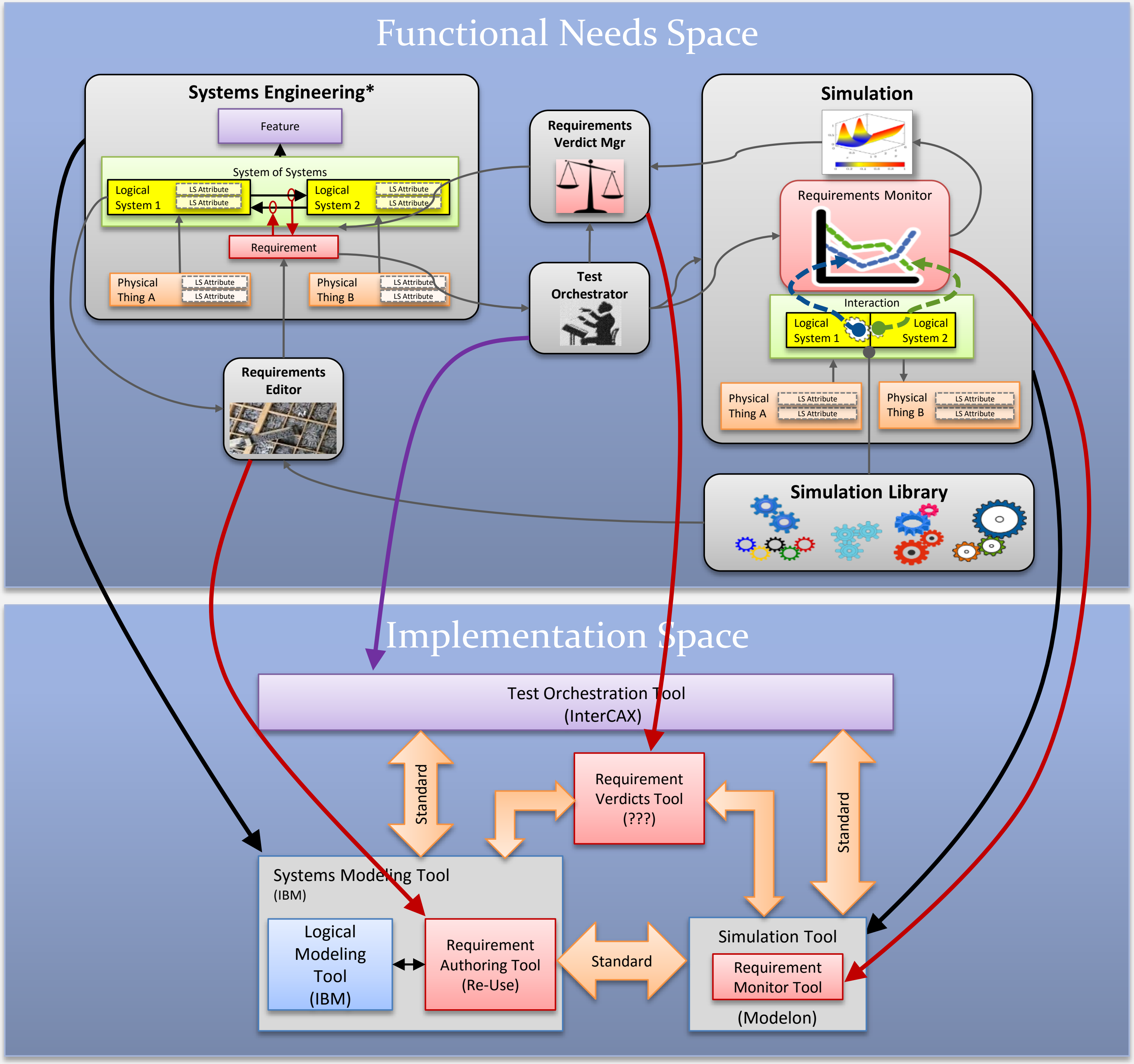
*The "Systems Engineering" metamodel is a representation of Bill Schindel's "Systematica" method.

Prospective SE and M&S Integration Strategy



*The "Systems Engineering" metamodel is a representation of Bill Schindel's "Systematica" method.

