

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

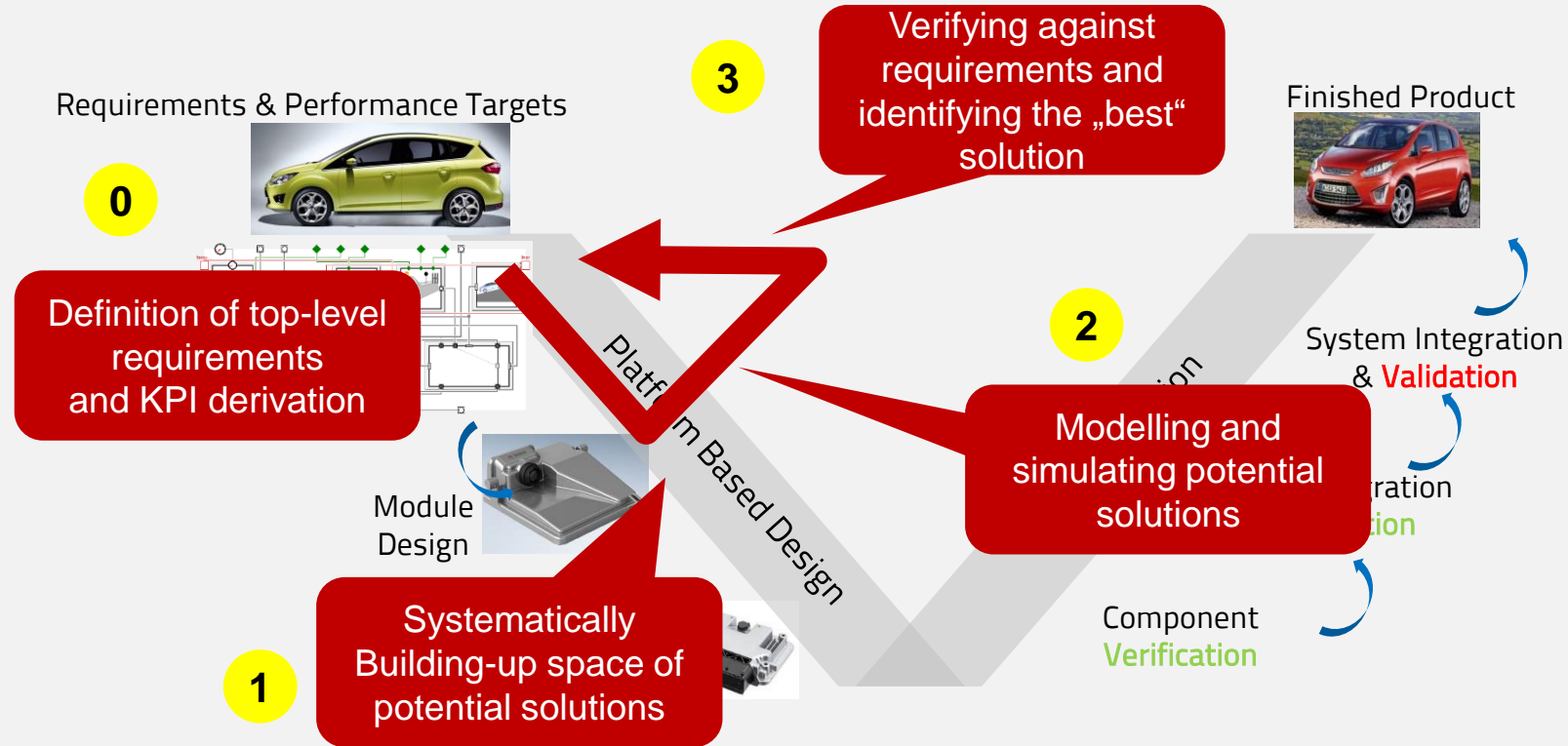
SYSTEMS ENGINEERING AND MODEL BASED DESIGN

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

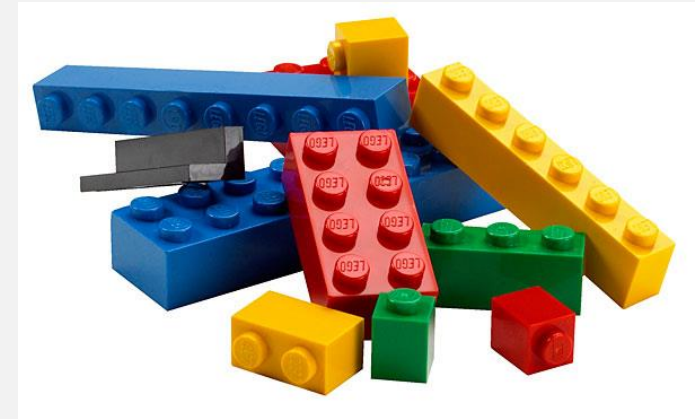


Many industries do this all the time, but the tools are not integrated!

MODELICA: THE OPEN STANDARDS SYSTEM LANGUAGE

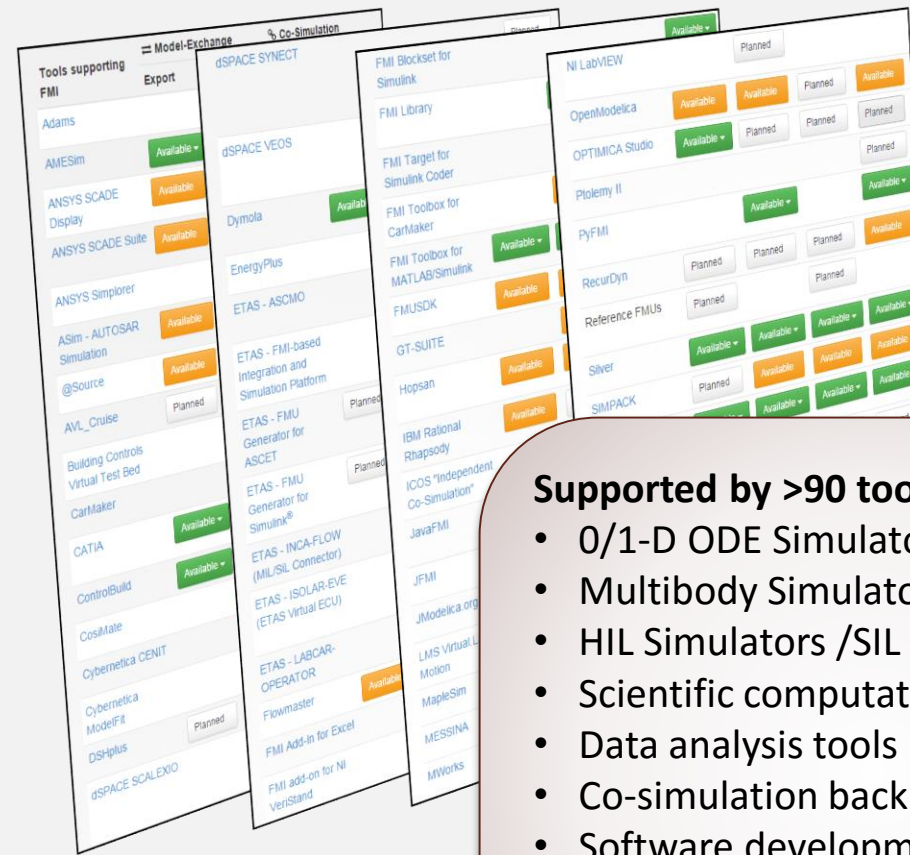
Modelica® is a non-proprietary, object-oriented, equation based language to conveniently model complex physical systems containing, e.g., mechanical, electrical, electronic, hydraulic, thermal, control, electric power or process-oriented subcomponents

- 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.

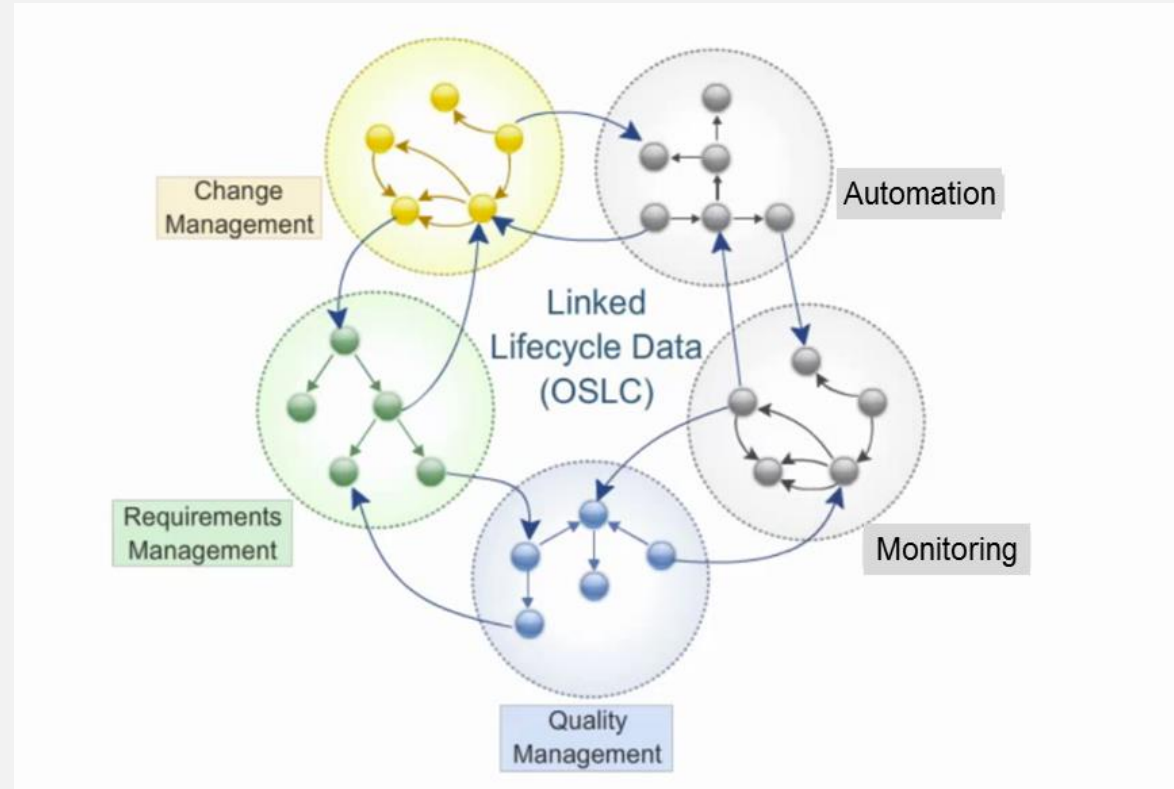


Supported by >90 tools:

