RIL: Requirements in the Loop

An End-to-End Requirement-based System Engineering and Validation Process
System analysis: MBSE (SysML) tools

- Use cases, functions, architecture… but:
  - Complex, no simulation
  - Ignore the requirements

Language formalization: in-house methods/tools, “boiler plates”

- Avoid natural language ambiguities

Formal verification: proof tools, MBSA

- Expertise required & scalability issues

Language verification: ontological tools

- Very heavy to deploy

Common Practice

Requirements definition: Natural language

- Reference for safety-critical standards

Requirements management: Doors, Reqtify…

- Versioning & traceability features

Advanced Practice

R&D Practice

Requirements definition: Natural language

- Reference for safety-critical standards

Requirements management: Doors, Reqtify…

- Versioning & traceability features

Formal verification: proof tools, MBSA

- Expertise required & scalability issues

Language verification: ontological tools

- Very heavy to deploy

System Engineering: State of the Art
• All transformations are manual
• 70% of the errors introduced in SW projects are introduced during the specification phase, only 4% are detected in that phase
RIL: Requirements In the Loop
Value Proposition

When lightIntensity > 60% and headLight was 'OFF', headLight shall be 'OFF'

- switch is stable during 3 [second]
- lightIntensity goes up and down respectively to 75% and 55%
Value Proposition

Possible Execution of the System
Requirements:

- All functional Requirements can be written using a set of templates
- Example:
  - Natural language requirement: “When the light intensity is less than 60% for more than one second, headlight shall be set to ON”
  - Equivalent template-based requirement:

```
When lightIntensity is less than 60% for more than 1 second, then headlight shall be ‘ON’
```

Or, in Korean:

```
lightIntensity 가 60 % 미만 인 상태가 1 [second] 이상 유지될 때, headLight 는 ‘ON’ 이어야한다
```
Each template has an executable semantics:

When lightIntensity is less then 60% for more than 1 second, then headlight shall be ‘ON

Benefits: Debug the Requirements as soon as you write them
**Use Case:** a set of constraints on the inputs and between the inputs

- \([\text{lightIntensity} \in [0, 1]]\)
- Initially \(\text{lightIntensity} \text{ is } [71\%]\)
- \(\text{lightIntensity goes up and down between } 70\% \text{ and } 55\%\)
- \([\text{derivative of lightIntensity} \in [-0.1] [1/\text{second}], \ 0.1 [1/\text{second}]]\)

One Use Case can be turned into as many Test Vectors as desired

(Automatic Test Vectors Generation)
Value Proposition

Requirements

Use Cases ➔ Requirements

Test Log Files (I/Os)

Observers (requirements)
Theoretical and Technical Backgrounds

Compiler

Data constraints:
- Logico-numerical solver processing the relationships among data

Simulator

Constraints over variables

BDD (Binary Decision Diagrams) + convex polyhedra

- Control graph
- Backtrack mechanism
- Company created in 2013. STIMULUS released in early 2015.
- STIMULUS users in avionics, automotive, transportation, energy.
- International presence: USA, UK, Germany, Spain, Israel, Japan, China, Korea, India
Requirements In the Loop

- Use Cases
- Requirements
- Architecture
- State-Machines

STIMULUS Model

Automated Validation Environment

Test Vectors (stimuli)

Observers (requirements)
STIMULUS
Debug Your Requirements

Demo
STIMULUS Test Campaign

Test Campaign: set of Use Cases

SUT (DLL, FMU...)

Observer (properties being verified)

Test Campaign Results

Details: which Requirement has been violated during which test case
Today’s Industry Practice (2/4)
Today’s Industry Practice (3/4)

In1 = 1, 5, 10, -3

• • • •

Out3 = true, false, true, false

Fixed values

System Under Test

4 test cases for In1
Today’s Industry Practice (4/4)

Input 2: In2 = 4, 6, 20, 0

Output 3: Out3 = false, false, false, true

Fixed values

4 test cases for In2
1. Define **constraints** on the inputs and between the inputs

2. **Generate** numerous test vectors within the constraints

Define the test acceptance criteria **independently** of each individual test vector: use **requirement**-based oracles
Refine & Validate Requirements over a System Architecture
Level 1..................

Level 2..............

Level 3 ...

Landing Gear

Analog Part

Digital Part and GUI

Gears

Doors

Analog Switch

Electro-Valve

Pressure Sensor

... ...

... ...

... ...

... ...

... ...

... ...
Level 1 Requirements

[ LS_RQ_001] When LG_cmd is 'DOWN',
Do:
gears_extended shall be true and doors_closed shall be true once within 15 [second]
afterwards

gears_extended shall be true
doors_closed shall be true

[ LS_RQ_002] When LG_cmd is 'UP',
Do:
gears_retracted shall be true and doors_closed shall be true once within 15 [second]
afterwards

gears_retracted shall be true
doors_closed shall be true

Level 1

Level 2

Level 3

Analog Part

Digital Part and GUI

Gears

Doors

Analog Switch

Electro-Valve

Pressure Sensor
Level 2 Component Requirements

Level 1 .....................................................

Level 2 ......

Level 3 ......

Analog Switch  Electro-Valve  Pressure Sensor

Landing Gear

Analog Part  Digital Part and GUI  Gears  Doors

| AP_REQ_002 | From each LG_cmd is unstable, Do: 
| State shall be 'closed' once within 2 [second] afterwards 
| During 20[second] - period, State shall be 'closed' 

| AP_REQ_003 | When LG_cmd is stable and last State is 'closed' has been true during more than 20 [second], State shall be 'open' 

| AP_REQ_004 | When state is 'closed', 
| When close general_EV has been true during more than 1 [second], pressurization shall be true
Level 2 Component Requirements

Level 1 ......................................................

Level 2 ......

Level 3 ......

Analog Switch  Electro-Valve  Pressure Sensor
Level 3 Component Requirements

Level 1

Level 2

Level 3

Analog Switch
Electro-Valve
Pressure Sensor
Level 3 Component Requirements

Level 1

Level 2

Level 3

- Analog Switch
- Electro-Valve
- Pressure Sensor

Requirements:

[ EV_REQ_001 ] When \( E \) has been true during more than 1 [second], \( H_{out} \) shall be \( H_{in} \)

[ EV_REQ_002 ] When \((\text{not } E)\) has been true during more than 4 [second], \( H_{out} \) shall be zero
Analog Switch
Electro-
Valve
Pressure
Sensor
Landing Gear
Use Cases
Analog Part
Digital Part and GUI
Gears
Doors
Use Cases
Analog Switch
Electro-
Valve
Pressure
Sensor
Use Cases
Analog Switch
Electro-Valve
Pressure Sensor

Landing Gear

Analog Part
Digital Part and GUI
Gears
Doors

Use Cases
STIMULUS – Simulink (or SCADE)

Step 1:
• Test automatically an FMU containing the Simulink model in STIMULUS
• STIMULUS identifies the **violations** of the specification
• Export the test cases leading to the errors
Step 2:

- Use the generated test cases (csv) to debug the Simulink model in Simulink.