Prospective Mapping of Functional Architecture to Tool Suppliers

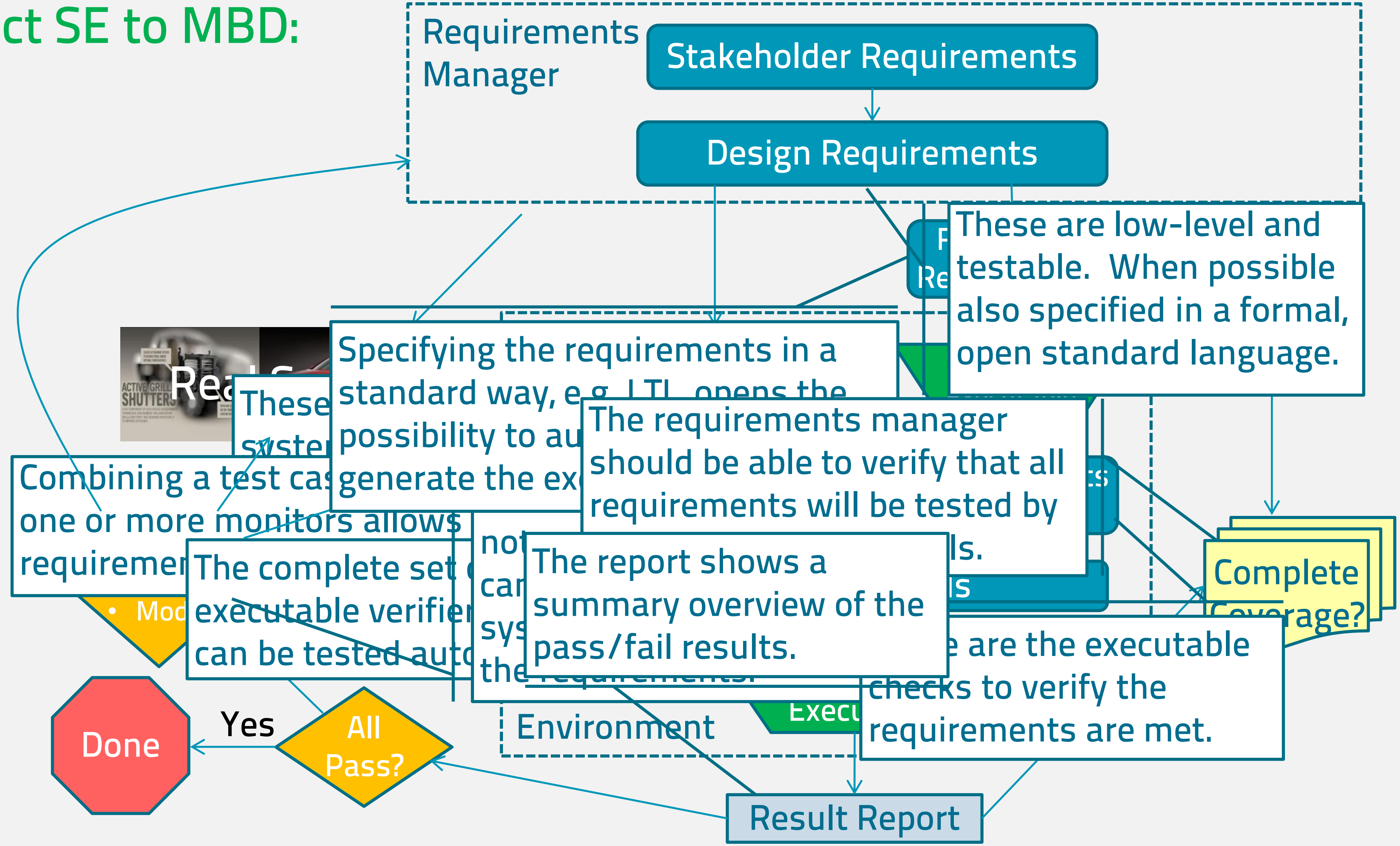


EXECUTABLE REQUIREMENTS

Continuous feedback on compliance of requirements

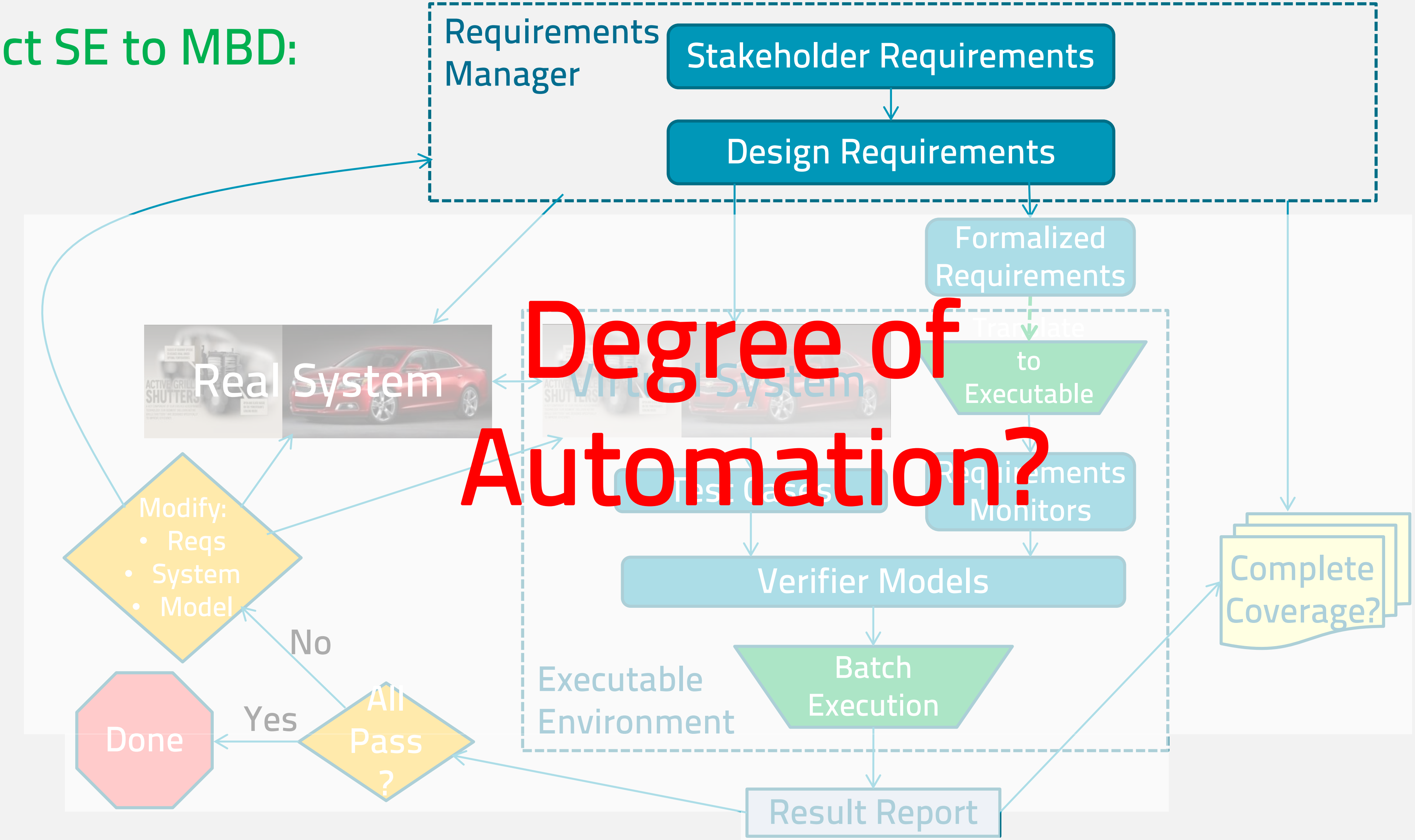
IN-THE-LOOP REQUIREMENTS VERIFICATION

Connect SE to MBD:



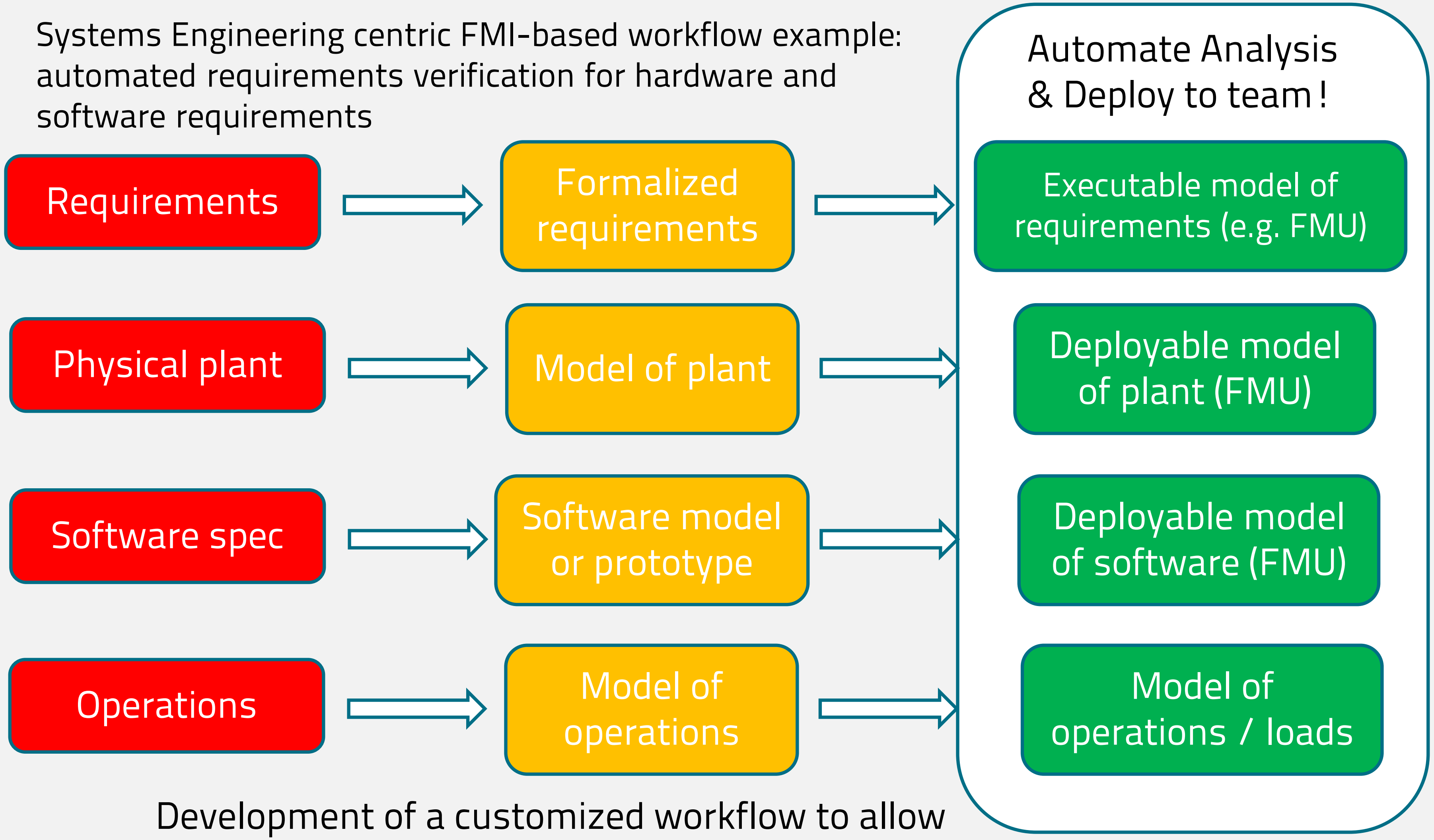
AUTOMATED REQUIREMENTS VERIFICATION

Connect SE to MBD:



AUTOMATED REQUIREMENTS VERIFICATION

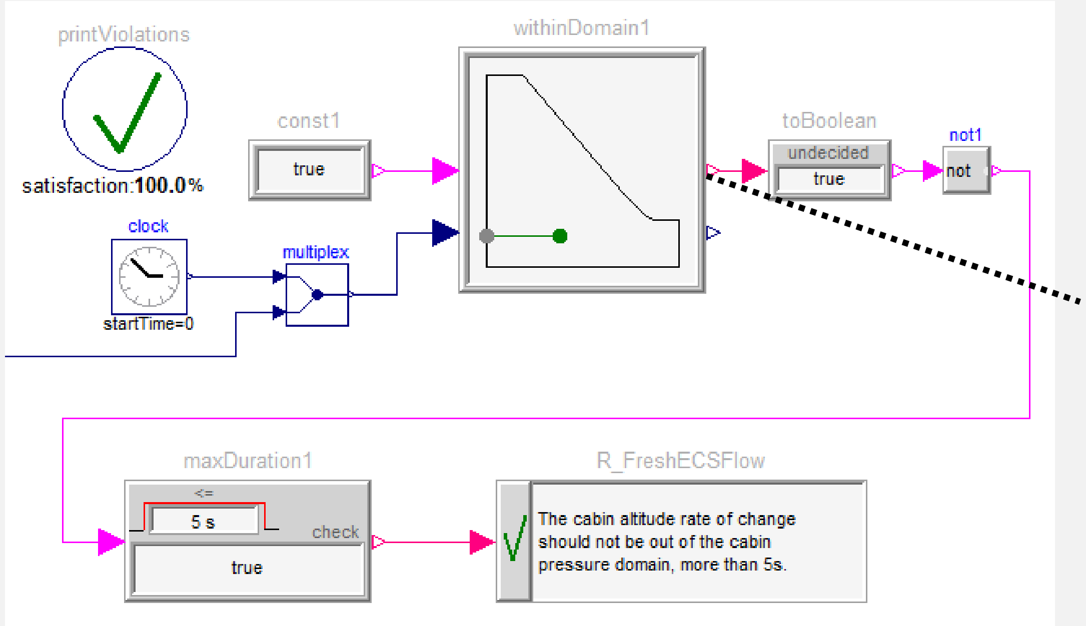
- Systems Engineering centric FMI-based workflow example: automated requirements verification for hardware and software requirements



Development of a customized workflow to allow rapid iterations of plant & software configuration

RESEARCH IMPLEMENTATION: REQUIREMENTS IN MODELICA

- Open Source Modelica library, based on 3-valued logic: Satisfied, Undecided, Violated
- Large Library of pre-defined requirement structures
- → Executable and formal model of requirements, in Modelica language