- 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)

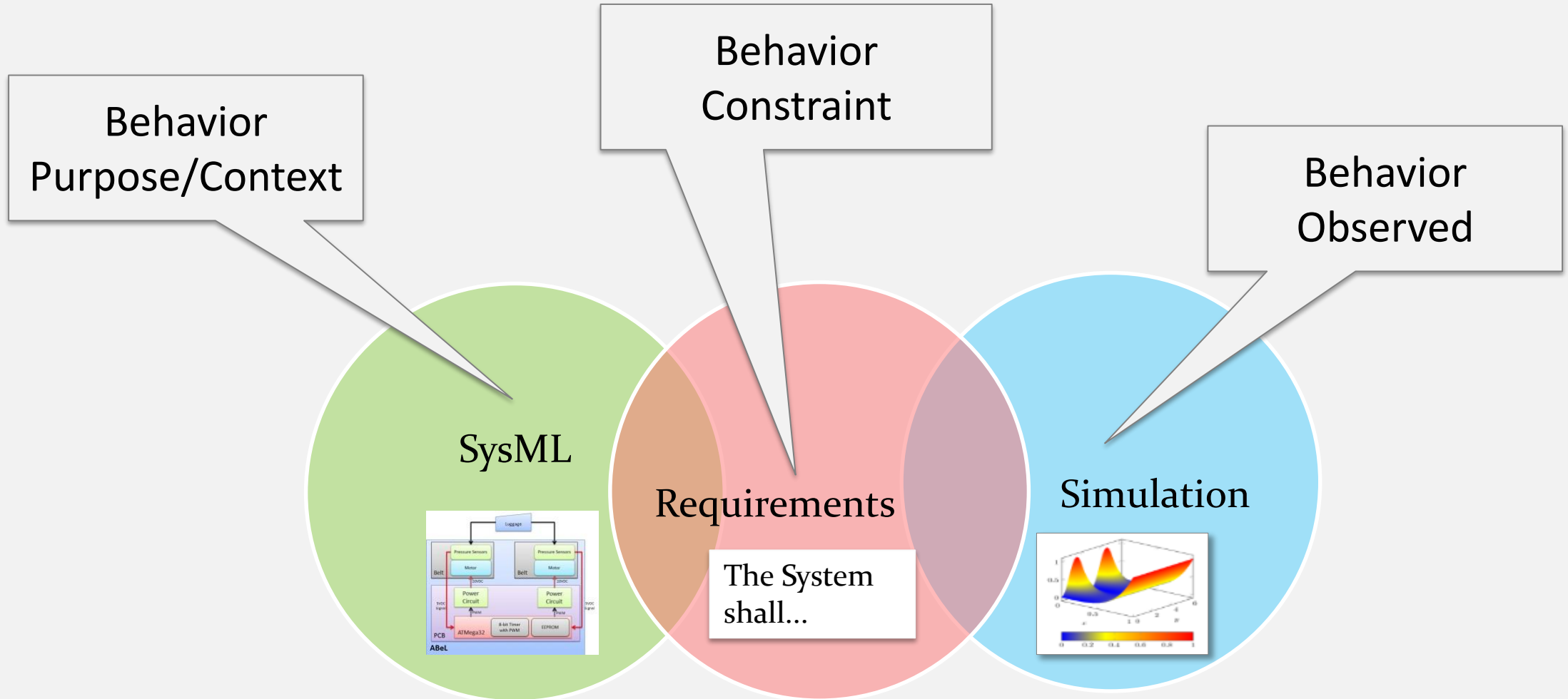
- OSLC = reusing web standards for tool integration
- Based on Web standards **linked data** and **RESTful Web services**
- Create specifications for interactions between tools
- Initiated by IBM, now managed by OASIS
- Focus on software-and systems engineering
- Not much traction (yet) with M&S tools



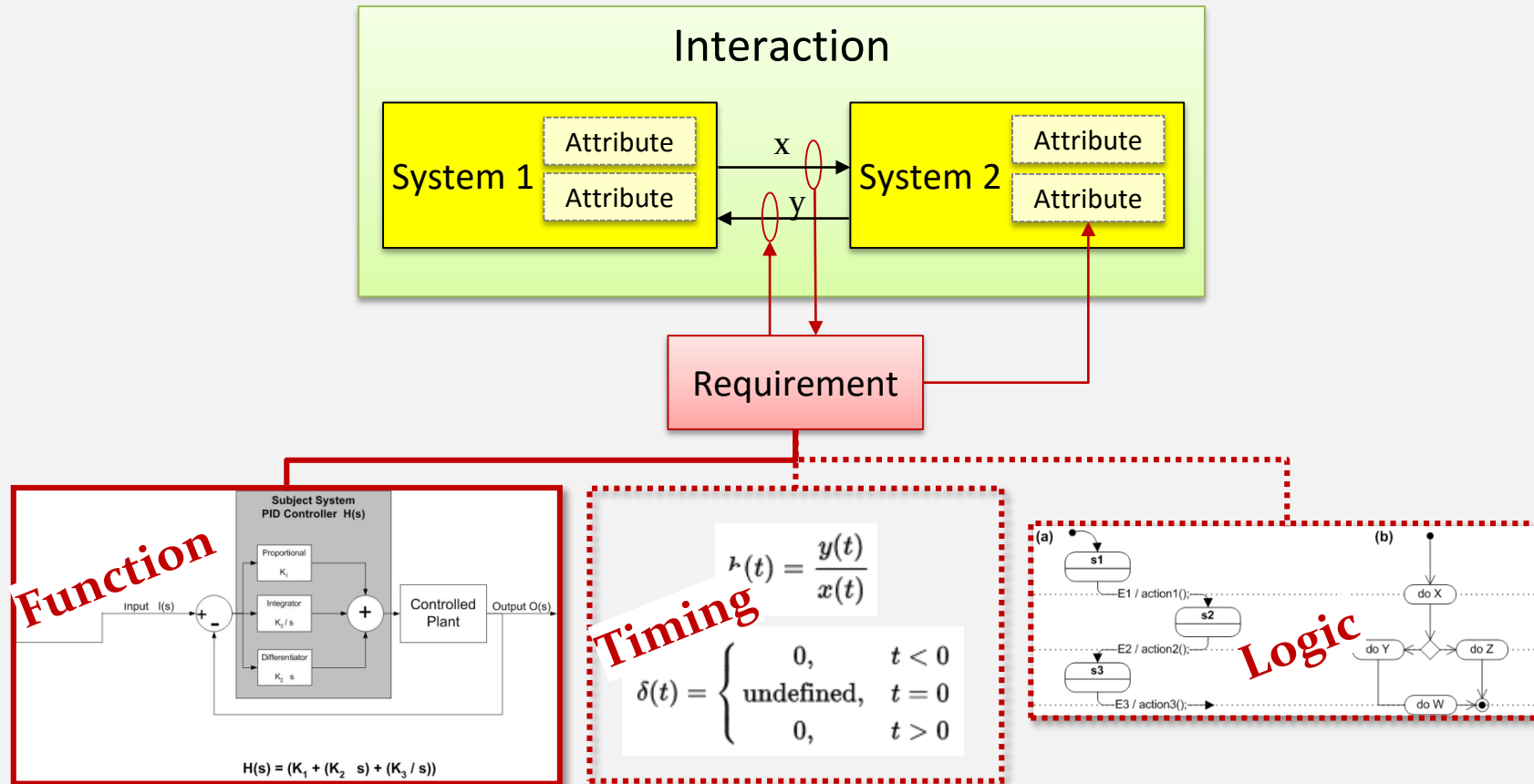
We built an open-source OSLC-to-FMI connector to link simulation results and parameters to life cycle tools

