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

The System shall...

Behavior Constraint

Behavior Observed

Purpose/Context

SysML

Requirements

Simulation
Purpose, Context & Anatomy of a Requirement

Bill Schindel (of ICTT):
“Requirements are Transfer Functions”
Example System
Example Requirement
(Transfer Function)

**Hinge AB Stiffness**

When subject to <Vacuum Picker Moment A>, [Hinge AB] transmits Score Stiffness Moment AB in accordance with attribute table (Scorebeam_K_value.xlsx).
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 ???
The "Systems Engineering" metamodel is a representation of Bill Schindel's "Systematica" method.

*Simulation library already knows set of all possible actions/flows.*
*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:

- Stakeholder Requirements
  - Design Requirements
    - Monitors
      - Verifier Models
    - Batch Execution
    - Result Report
  - Test Cases
    - Requirements
      - Specifying the requirements in a standard way, e.g., LTL, opens the possibility to automatically generate the executable verifier models.
    - The complete set of executable verifier models can be tested automatically.
    - When requirements are not met, modifications can be made to the system, model, or even the requirements.
    - The requirements manager should be able to verify that all requirements will be tested by the set of verifier models.
    - The report shows a summary overview of the pass/fail results.
  - These are low-level and testable. When possible, also specified in a formal, open standard language.
  - Formalized Requirements
    - These are low-level and testable. When possible, also specified in a formal, open standard language.

- Requirements Manager
  - Connect SE to MBD:
    - Virtual System
      - Real System
    - Executable Environment
      - Yes
      - No
      - These exercise the system dynamics.

- These exercise the system dynamics.

- Combining a test case with one or more monitors allows requirements to be verified.

- The report shows a summary overview of the pass/fail results.
  - These are the executable checks to verify the requirements are met.
  - Result Report
  - Complete Coverage?

- Modify:
  - Reqs
  - System
  - Model

- These exercise the system dynamics.

- Combining a test case with one or more monitors allows requirements to be verified.

- The complete set of executable verifier models can be tested automatically.

- When requirements are not met, modifications can be made to the system, model, or even the requirements.

- The requirements manager should be able to verify that all requirements will be tested by the set of verifier models.

- The report shows a summary overview of the pass/fail results.

- These are the executable checks to verify the requirements are met.

- Result Report

- Complete Coverage?

- Modify:
  - Reqs
  - System
  - Model
AUTOMATED REQUIREMENTS VERIFICATION

Connect SE to MBD:

- Requirements Manager
  - Stakeholder Requirements
  - Design Requirements
  - Formalized Requirements
  - Verifier Models

- Virtual System
- Real System

- Batch Execution
- Result Report

- Complete Coverage?
- Degree of Automation?

Modify:
- Reqs
- System
- Model

Yes
- Done
- All Pass

No

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)
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
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

Method

- NL Requirement
- Conceptual graph representation
- Formal representation

Requirements Pattern Matching
Formalized output Transformation
Formal Analysis or Simulation based verification

- **Method**
  - **LTL Analysis Result**
  - **Simulation Analysis Result**

- **Requirements Authoring Tool**: RAT
  - **Conceptual graph representation**
    - **Requirements Pattern Matching**
    - **Formal Transformation**
      - **Formal representation**
        - **Compilation to FMU (FMI standard)**
          - **Simulation based Analysis**
            - **Simulation Analysis Result**

- **LTL Analysis Tool**
  - **Simulation Analysis Result**

- **Quality Analyzer**: RQA
  - **Modelica Representation**

- **RQA (Requirements Quality Analyzer)**
  - **Acacia**

- **Modelica**
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

\[
G((\text{flying\_engine}=1) \rightarrow X((\text{propeller}=0)U(\text{ignition}=1)));
G((\text{aircraft}=1) \rightarrow X((\text{wheel}=0)U(\text{electrical\_power\_system}=1)));
G((\text{ignition}=0) + (\text{electrical\_power\_system}=0));
\]
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) \rightarrow X((propeller=0)U(ignition=1)));
G((aircraft=1) \rightarrow X((wheel=0)U(electrical_power_system=1)));
G((navigation_system=1) \rightarrow X((auto_control_mode=0)U(gearshift=1)));
G(((electrical_power_system=0) \times (gearshift=0)) +
  ((ignition=0) \times (gearshift=0)) +
  ((ignition=0) \times (electrical_power_system=0)));
Ontology Building
When the flying engine activates, the propeller shall be canceled until the ignition starts.

\[ G((\text{flying_engine}=1) \rightarrow X((\text{propeller}=0)U(\text{ignition}=1))); \]
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

Requirement Authoring Pane
RAT Plug-in running on top of DOORS

Quality Pane: Correctness
RAT Plug-in running on top of DOORS

Decision Support Pane
RAT Plug-in running on top of DOORS

RAT Overview

Correctness

Quality

Value

Structural

Quality

Value

Correctness

Quality

Value
RAT Plug-in running on top of DOORS

- RAT Overview

Consistency
Issues

Completeness
Issues

Terminology
Coverage

Overlapping
Requirements
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!