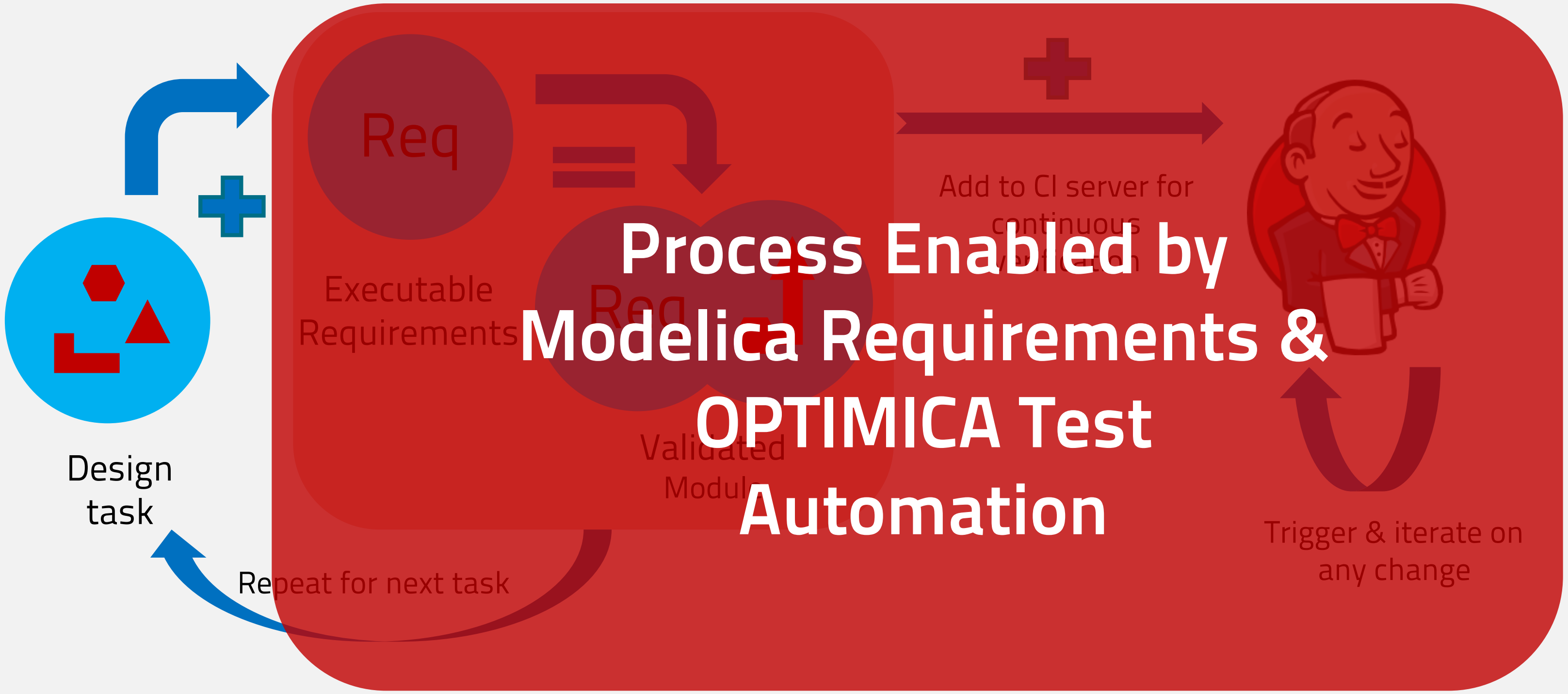


(x,y) coordinates of input must stay within closed polygon (output: closest distance to polygon + property)

CONTINUOUS INTEGRATION OF REQUIREMENTS VERIFICATION

Test Automation with Optimica Testing Tools (OTT)

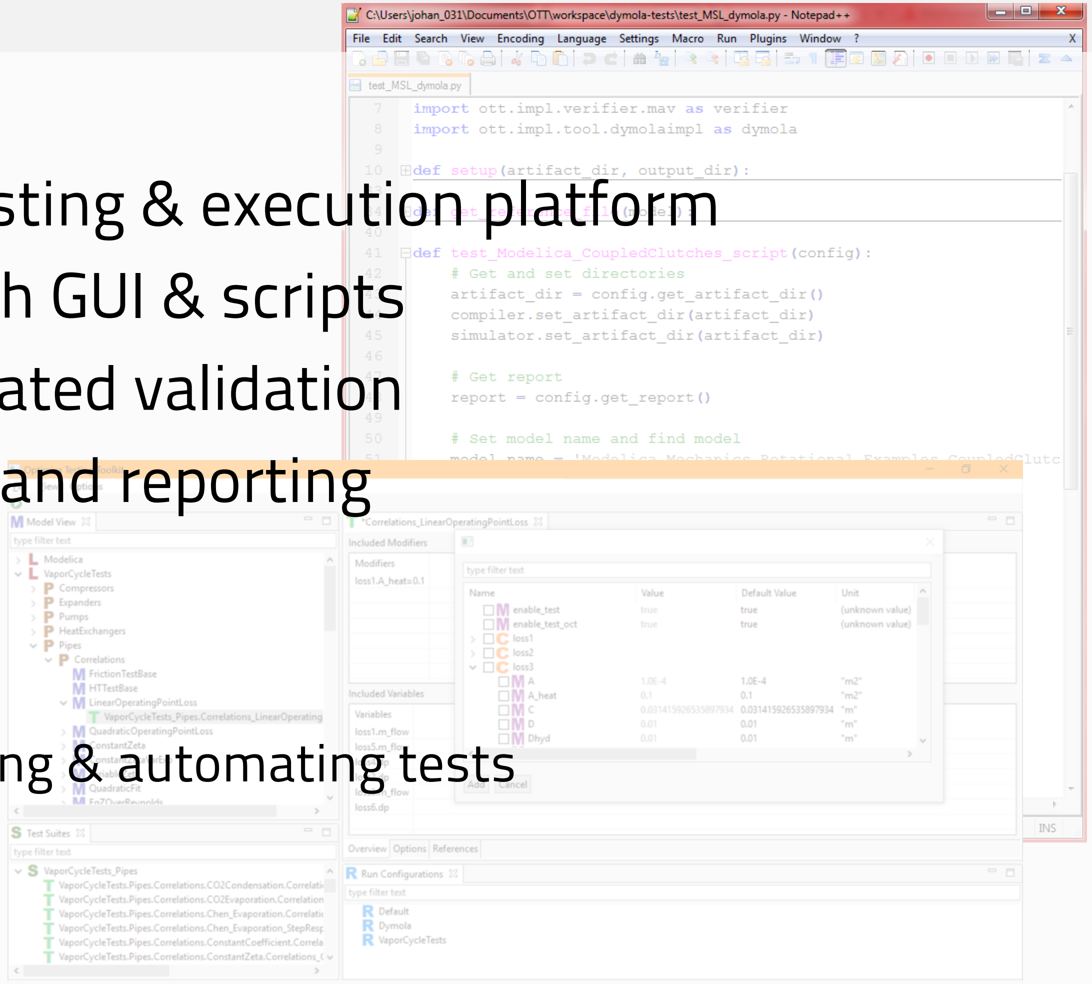
EXECUTABLE REQUIREMENTS FOR DESIGN ENGINEERS



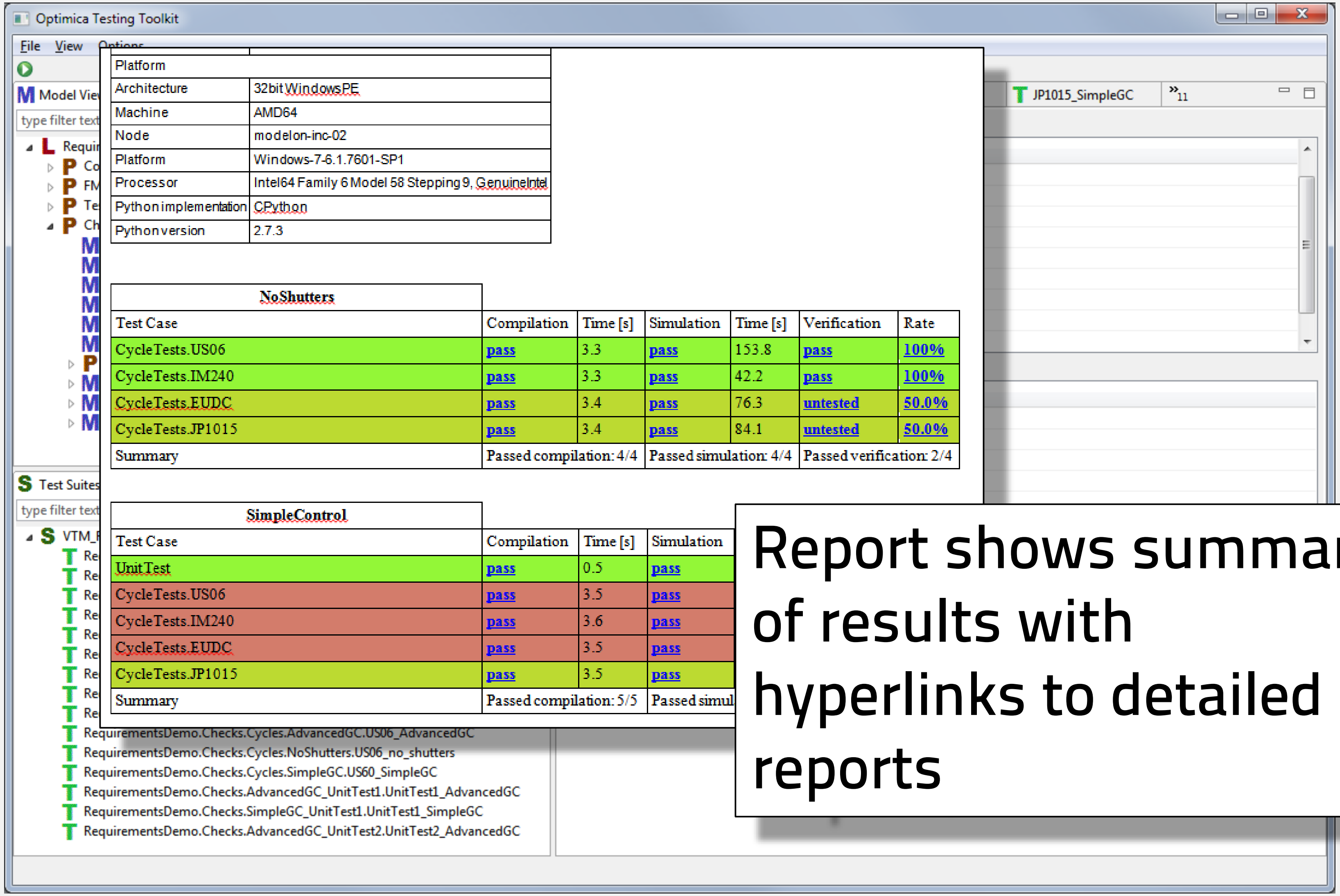
1. Design task (e.g. controller with given performance metric)
   1. Designer has access to a model with executable requirements monitors
   2. Designer validates requirements with each design iteration interactively
   3. Designer adds finished models of design and requirements to Continuous Integration server & trigger for automated re-testing
2. Designer moves to next task and repeats process
3. Observe productivity gain and fewer turn-backs