AN IDEAL PROCESS TO INTEGRATE SYSTEMS ENGINEERING WITH MODEL BASED DESIGN

Semantic Integration

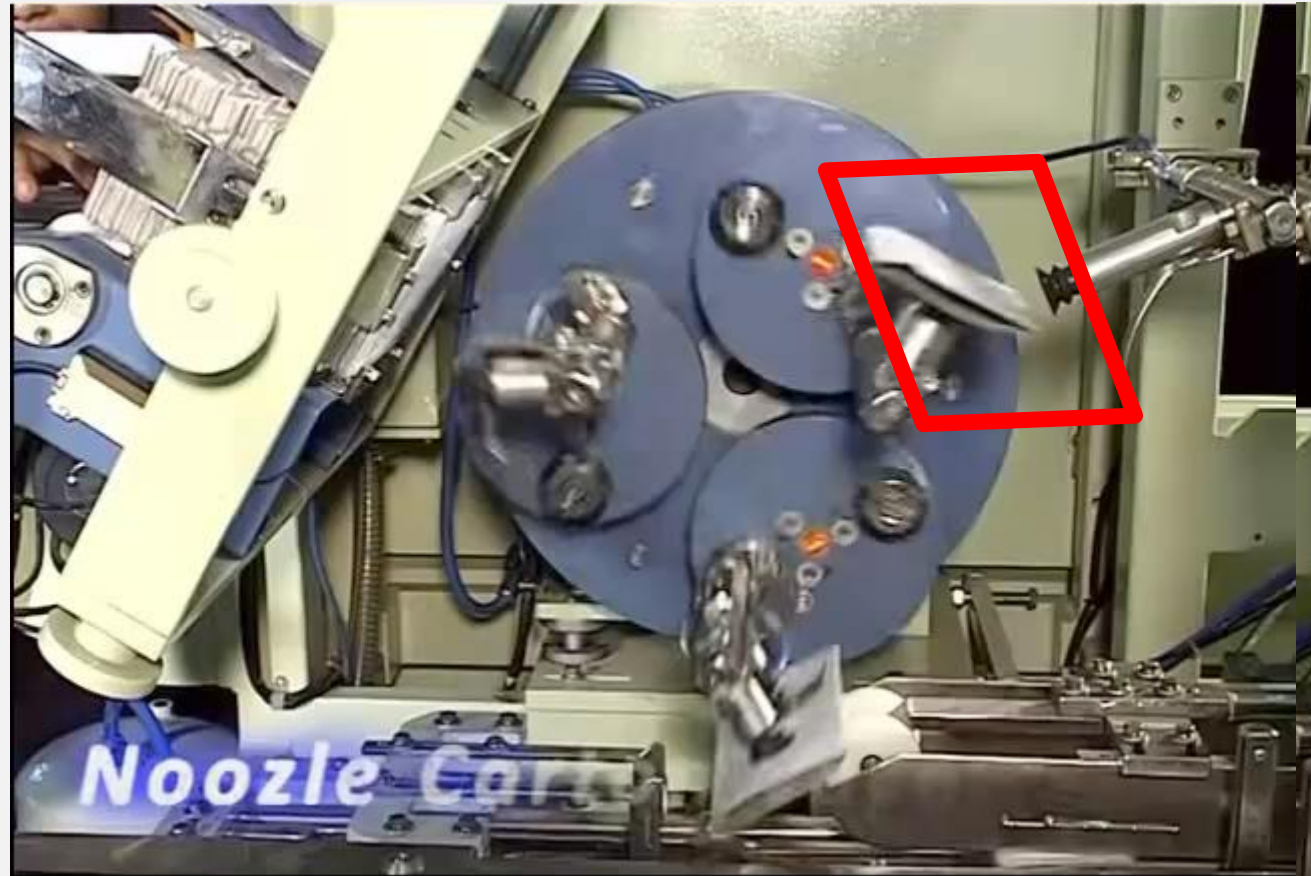


Purpose, Context & Anatomy of a Requirement

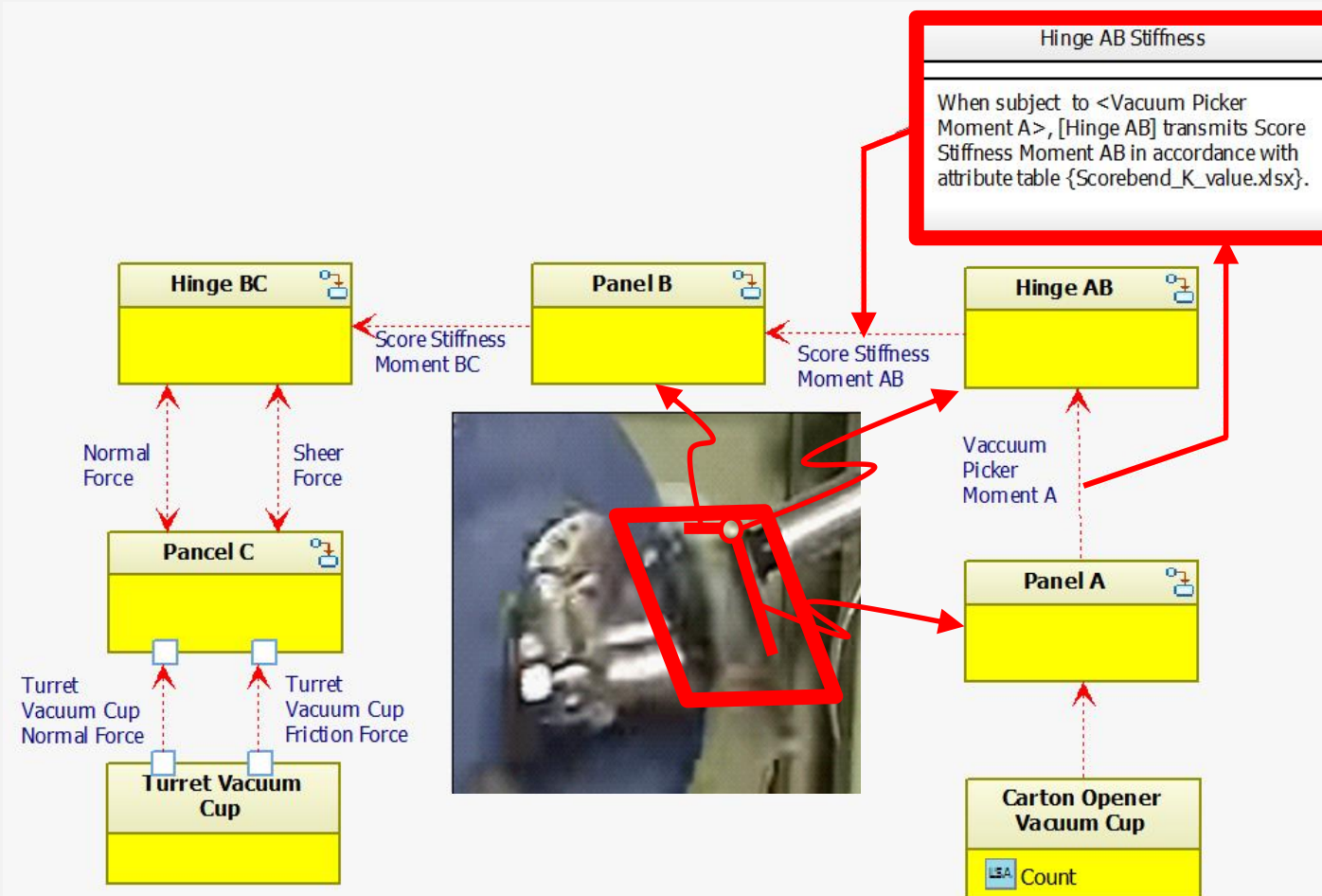


Bill Schindel (of ICTT):
"Requirements are Transfer Functions"

Example System



Example Requirement (Transfer Function)



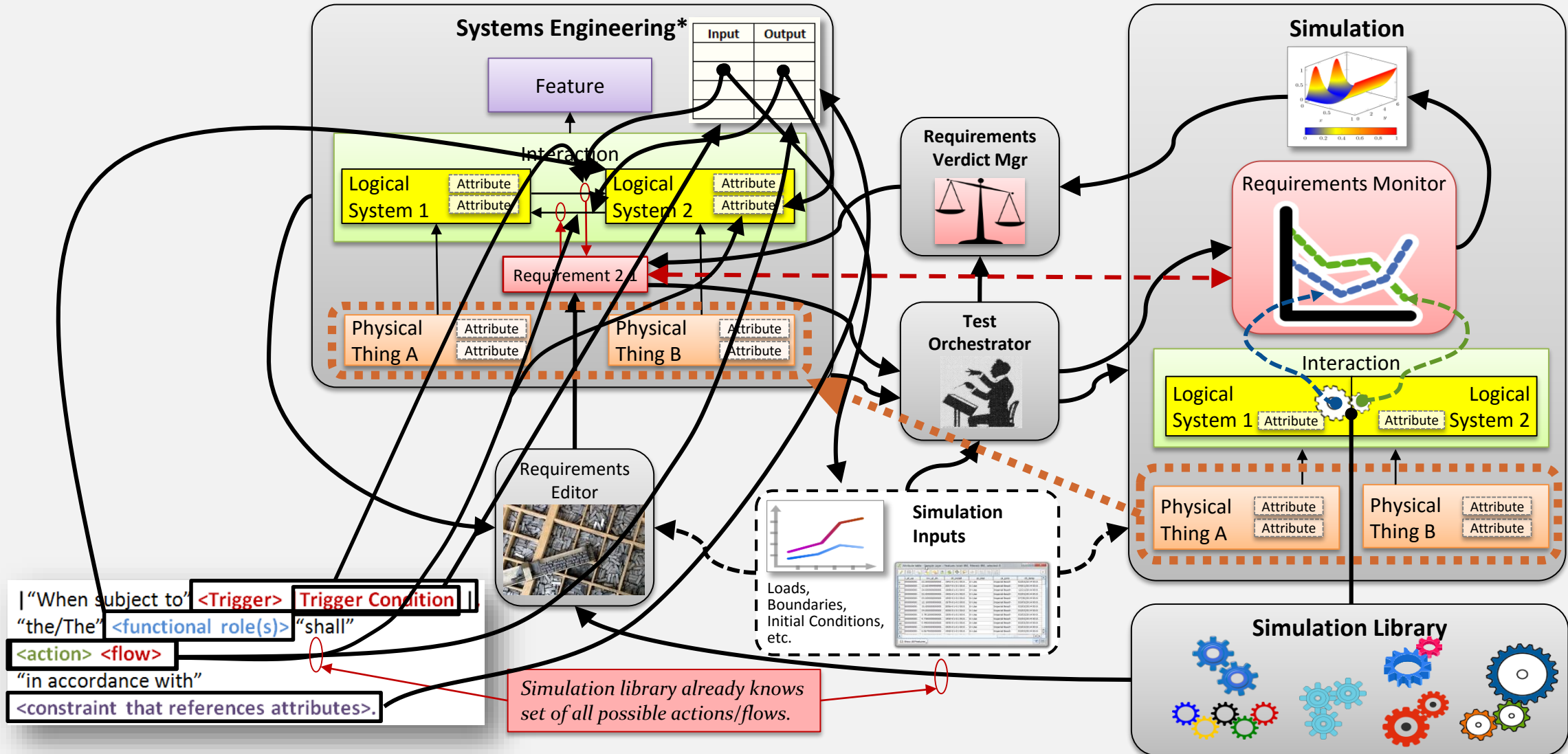
Example Requirement (Transfer Function)

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>.