OPTIMICA TESTING TOOLKIT

- Key features
  - Modelica and FMI cross testing & execution platform
  - Flexible test authoring, with GUI & scripts
  - Simulation-specific automated validation
  - Automated test execution and reporting
- Architecture
  - Core
    - Command line tool for running & automating tests
    - Integrated with Jenkins
  - GUI
    - Tool for creating, updating and running tests
    - Reviewing and updating results



OPTIMICA TESTING TOOLKIT GUI



Platform	
Architecture	32bit WindowsPE
Machine	AMD64
Node	modelon-inc-02
Platform	Windows-7-6.1.7601-SP1
Processor	Intel64 Family 6 Model 58 Stepping 9, GenuineIntel
Python implementation	CPython
Python version	2.7.3

NoShutters						
Test Case	Compilation	Time [s]	Simulation	Time [s]	Verification	Rate
CycleTests.US06	pass	3.3	pass	153.8	pass	100%
CycleTests.IM240	pass	3.3	pass	42.2	pass	100%
CycleTests.EUDC	pass	3.4	pass	76.3	untested	50.0%
CycleTests.JP1015	pass	3.4	pass	84.1	untested	50.0%
Summary	Passed compilation: 4/4		Passed simulation: 4/4		Passed verification: 2/4	

SimpleControl			
Test Case	Compilation	Time [s]	Simulation
UnitTest	pass	0.5	pass
CycleTests.US06	pass	3.5	pass
CycleTests.IM240	pass	3.6	pass
CycleTests.EUDC	pass	3.5	pass
CycleTests.JP1015	pass	3.5	pass
Summary	Passed compilation: 5/5		Passed simul

Report shows summary of results with hyperlinks to detailed reports

TRANSFORMING NATURAL LANGUAGE TO A FORMAL REPRESENTATION

Closing the gaps

MOTIVATION I

Several ways to verify & validate requirements:

- Formal methods: check e.g. consistency of a set of logical requirements
- Simulation: verify that requirements are consistent with physical reality of system
- Both require formalized and executable requirements

MOTIVATION II

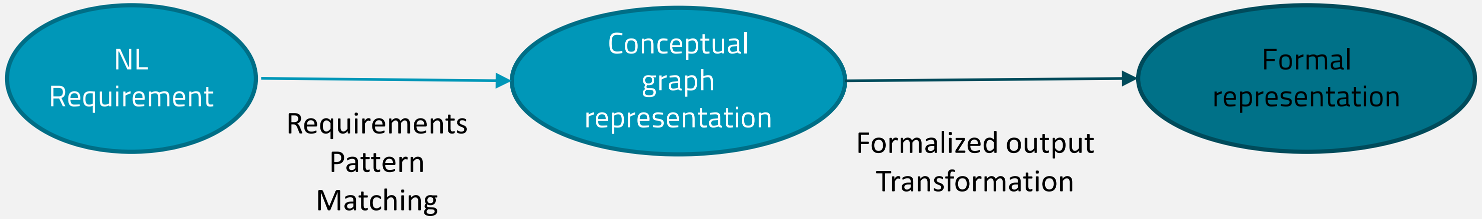
- Need to ensure that the requirements are consistent in terms of time

Req-08: **If** Air Ok signal remains low, auto-control mode is terminated **in 3 seconds**.
Req-17: **When** auto-control mode is entered, **eventually** the cuff will be inflated.
Req-28: **If** a valid pressure is unavailable **in 180 seconds**, manual mode should be triggered.

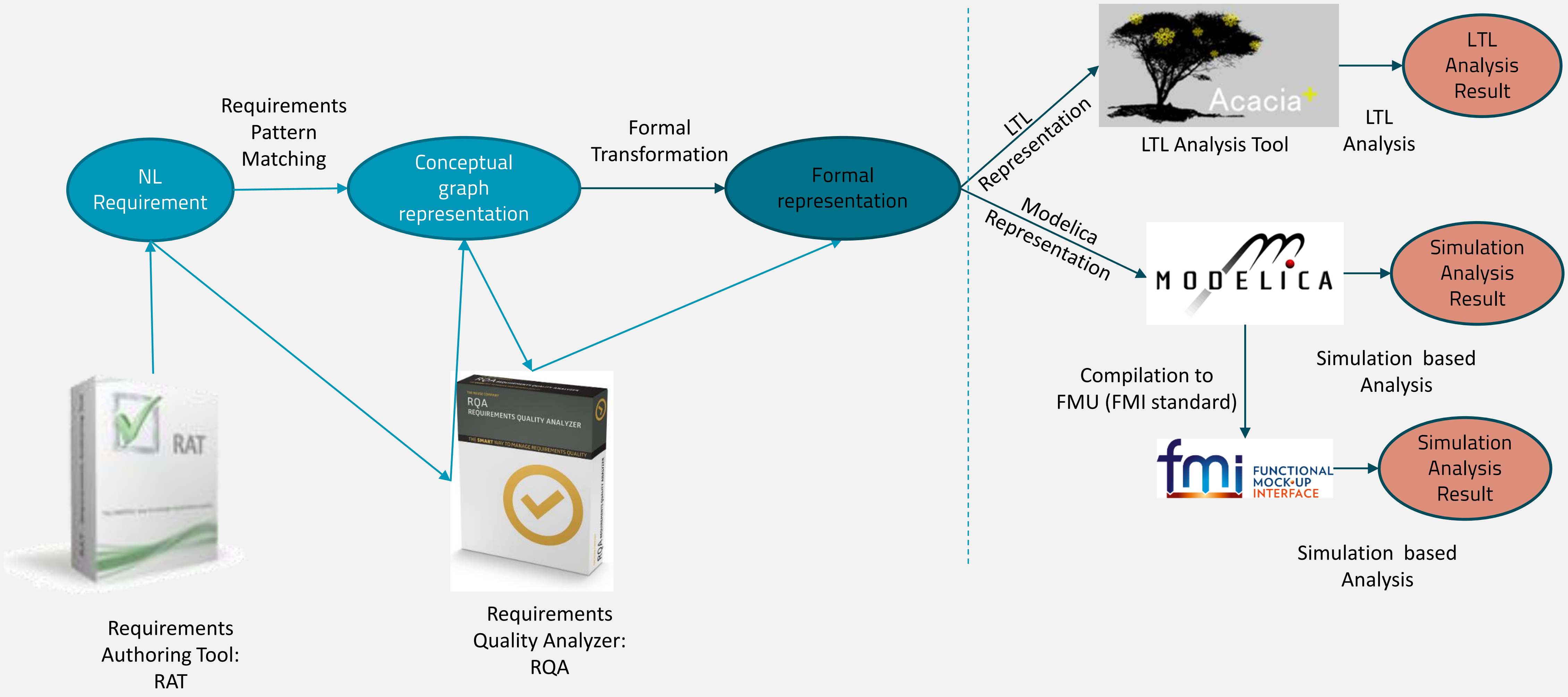
- Proposal:
  - analyze NL requirements,
  - detect temporal elements,
  - formalize them
  - assess temporal quality and show results using a The REUSE Company's RQA Custom-coded metric

Method

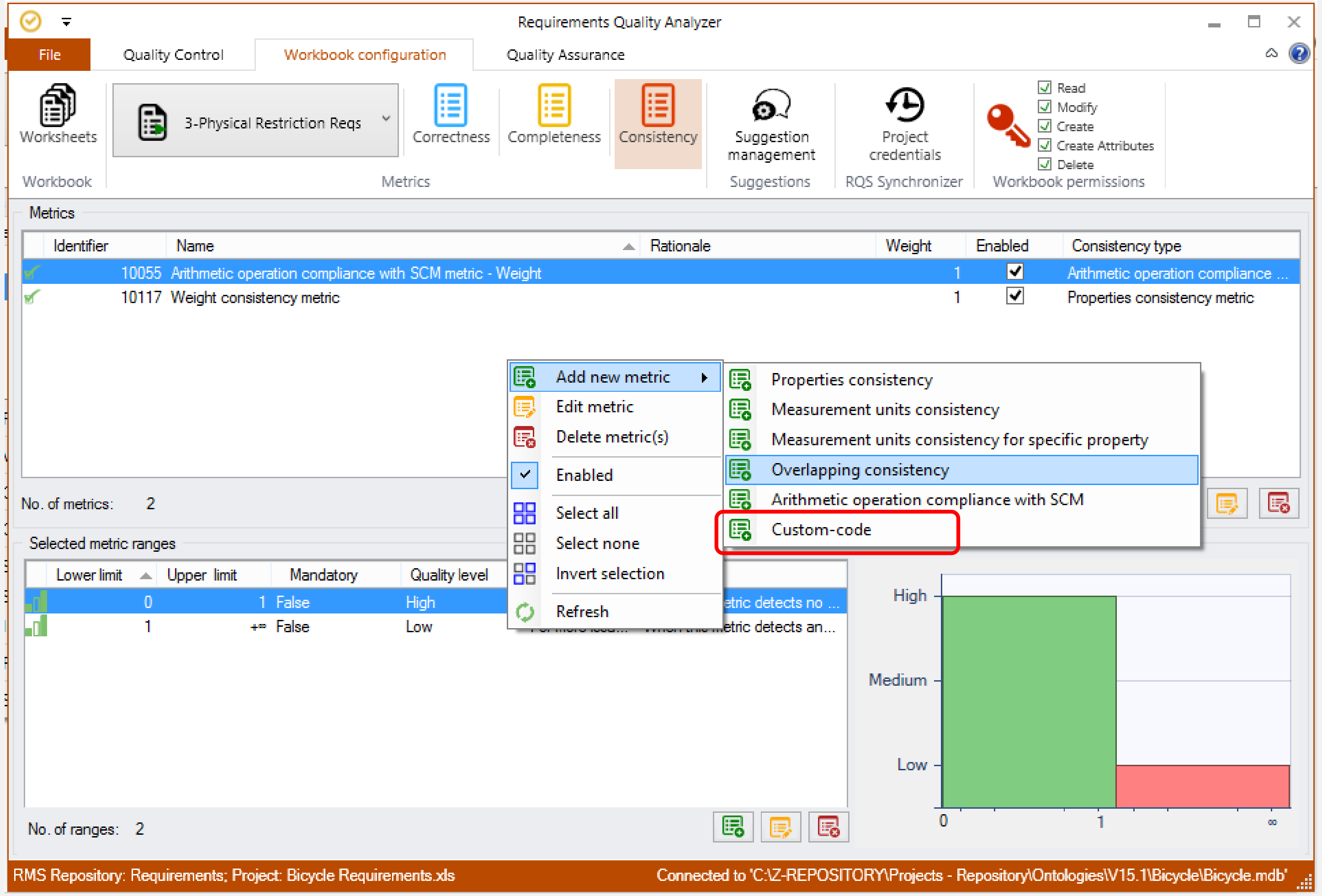
Automatic Translation from Natural Language to Formal representation