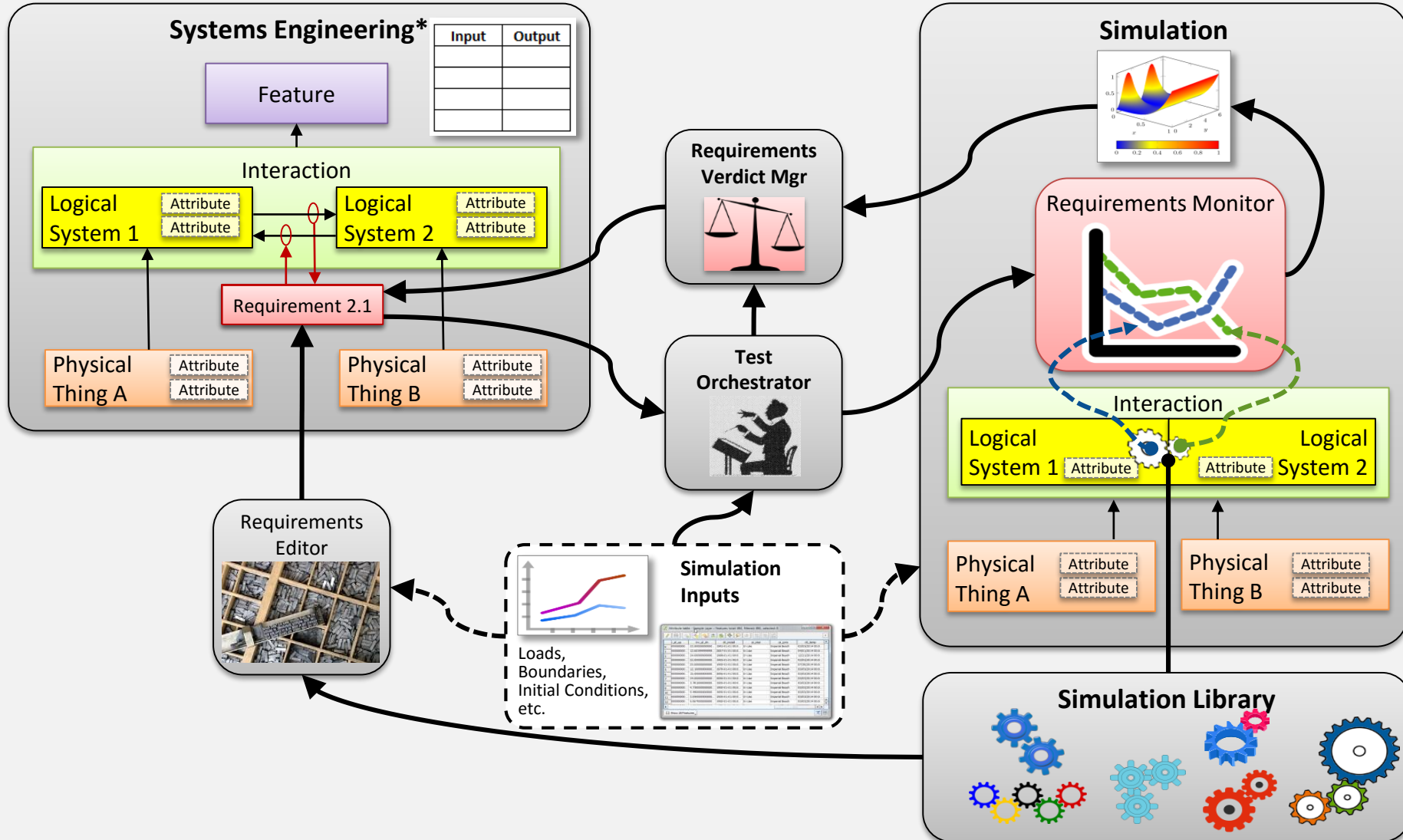
??? Machine readable requirements statement **???**

Prospective SE and M&S Integration Strategy



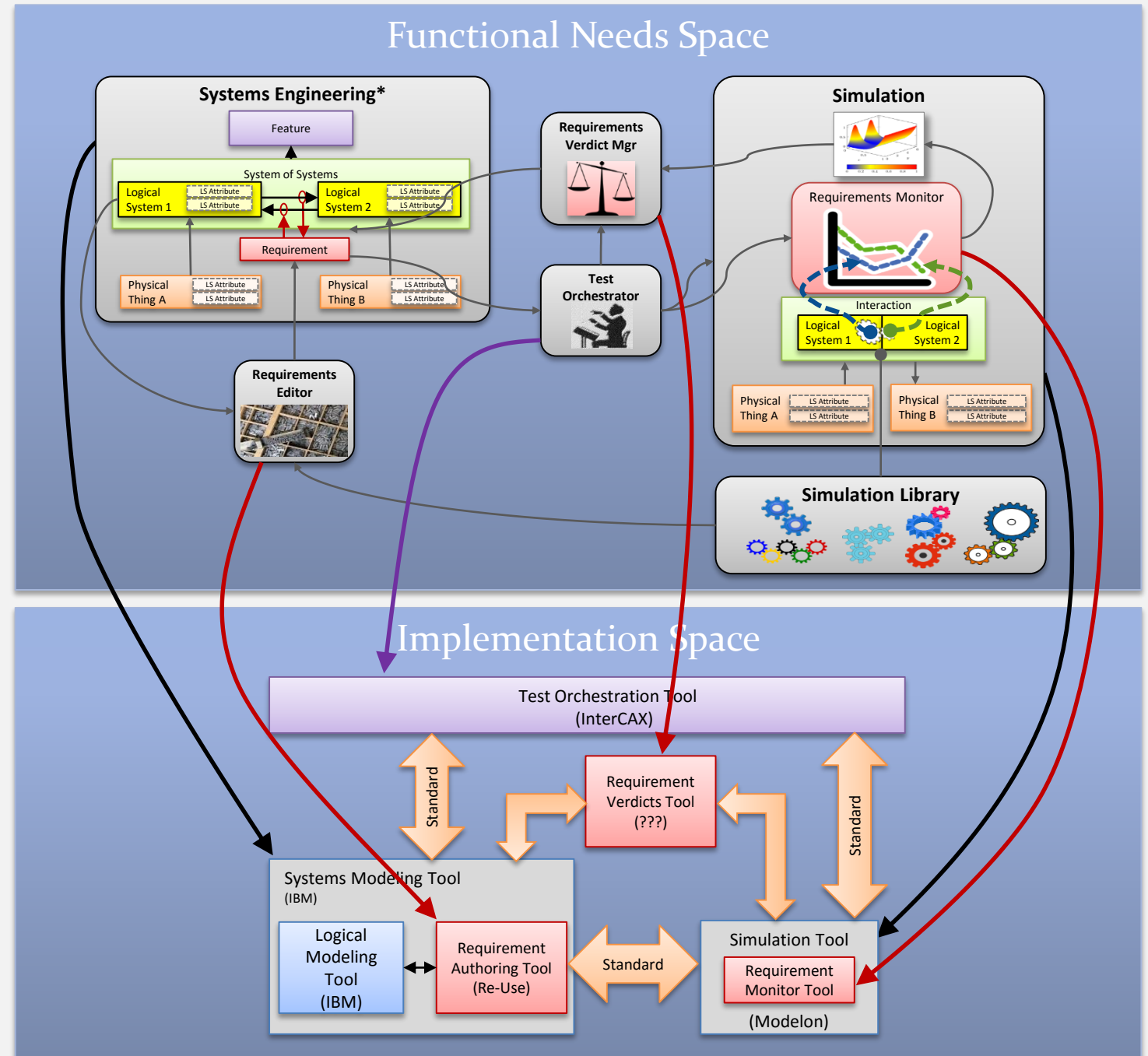
*The "Systems Engineering" metamodel is a representation of Bill Schindel's "Systematica" method.

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

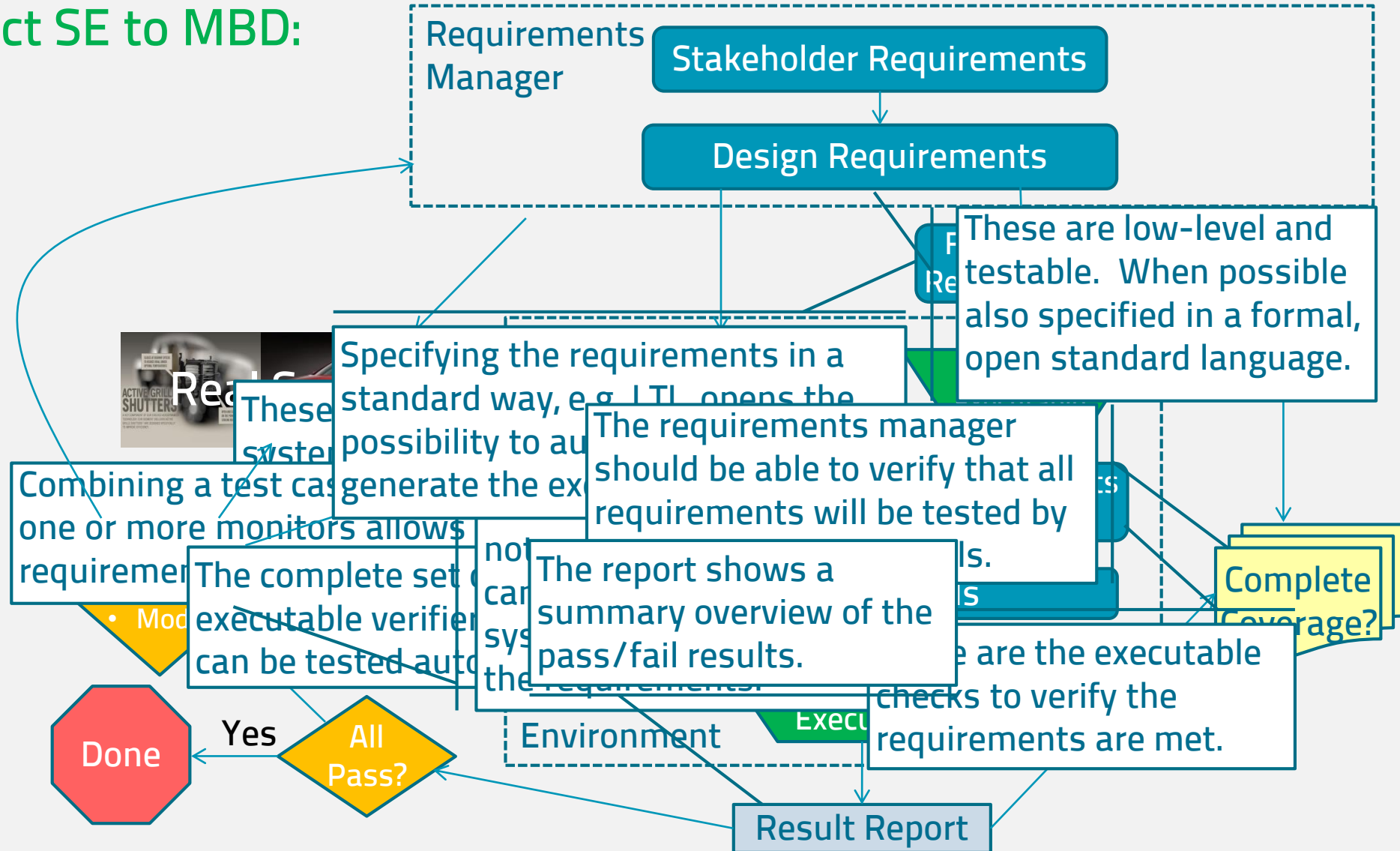


EXECUTABLE REQUIREMENTS

Continuous feedback on compliance of requirements

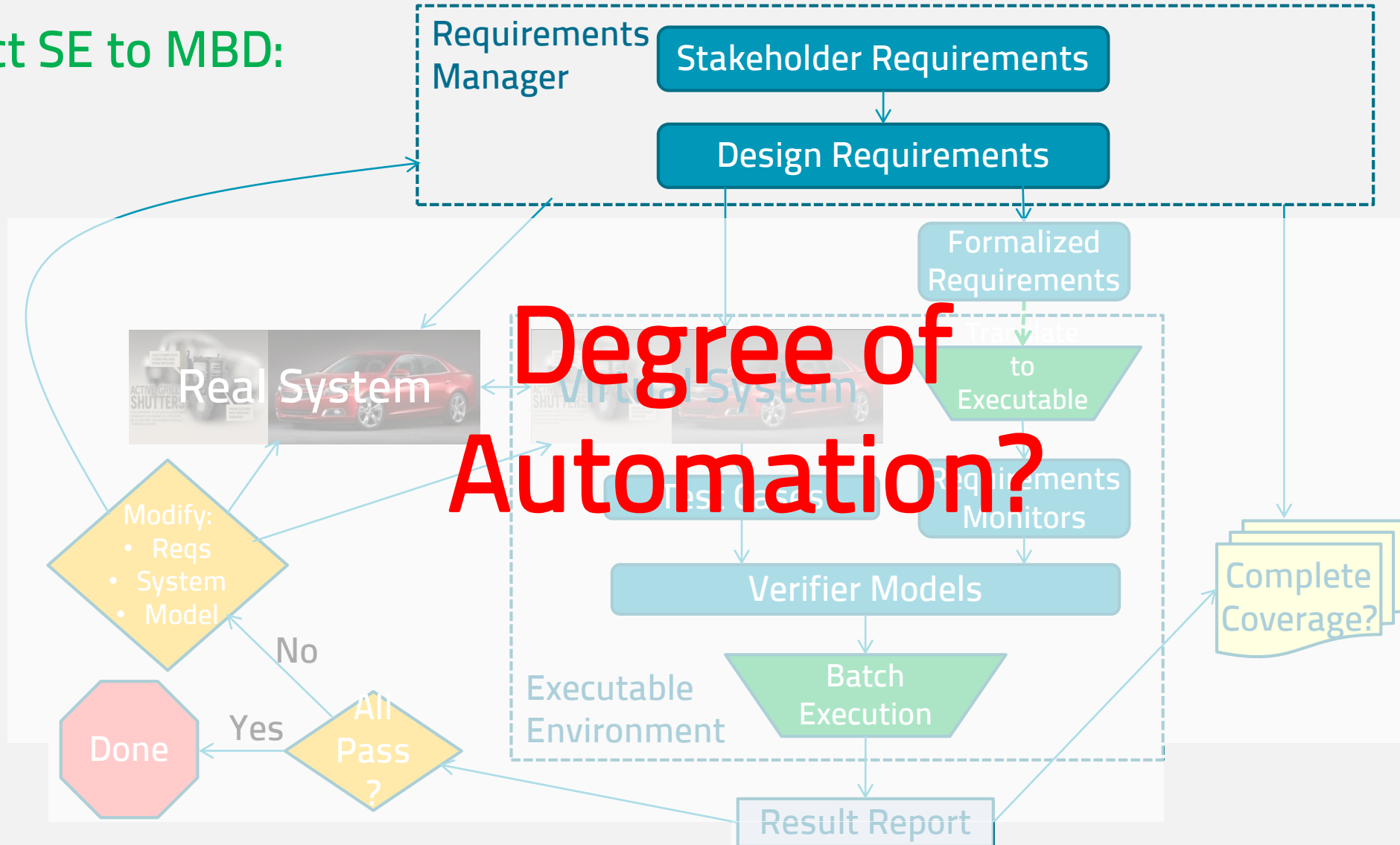
IN-THE-LOOP REQUIREMENTS VERIFICATION

Connect SE to MBD:



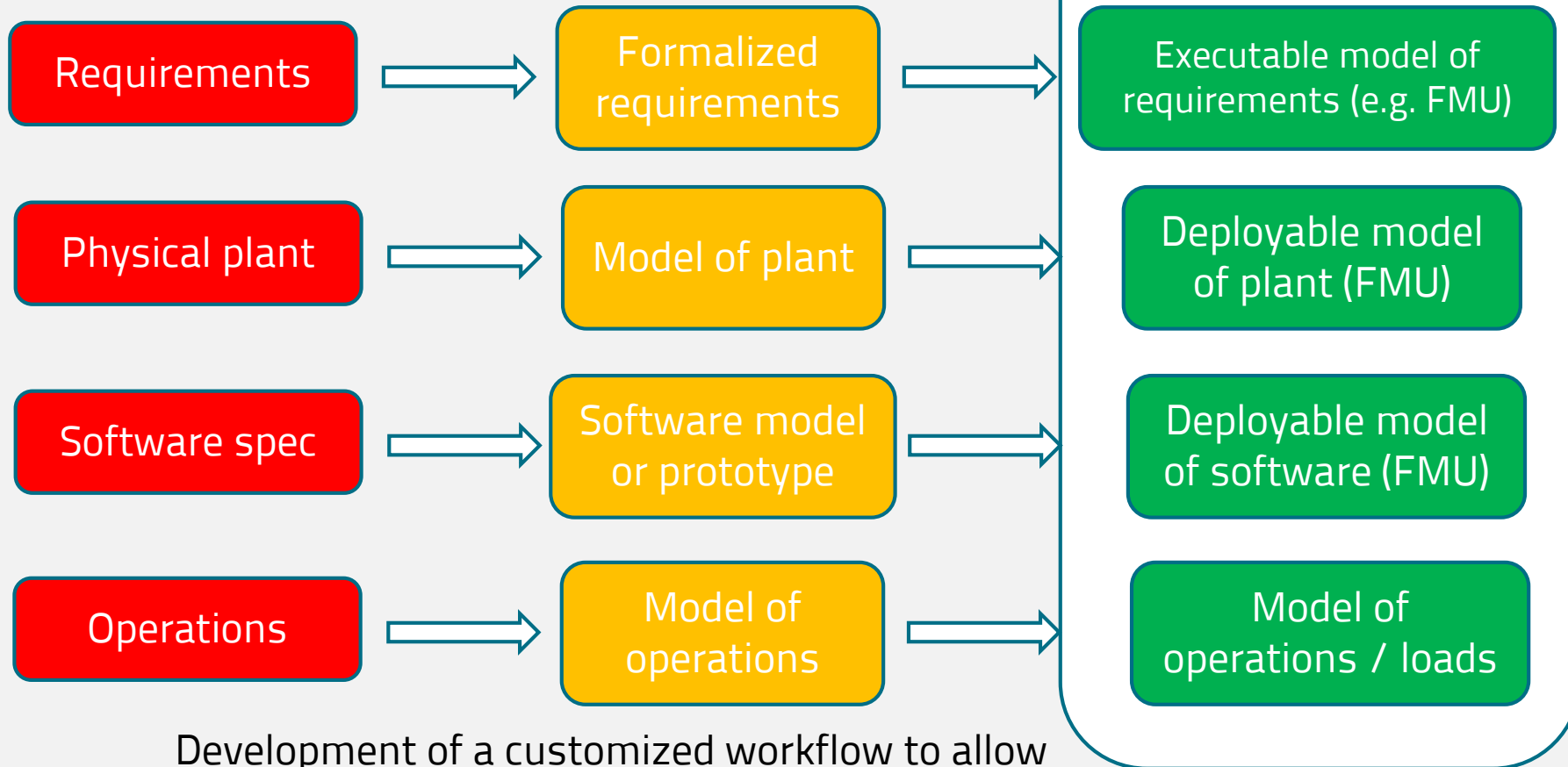
AUTOMATED REQUIREMENTS VERIFICATION

Connect SE to MBD:



AUTOMATED REQUIREMENTS VERIFICATION

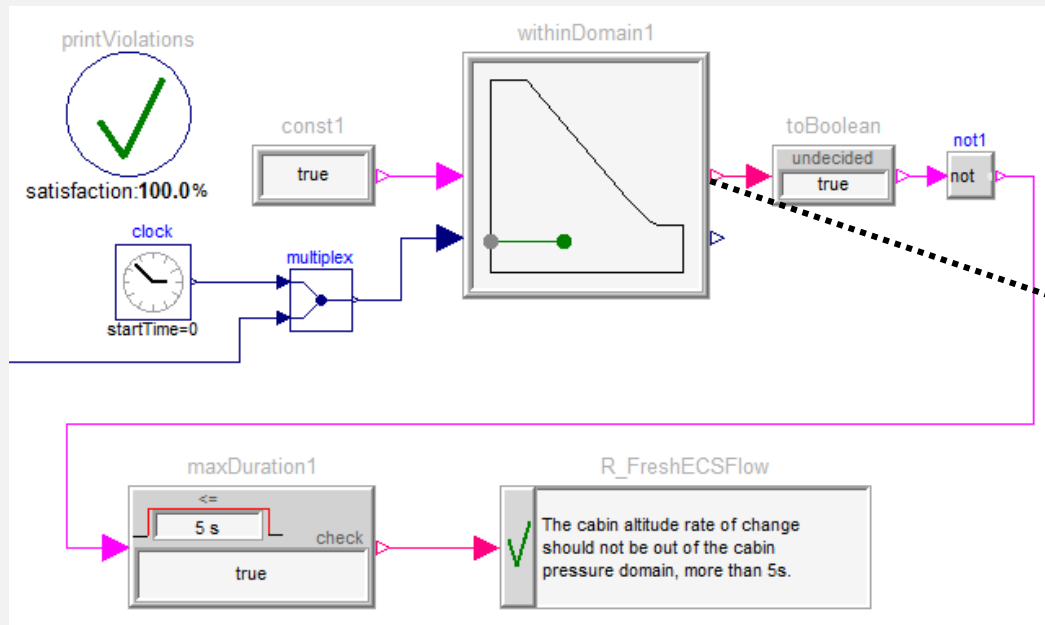
- Systems Engineering centric FMI-based workflow example: automated requirements verification for hardware and software requirements