Formal Analysis or Simulation based verification



Create a Metric for LTL consistency: Custom Code in RQA



Example

Shared Resource Arbiter

- SRA_2

Client

*When the flying engine activates, the propeller shall be canceled until the **ignition** starts*

*When the aircraft departures, the wheels shall be closed until the **electrical power system** activates*

Mutex

*When **ignition** starts, **electrical power system** shall be stopped*

*When **electrical power system** activates, **ignition** shall be deactivated*

```
G((flying_engine=1) → X((propeller=0)U(ignition=1)));
G((aircraft=1) → X((wheel=0)U(electrical_power_system=1)));
G((ignition=0) + (electrical_power_system=0));
```

Shared Resource Arbiter

- SRA_3

Client
- *When the flying engine activates, the propeller shall be canceled until the* ***ignition*** *starts*
- *When the aircraft launches, the wheels shall be closed until the* ***electrical power system*** *activates*
- *When the navigation system starts, the control mode shall be stopped until the* ***gearshift*** *enables*

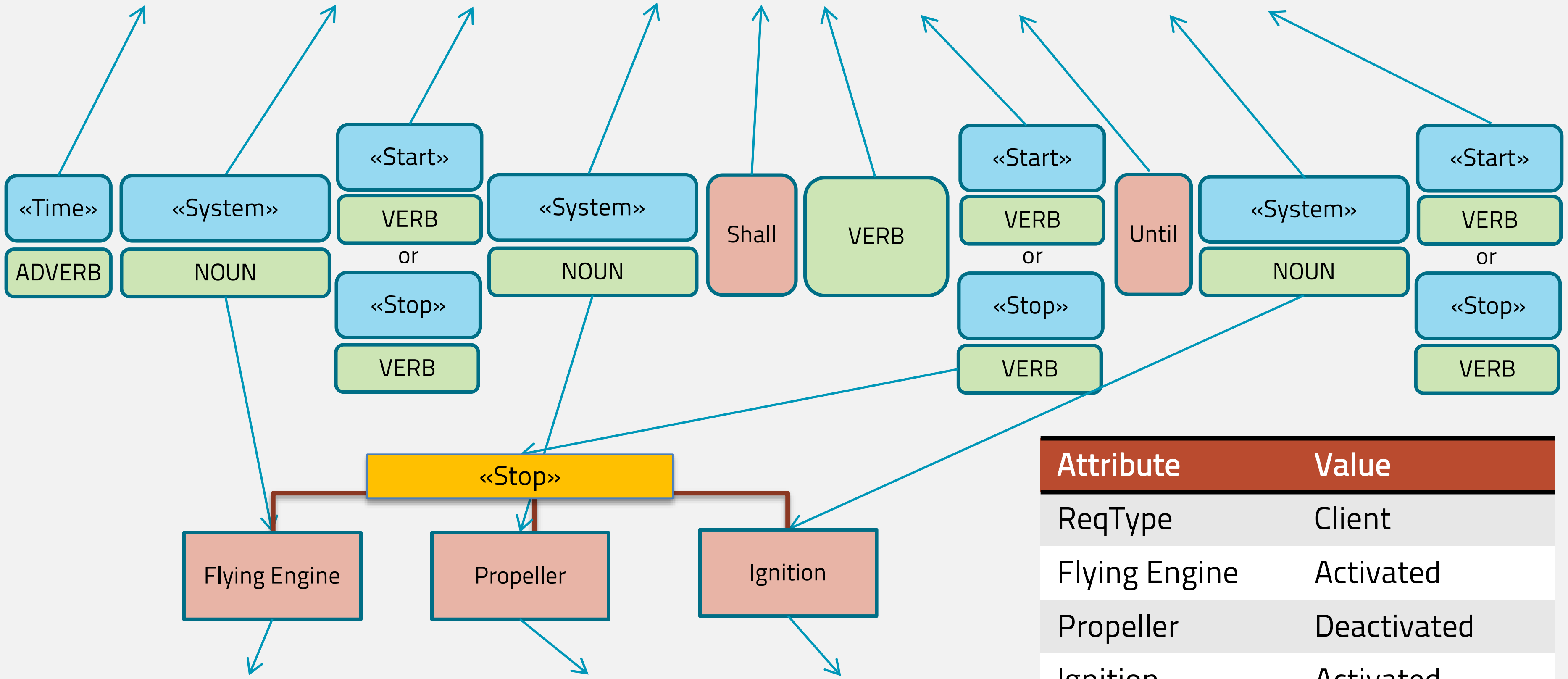
Mutex
- *When* ***ignition*** *starts,* ***electrical power system*** *and* ***gearshift*** *shall be stopped*
- *When* ***electrical power system*** *activates,* ***ignition*** *and* ***gearshift*** *shall be deactivated*
- *When* ***gearshift*** *begins,* ***ignition*** *and* ***electrical power system*** *shall be terminated*

```
G((flying_engine=1) → X((propeller=0)U(ignition=1)));
G((aircraft=1) → X((wheel=0)U(electrical_power_system=1)));
G((navigation_system=1) → X((auto_control_mode=0)U(gearshift=1)));
G(((electrical_power_system=0) * (gearshift=0)) +
  ((ignition=0) * (gearshift=0)) +
  ((ignition=0) * (electrical_power_system=0)));
```

Ontology Building

Pattern matching and Formalization

When the flying engine activates, the propeller shall be canceled until the ignition starts

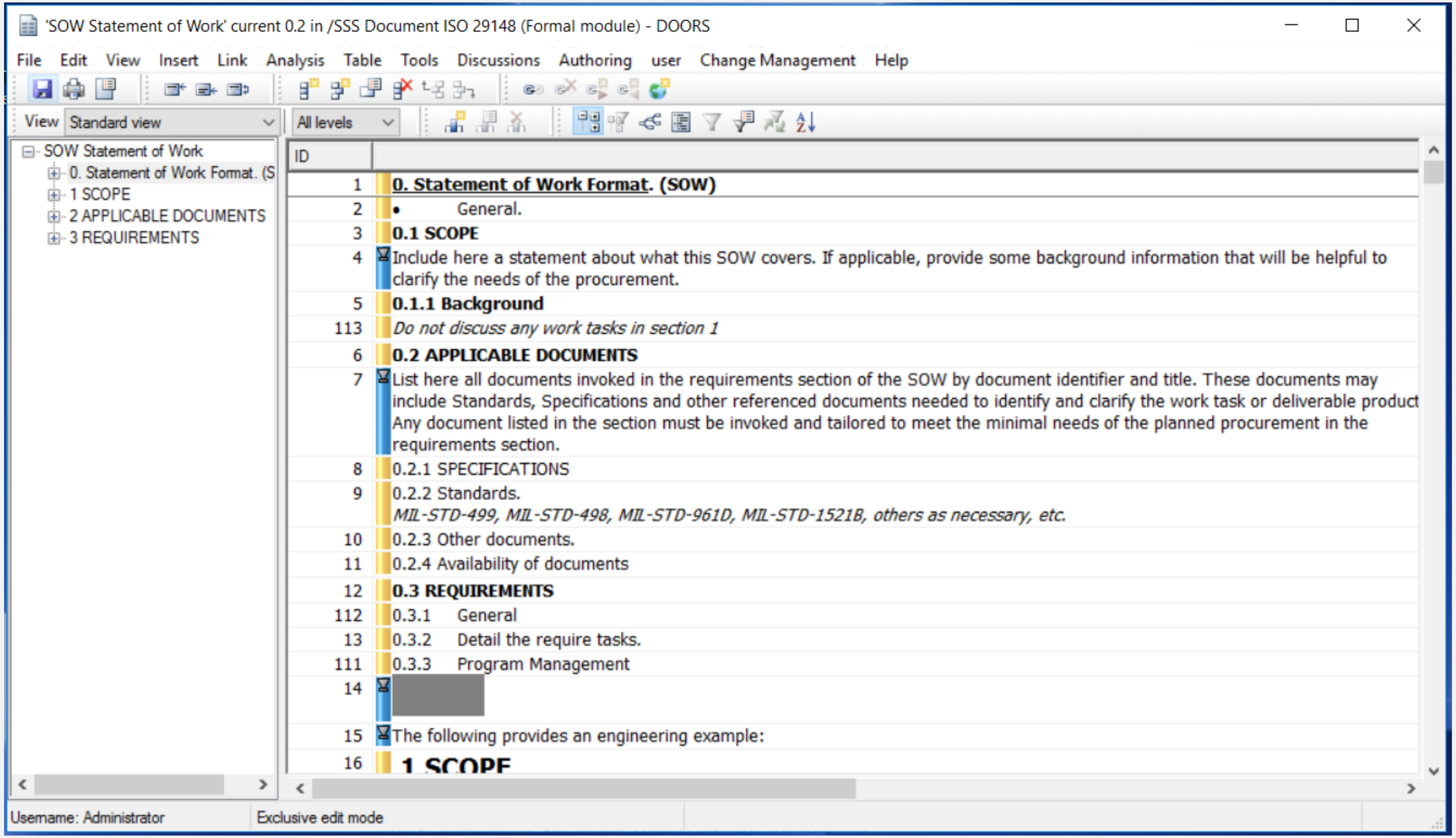


G((flying_engine=1) → X((propeller=0)U(ignition=1)));