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

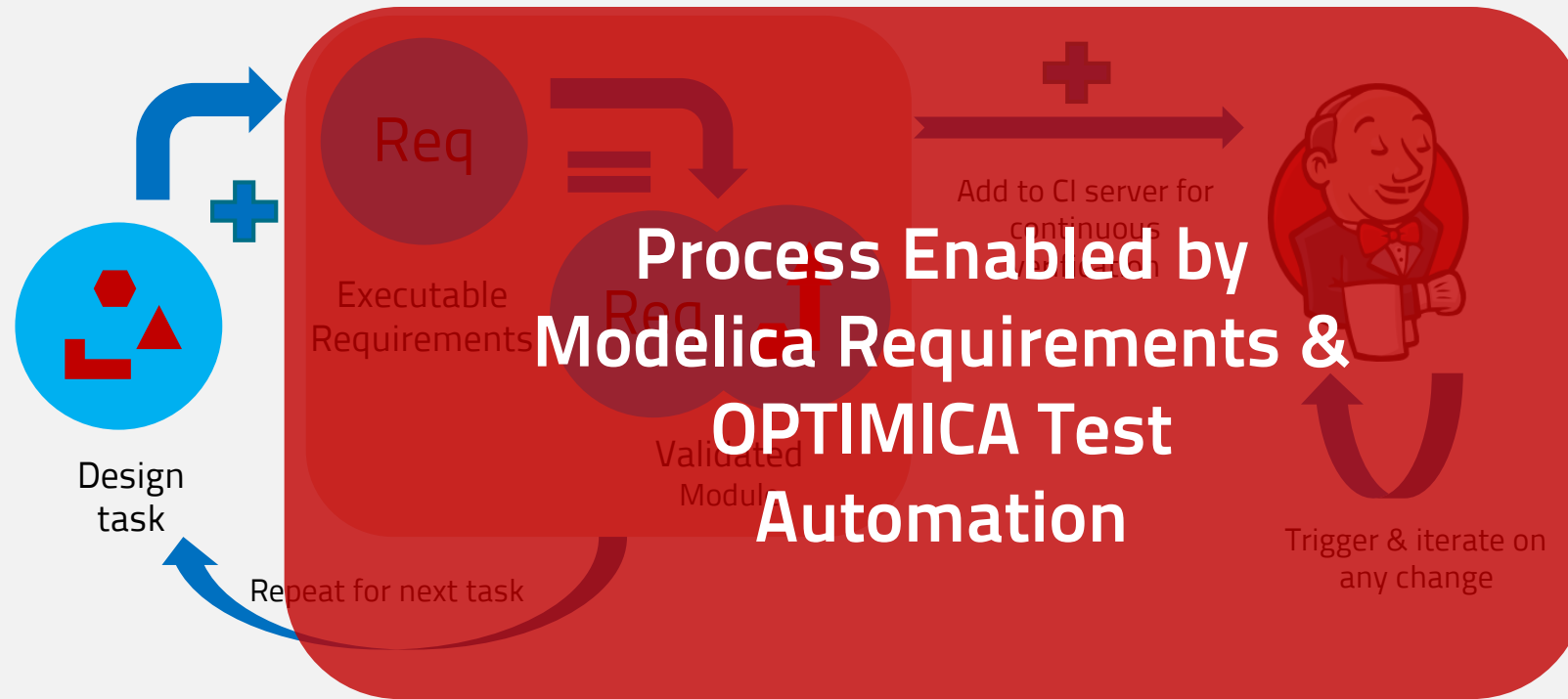


(x,y) coordinates of input must stay within closed polygon (output: closest distance to polygon + property)

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

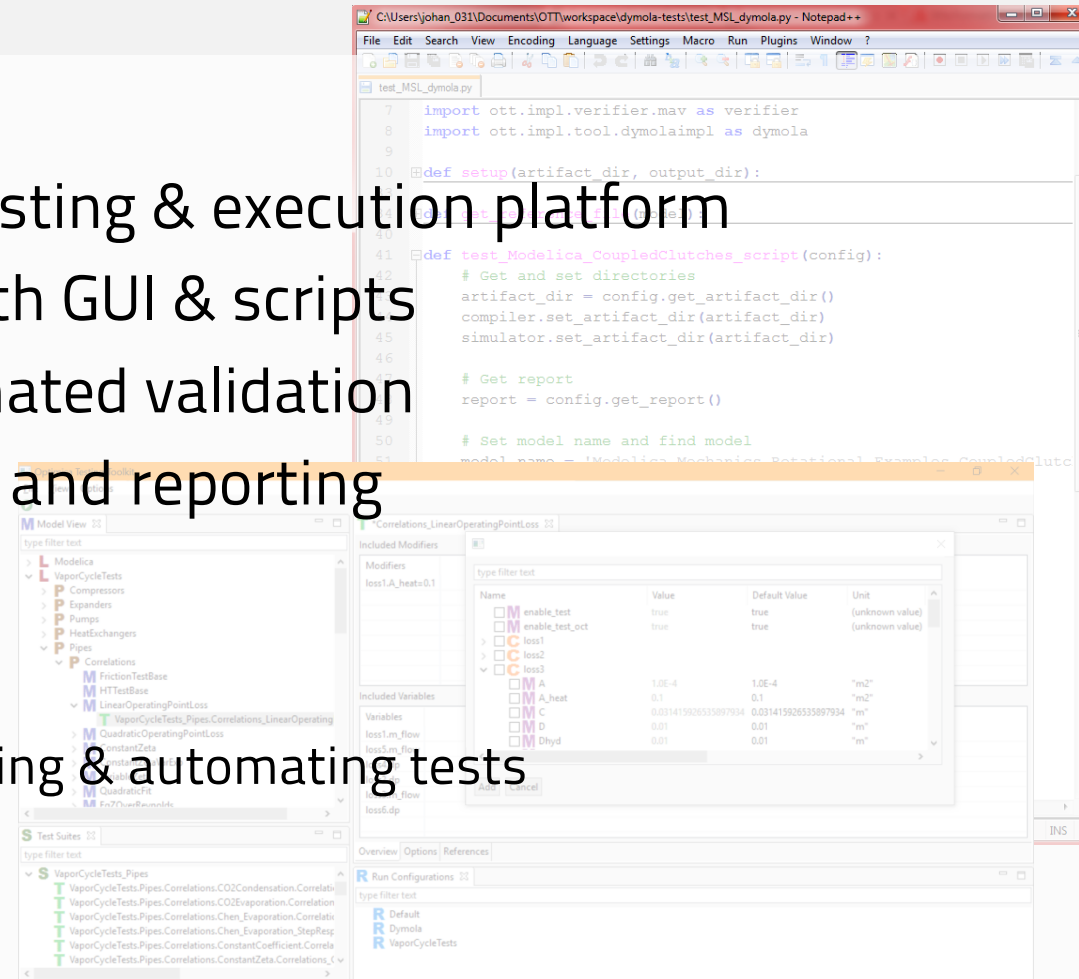
- Architecture

- 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

The screenshot displays the Optimica Testing Toolkit interface. On the left, a tree view shows the project structure with folders for Model View, Requirements, and Test Suites. The main area is divided into two sections: 'NoShutters' and 'SimpleControl', each containing a table of test results. The 'NoShutters' table shows four test cases, all of which passed compilation and simulation, but only two passed verification. The 'SimpleControl' table shows five test cases, all of which passed compilation and simulation, but only one passed verification. A summary row for each table provides overall statistics. A callout box on the right highlights that the report includes hyperlinks to detailed reports for each test case.

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
Unit Test	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 simulation: 5/5

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**.

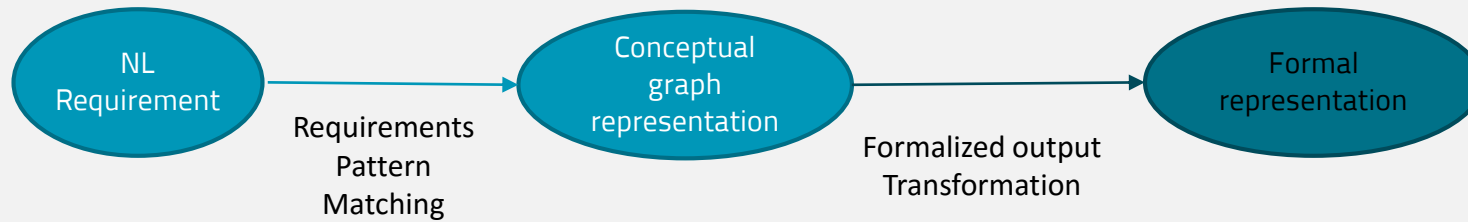
Req-17: **When** auto-control mode is entered, **eventually** the cuff will be inflated.

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

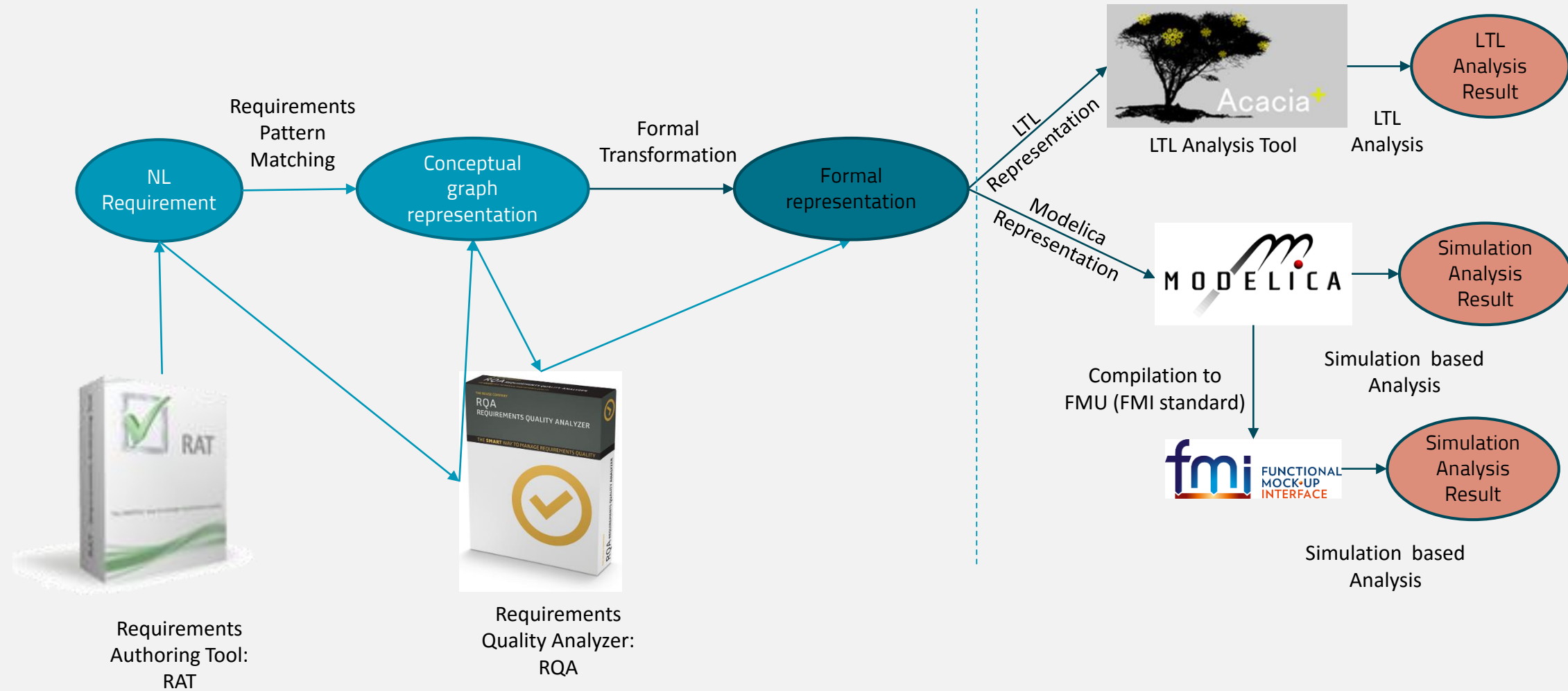
Method

Automatic Translation from Natural Language to Formal representation



Formal Analysis or Simulation based verification

Method



Create a Metric for LTL consistency: Custom Code in RQA

The screenshot displays the Requirements Quality Analyzer (RQA) interface. The main window is titled "Requirements Quality Analyzer" and shows the "Workbook configuration" tab. The "Metrics" section is active, displaying a table of metrics and a context menu for adding a new metric.

Metrics Table:

Identifier	Name	Rationale	Weight	Enabled	Consistency type
10055	Arithmetic operation compliance with SCM metric - Weight		1	<input checked="" type="checkbox"/>	Arithmetic operation compliance ...
10117	Weight consistency metric		1	<input checked="" type="checkbox"/>	Properties consistency metric

Context Menu (Add new metric):

- Properties consistency
- Measurement units consistency
- Measurement units consistency for specific property
- Overlapping consistency
- Arithmetic operation compliance with SCM
- Custom-code** (highlighted with a red box)

Selected metric ranges:

Lower limit	Upper limit	Mandatory	Quality level
0	1	False	High
1	∞	False	Low

Bar Chart:

The bar chart shows the distribution of metrics across quality levels. The Y-axis represents quality levels (High, Medium, Low) and the X-axis represents the number of metrics (0, 1, ∞). A green bar at the High level extends to 1, and a red bar at the Low level extends to ∞.

Footer: RMS Repository: Requirements; Project: Bicycle Requirements.xls Connected to 'C:\Z-REPOSITORY\Projects - Repository\Ontologies\15.1\Bicycle\Bicycle.mdb'

Example

Shared Resource Arbiter

► SRA_2

Client {
 When the *flying engine* activates, the *propeller* shall be canceled until the **ignition** starts
 When the *aircraft* departs, 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