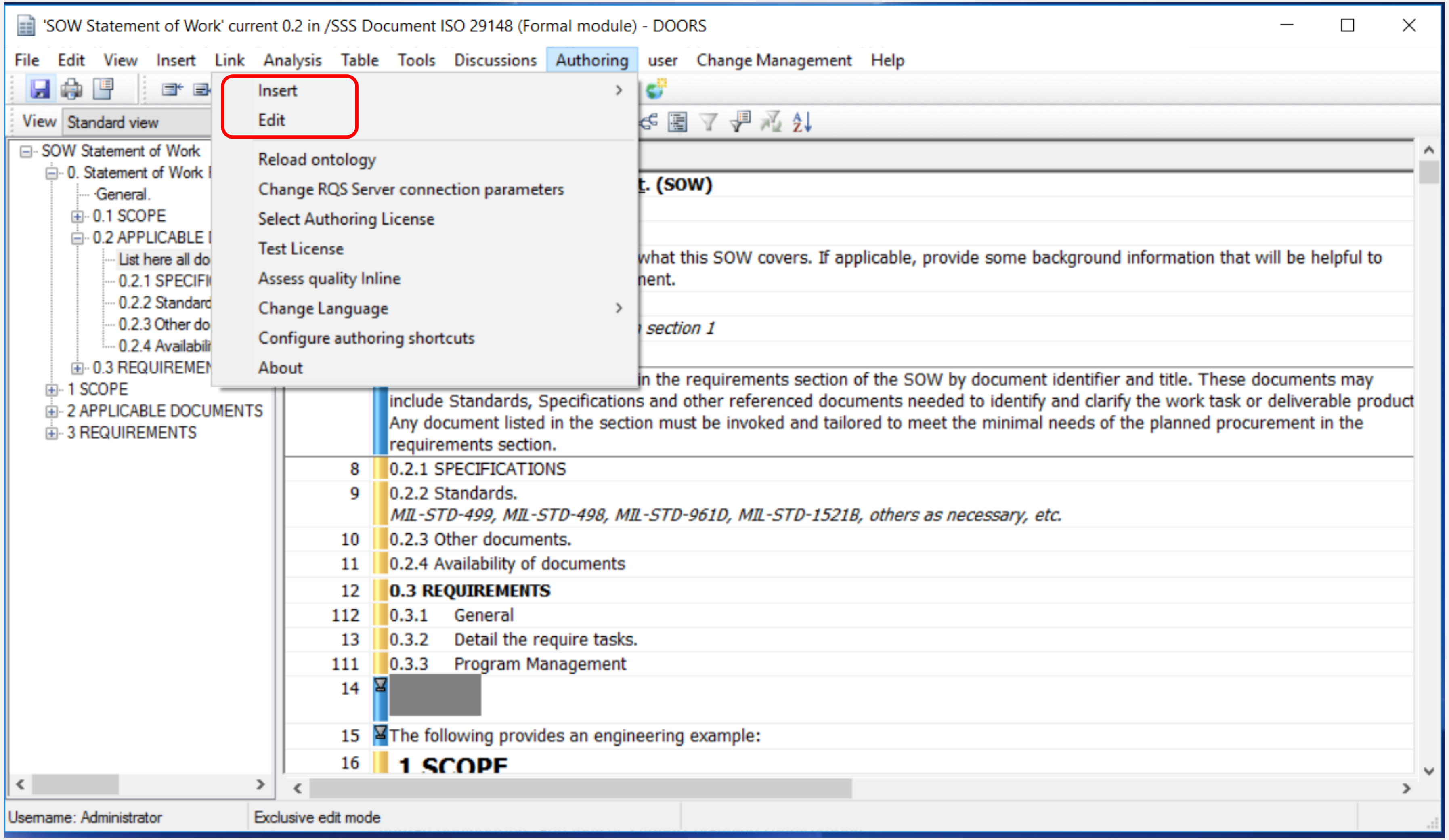
Attribute	Value
ReqType	Client
Flying Engine	Activated
Propeller	Deactivated
Ignition	Activated

RAT overview

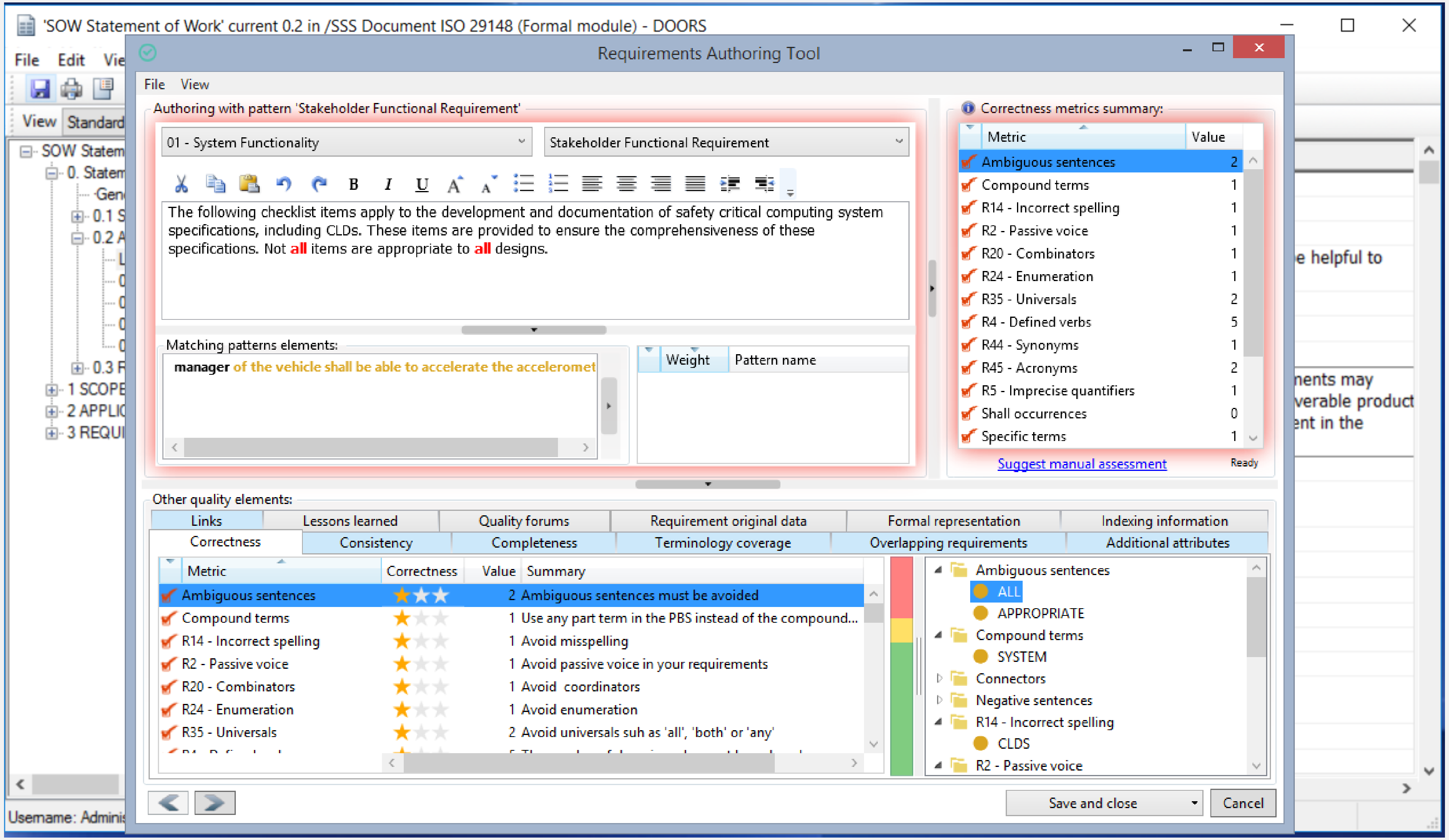
Plug-in for IBM rational DOORS



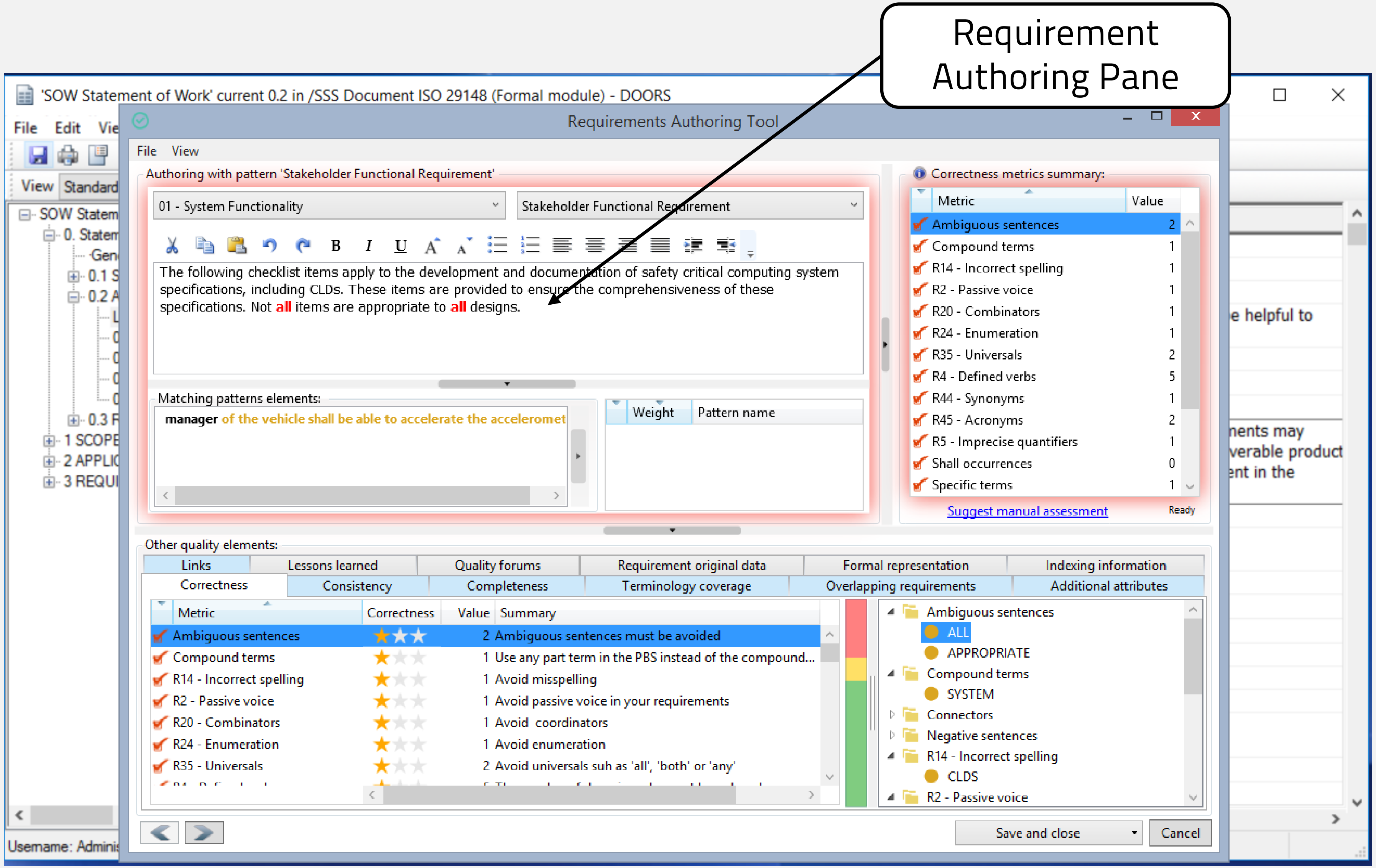
Allows Requirements Authoring



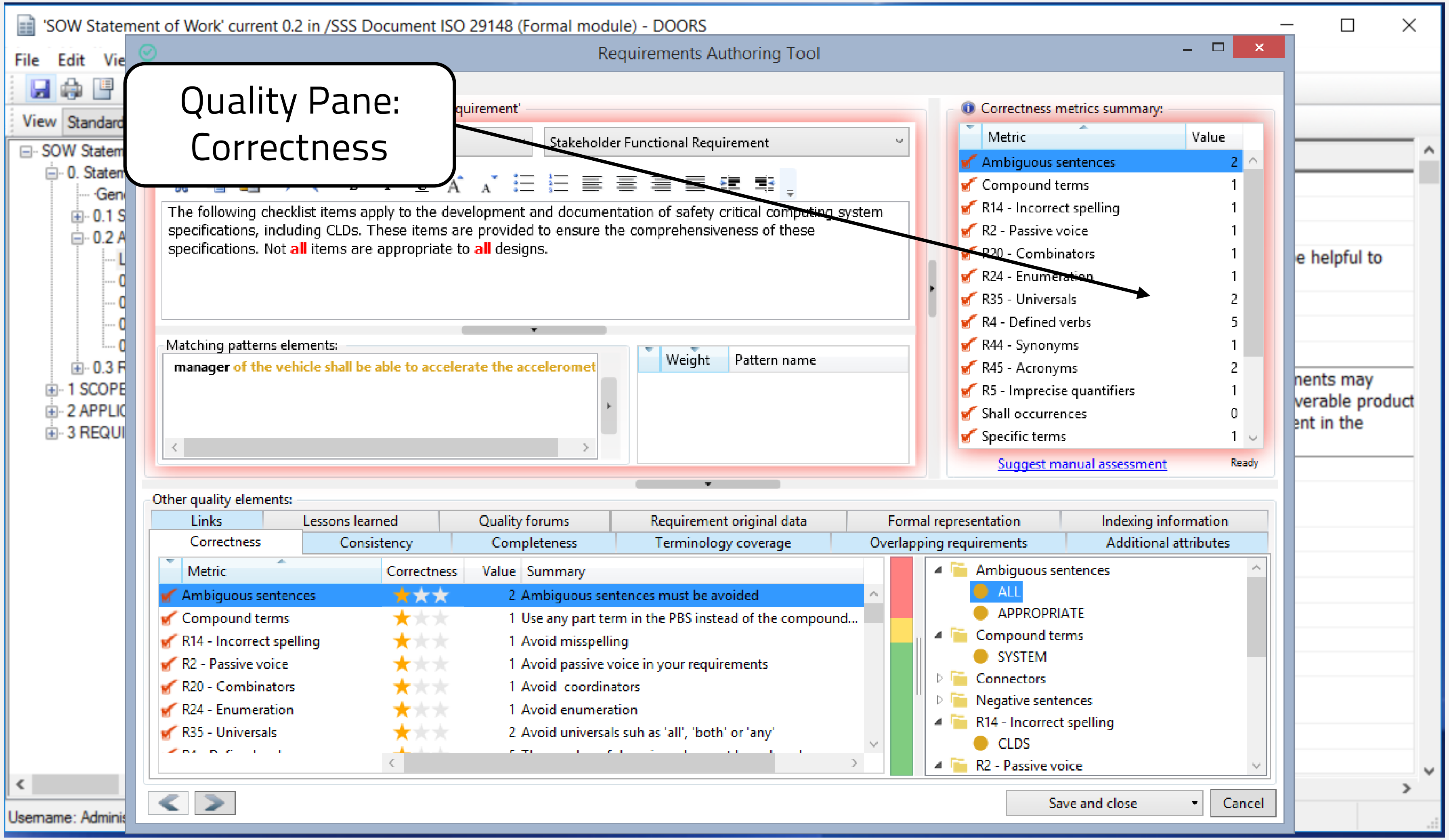
RAT Plug-in running on top of DOORS



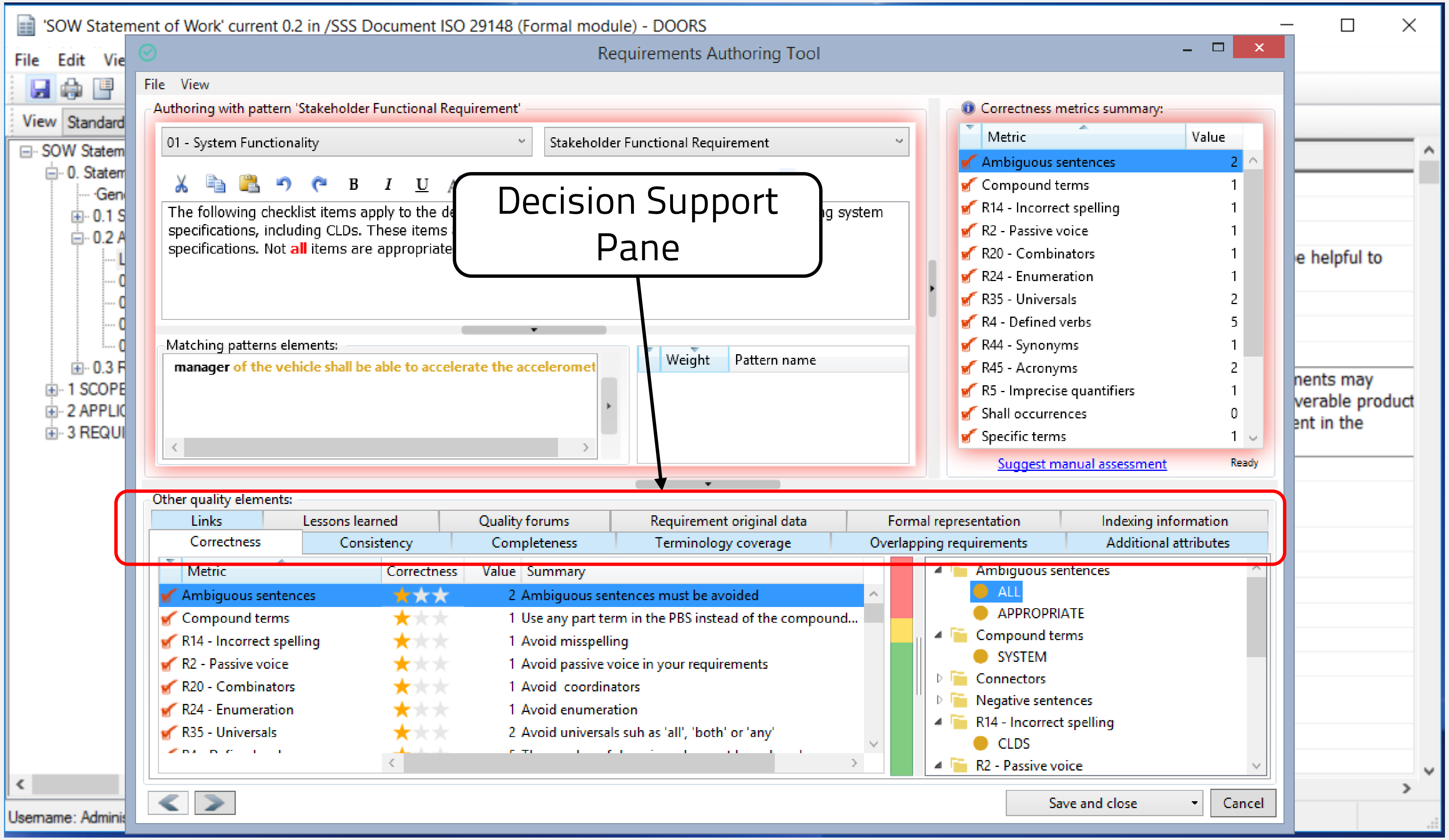
RAT Plug-in running on top of DOORS



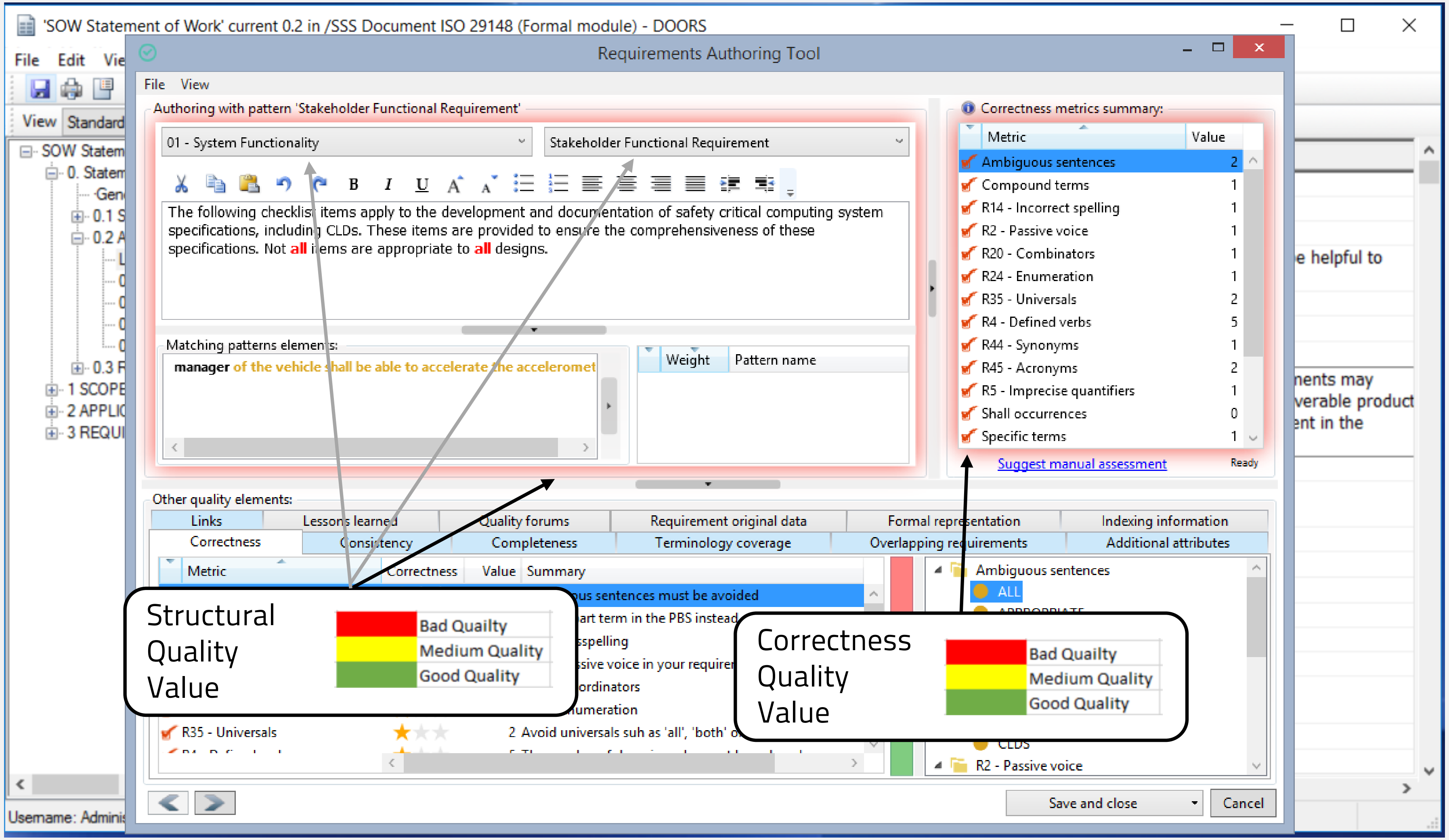
RAT Plug-in running on top of DOORS



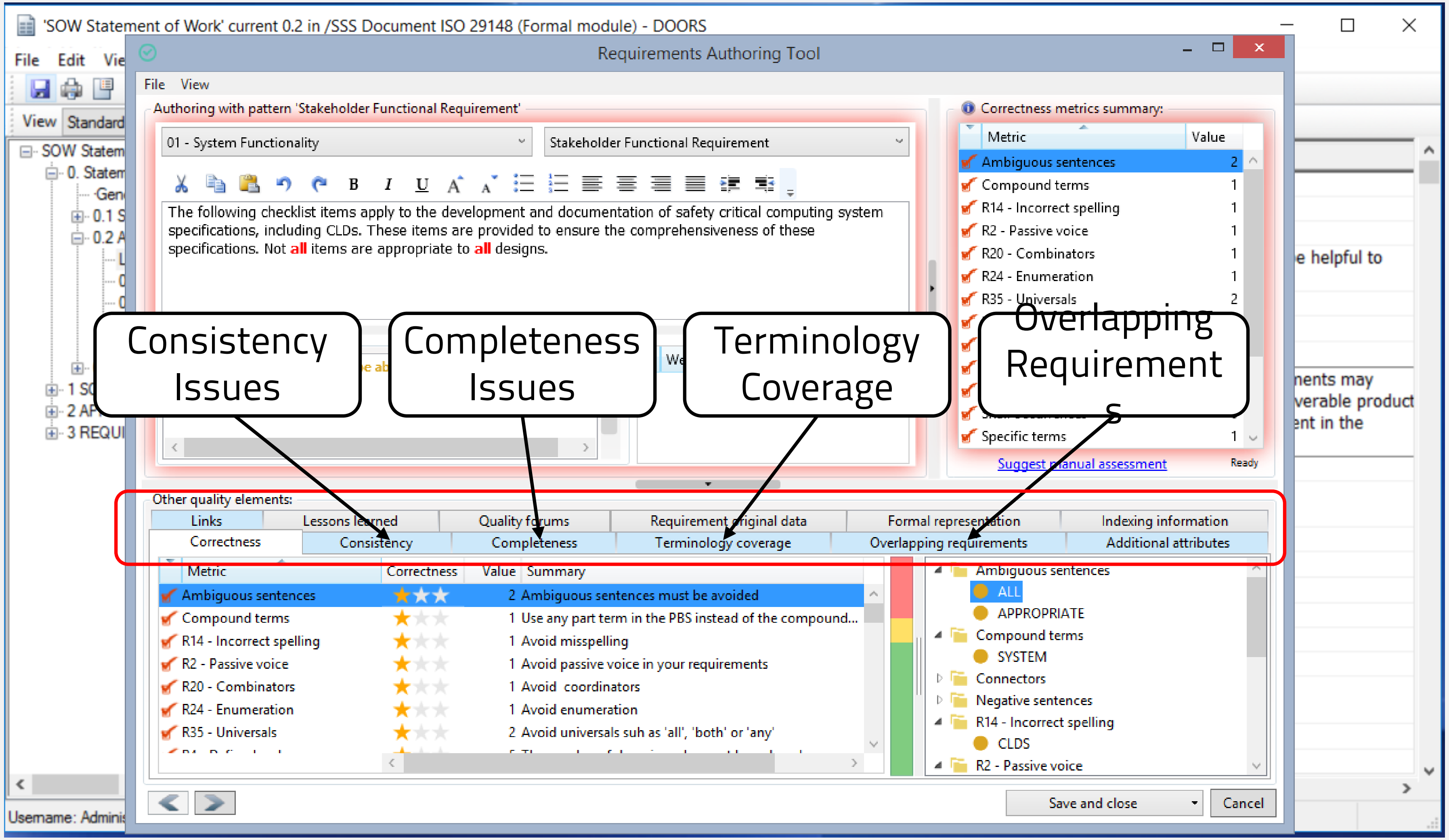
RAT Plug-in running on top of DOORS



RAT Plug-in running on top of DOORS



RAT Plug-in running on top of DOORS



WHERE DOES THIS LEAVE US OVER ALL?

- We have a vision of an integrated process and tool landscape to bring together Systems Engineering and Model Based Design
- A few good things can be done today:
  - The RAT allows to write high quality requirements, integrated into requirements management
  - We can use Modelica to make requirements executable
  - We can give the requirements to design engineers and enable automated requirements verification with Optimica Testing Tools
  - We can transform natural language requirements to a formal representation for formal or simulation based verification
- There are still many missing links to fill the gaps !

CALL TO ACTION

- We are looking for other systems engineering users that support the same vision
- We are looking for more tool vendors on the systems engineering and modeling and simulation side that share our vision
- We strongly believe in open standards to connect SE & MBD
- Let's work together to make this a reality:
  We need better tool integration to enable engineers to design complex systems !