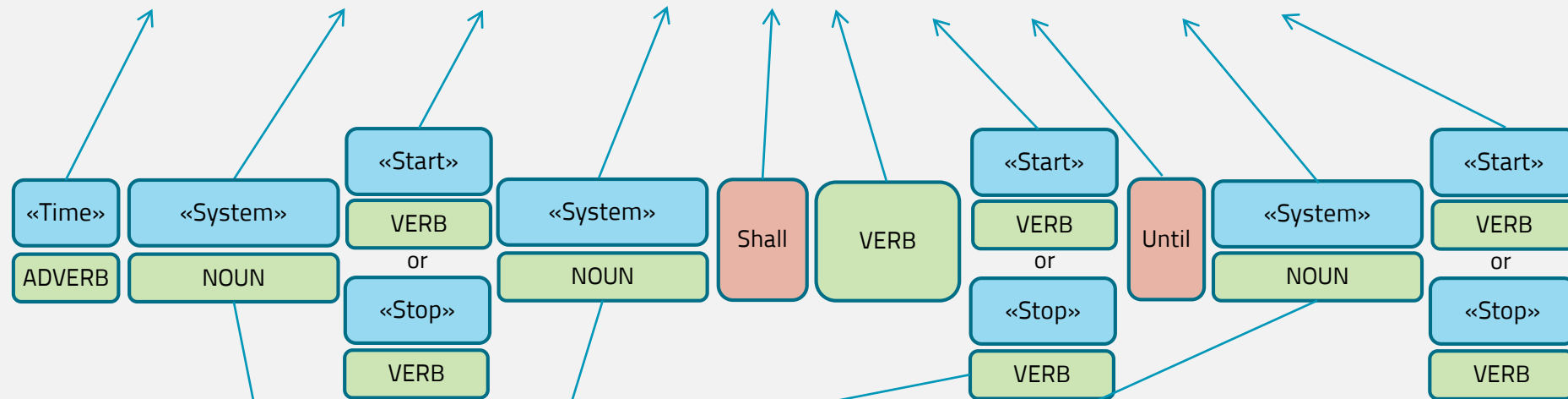


```
G((flying_engine=1) → X((propeller=0)U(ignition=1)));
G((aircraft=1) → X((wheel=0)U(electrical_power_system=1)));
G((navigation_system=1) → X((auto_control_mode=0)U(gearshift=1)));
G(((electrical_power_system=0) * (gearshift=0)) +
  ((ignition=0) * (gearshift=0)) +
  ((ignition=0) * (electrical_power_system=0)));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

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

The screenshot displays the IBM Rational DOORS software interface. The window title is "'SOW Statement of Work' current 0.2 in /SSS Document ISO 29148 (Formal module) - DOORS". The menu bar includes File, Edit, View, Insert, Link, Analysis, Table, Tools, Discussions, Authoring, user, Change Management, and Help. The toolbar contains various icons for document manipulation. The left pane shows a tree view of the document structure:

- SOW Statement of Work
 - 0. Statement of Work Format. (S)
 - 1 SCOPE
 - 2 APPLICABLE DOCUMENTS
 - 3 REQUIREMENTS

The main pane displays the content of the selected section, showing a table with ID, a checkbox, and text:

ID	
1	0. Statement of Work Format. (SOW)
2	• General.
3	0.1 SCOPE
4	<input checked="" type="checkbox"/> Include here a statement about what this SOW covers. If applicable, provide some background information that will be helpful to clarify the needs of the procurement.
5	0.1.1 Background
113	<i>Do not discuss any work tasks in section 1</i>
6	0.2 APPLICABLE DOCUMENTS
7	<input checked="" type="checkbox"/> List here all documents invoked in the requirements section of the SOW by document identifier and title. These documents may include Standards, Specifications and other referenced documents needed to identify and clarify the work task or deliverable product Any document listed in the section must be invoked and tailored to meet the minimal needs of the planned procurement in the requirements section.
8	0.2.1 SPECIFICATIONS
9	0.2.2 Standards. <i>MIL-STD-499, MIL-STD-498, MIL-STD-961D, MIL-STD-1521B, others as necessary, etc.</i>
10	0.2.3 Other documents.
11	0.2.4 Availability of documents
12	0.3 REQUIREMENTS
112	0.3.1 General
13	0.3.2 Detail the require tasks.
111	0.3.3 Program Management
14	<input checked="" type="checkbox"/> [Redacted]
15	<input checked="" type="checkbox"/> The following provides an engineering example:
16	1 SCOPE

At the bottom of the window, the status bar shows "Username: Administrator" and "Exclusive edit mode".

Allows Requirements Authoring

The screenshot shows the DOORS software interface. The title bar reads: "'SOW Statement of Work' current 0.2 in /SSS Document ISO 29148 (Formal module) - DOORS". The menu bar includes: File, Edit, View, Insert, Link, Analysis, Table, Tools, Discussions, Authoring, user, Change Management, Help. The 'Authoring' menu is open, showing options: Insert, Edit, Reload ontology, Change RQS Server connection parameters, Select Authoring License, Test License, Assess quality Inline, Change Language, Configure authoring shortcuts, and About. The 'Insert' and 'Edit' options are highlighted with a red box. The main window displays a document structure for 'SOW Statement of Work' with sections for 'SCOPE', 'APPLICABLE DOCUMENTS', and 'REQUIREMENTS'. The 'REQUIREMENTS' section is expanded, showing a list of requirements with their IDs and descriptions.

ID	Description
8	0.2.1 SPECIFICATIONS
9	0.2.2 Standards. <i>MIL-STD-499, MIL-STD-498, MIL-STD-961D, MIL-STD-1521B, others as necessary, etc.</i>
10	0.2.3 Other documents.
11	0.2.4 Availability of documents
12	0.3 REQUIREMENTS
112	0.3.1 General
13	0.3.2 Detail the require tasks.
111	0.3.3 Program Management
14	[Redacted]
15	The following provides an engineering example:
16	1 SCOPE

Username: Administrator Exclusive edit mode

RAT Plug-in running on top of DOORS

Requirements Authoring Tool

Authoring with pattern 'Stakeholder Functional Requirement'

01 - System Functionality Stakeholder Functional Requirement

The following checklist items apply to the development and documentation of safety critical computing system specifications, including CLDs. These items are provided to ensure the comprehensiveness of these specifications. Not all items are appropriate to all designs.

Matching patterns elements:

Weight	Pattern name
	manager of the vehicle shall be able to accelerate the acceleromet

Correctness metrics summary:

Metric	Value
<input checked="" type="checkbox"/> Ambiguous sentences	2
<input checked="" type="checkbox"/> Compound terms	1
<input checked="" type="checkbox"/> R14 - Incorrect spelling	1
<input checked="" type="checkbox"/> R2 - Passive voice	1
<input checked="" type="checkbox"/> R20 - Combinators	1
<input checked="" type="checkbox"/> R24 - Enumeration	1
<input checked="" type="checkbox"/> R35 - Universals	2
<input checked="" type="checkbox"/> R4 - Defined verbs	5
<input checked="" type="checkbox"/> R44 - Synonyms	1
<input checked="" type="checkbox"/> R45 - Acronyms	2
<input checked="" type="checkbox"/> R5 - Imprecise quantifiers	1
<input checked="" type="checkbox"/> Shall occurrences	0
<input checked="" type="checkbox"/> Specific terms	1

[Suggest manual assessment](#) Ready

Other quality elements:

Links	Lessons learned	Quality forums	Requirement original data	Formal representation	Indexing information
Correctness	Consistency	Completeness	Terminology coverage	Overlapping requirements	Additional attributes
Metric	Correctness	Value	Summary		
<input checked="" type="checkbox"/> Ambiguous sentences	☆☆☆	2	Ambiguous sentences must be avoided	<input checked="" type="radio"/> ALL	
<input checked="" type="checkbox"/> Compound terms	☆☆☆	1	Use any part term in the PBS instead of the compound...	<input type="radio"/> APPROPRIATE	
<input checked="" type="checkbox"/> R14 - Incorrect spelling	☆☆☆	1	Avoid misspelling	<input type="radio"/> SYSTEM	
<input checked="" type="checkbox"/> R2 - Passive voice	☆☆☆	1	Avoid passive voice in your requirements		
<input checked="" type="checkbox"/> R20 - Combinators	☆☆☆	1	Avoid coordinators		
<input checked="" type="checkbox"/> R24 - Enumeration	☆☆☆	1	Avoid enumeration		
<input checked="" type="checkbox"/> R35 - Universals	☆☆☆	2	Avoid universals such as 'all', 'both' or 'any'		

Save and close Cancel

RAT Plug-in running on top of DOORS

Requirement Authoring Pane

Authoring with pattern 'Stakeholder Functional Requirement'

01 - System Functionality Stakeholder Functional Requirement

The following checklist items apply to the development and documentation of safety critical computing system specifications, including CLDs. These items are provided to ensure the comprehensiveness of these specifications. Not **all** items are appropriate to **all** designs.

Matching patterns elements:

Weight	Pattern name
	manager of the vehicle shall be able to accelerate the acceleromet

Correctness metrics summary:

Metric	Value
<input checked="" type="checkbox"/> Ambiguous sentences	2
<input checked="" type="checkbox"/> Compound terms	1
<input checked="" type="checkbox"/> R14 - Incorrect spelling	1
<input checked="" type="checkbox"/> R2 - Passive voice	1
<input checked="" type="checkbox"/> R20 - Combinators	1
<input checked="" type="checkbox"/> R24 - Enumeration	1
<input checked="" type="checkbox"/> R35 - Universals	2
<input checked="" type="checkbox"/> R4 - Defined verbs	5
<input checked="" type="checkbox"/> R44 - Synonyms	1
<input checked="" type="checkbox"/> R45 - Acronyms	2
<input checked="" type="checkbox"/> R5 - Imprecise quantifiers	1
<input checked="" type="checkbox"/> Shall occurrences	0
<input checked="" type="checkbox"/> Specific terms	1

[Suggest manual assessment](#) Ready

Other quality elements:

Links	Lessons learned	Quality forums	Requirement original data	Formal representation	Indexing information
Correctness	Consistency	Completeness	Terminology coverage	Overlapping requirements	Additional attributes
Metric	Correctness	Value	Summary		
<input checked="" type="checkbox"/> Ambiguous sentences	☆☆☆	2	Ambiguous sentences must be avoided	ALL	
<input checked="" type="checkbox"/> Compound terms	☆☆☆	1	Use any part term in the PBS instead of the compound...	APPROPRIATE	
<input checked="" type="checkbox"/> R14 - Incorrect spelling	☆☆☆	1	Avoid misspelling	SYSTEM	
<input checked="" type="checkbox"/> R2 - Passive voice	☆☆☆	1	Avoid passive voice in your requirements		
<input checked="" type="checkbox"/> R20 - Combinators	☆☆☆	1	Avoid coordinators		
<input checked="" type="checkbox"/> R24 - Enumeration	☆☆☆	1	Avoid enumeration		
<input checked="" type="checkbox"/> R35 - Universals	☆☆☆	2	Avoid universals such as 'all', 'both' or 'any'		

Save and close Cancel

RAT Plug-in running on top of DOORS

Quality Pane: Correctness

The following checklist items apply to the development and documentation of safety critical computing system specifications, including CLDs. These items are provided to ensure the comprehensiveness of these specifications. Not **all** items are appropriate to **all** designs.

Matching patterns elements:

manager of the vehicle shall be able to accelerate the acceleromet

Metric	Value
✓ Ambiguous sentences	2
✓ Compound terms	1
✓ R14 - Incorrect spelling	1
✓ R2 - Passive voice	1
✓ R20 - Combinators	1
✓ R24 - Enumeration	1
✓ R35 - Universals	2
✓ R4 - Defined verbs	5
✓ R44 - Synonyms	1
✓ R45 - Acronyms	2
✓ R5 - Imprecise quantifiers	1
✓ Shall occurrences	0
✓ Specific terms	1

[Suggest manual assessment](#) Ready

Other quality elements:

Metric	Correctness	Value	Summary
✓ Ambiguous sentences	☆☆☆	2	Ambiguous sentences must be avoided
✓ Compound terms	☆☆☆	1	Use any part term in the PBS instead of the compound...
✓ R14 - Incorrect spelling	☆☆☆	1	Avoid misspelling
✓ R2 - Passive voice	☆☆☆	1	Avoid passive voice in your requirements
✓ R20 - Combinators	☆☆☆	1	Avoid coordinators
✓ R24 - Enumeration	☆☆☆	1	Avoid enumeration
✓ R35 - Universals	☆☆☆	2	Avoid universals such as 'all', 'both' or 'any'

Username: Admin

Save and close Cancel

RAT Plug-in running on top of DOORS

The screenshot displays the Requirements Authoring Tool (RAT) interface. A red box highlights the 'Decision Support Pane' which contains the following content:

Authoring with pattern 'Stakeholder Functional Requirement'

01 - System Functionality Stakeholder Functional Requirement

The following checklist items apply to the document specifications, including CLDs. These items are appropriate for the document specifications. Not all items are appropriate.

Matching patterns elements:

Weight	Pattern name
	manager of the vehicle shall be able to accelerate the accelerometer

Correctness metrics summary:

Metric	Value
<input checked="" type="checkbox"/> Ambiguous sentences	2
<input checked="" type="checkbox"/> Compound terms	1
<input checked="" type="checkbox"/> R14 - Incorrect spelling	1
<input checked="" type="checkbox"/> R2 - Passive voice	1
<input checked="" type="checkbox"/> R20 - Combinators	1
<input checked="" type="checkbox"/> R24 - Enumeration	1
<input checked="" type="checkbox"/> R35 - Universals	2
<input checked="" type="checkbox"/> R4 - Defined verbs	5
<input checked="" type="checkbox"/> R44 - Synonyms	1
<input checked="" type="checkbox"/> R45 - Acronyms	2
<input checked="" type="checkbox"/> R5 - Imprecise quantifiers	1
<input checked="" type="checkbox"/> Shall occurrences	0
<input checked="" type="checkbox"/> Specific terms	1

Other quality elements:

Metric	Correctness	Value	Summary
<input checked="" type="checkbox"/> Ambiguous sentences	☆☆☆	2	Ambiguous sentences must be avoided
<input checked="" type="checkbox"/> Compound terms	☆☆☆	1	Use any part term in the PBS instead of the compound...
<input checked="" type="checkbox"/> R14 - Incorrect spelling	☆☆☆	1	Avoid misspelling
<input checked="" type="checkbox"/> R2 - Passive voice	☆☆☆	1	Avoid passive voice in your requirements
<input checked="" type="checkbox"/> R20 - Combinators	☆☆☆	1	Avoid coordinators
<input checked="" type="checkbox"/> R24 - Enumeration	☆☆☆	1	Avoid enumeration
<input checked="" type="checkbox"/> R35 - Universals	☆☆☆	2	Avoid universals such as 'all', 'both' or 'any'

RAT Plug-in running on top of DOORS

The screenshot displays the Requirements Authoring Tool (RAT) interface. The main window shows a document titled "SOW Statement of Work" current 0.2 in /SSS Document ISO 29148 (Formal module) - DOORS. The tool is currently authoring with the pattern "Stakeholder Functional Requirement".

The main text area contains the following text: "The following checklist items apply to the development and documentation of safety critical computing system specifications, including CLDs. These items are provided to ensure the comprehensiveness of these specifications. Not all items are appropriate to all designs." Below this text, a matching pattern element is shown: "manager of the vehicle shall be able to accelerate the acceleromet".

The "Correctness metrics summary" panel on the right lists various metrics and their values:

Metric	Value
<input checked="" type="checkbox"/> Ambiguous sentences	2
<input checked="" type="checkbox"/> Compound terms	1
<input checked="" type="checkbox"/> R14 - Incorrect spelling	1
<input checked="" type="checkbox"/> R2 - Passive voice	1
<input checked="" type="checkbox"/> R20 - Combinators	1
<input checked="" type="checkbox"/> R24 - Enumeration	1
<input checked="" type="checkbox"/> R35 - Universals	2
<input checked="" type="checkbox"/> R4 - Defined verbs	5
<input checked="" type="checkbox"/> R44 - Synonyms	1
<input checked="" type="checkbox"/> R45 - Acronyms	2
<input checked="" type="checkbox"/> R5 - Imprecise quantifiers	1
<input checked="" type="checkbox"/> Shall occurrences	0
<input checked="" type="checkbox"/> Specific terms	1

Below the metrics summary, there are two callout boxes:

- Structural Quality Value:** A legend showing a color scale from red (Bad Quality) to green (Good Quality).
- Correctness Quality Value:** A legend showing a color scale from red (Bad Quality) to green (Good Quality).

The bottom of the interface shows a table with columns for "Metric", "Correctness", "Value", and "Summary". The "Ambiguous sentences" row is highlighted in blue. The "Correctness" column for this row shows a red indicator, and the "Value" column shows "2".

RAT Plug-in running on top of DOORS

The screenshot shows the Requirements Authoring Tool (RAT) interface. The main window displays a document titled "SOW Statement of Work" with a checklist of items for development and documentation. A "Correctness metrics summary" table is visible on the right, listing various metrics and their values. Below the main window, a "Other quality elements" section is highlighted with a red box, containing a table with columns for Metric, Correctness, Value, and Summary. Annotations with arrows point from text boxes to specific elements in the interface.

Annotations:

- Consistency Issues
- Completeness Issues
- Terminology Coverage
- Overlapping Requirement

Correctness metrics summary table:

Metric	Value
Ambiguous sentences	2
Compound terms	1
R14 - Incorrect spelling	1
R2 - Passive voice	1
R20 - Combinators	1
R24 - Enumeration	1
R35 - Universals	2
Specific terms	1

Other quality elements table:

Metric	Correctness	Value	Summary
Ambiguous sentences	☆☆☆	2	Ambiguous sentences must be avoided
Compound terms	☆☆☆	1	Use any part term in the PBS instead of the compound...
R14 - Incorrect spelling	☆☆☆	1	Avoid misspelling
R2 - Passive voice	☆☆☆	1	Avoid passive voice in your requirements
R20 - Combinators	☆☆☆	1	Avoid coordinators
R24 - Enumeration	☆☆☆	1	Avoid enumeration
R35 - Universals	☆☆☆	2	Avoid universals such as 'all', 'both' or 'any'

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!