VALU3S

Formalising Verifiable Requirements for an Aircraft Engine Controller

Marie Farrell, Matt Luckcuck, Rosemary Monahan (Email: valu3s@mu.ie)

11th June 2021

Dissemination - Public (PU)

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey. Through ESCEL JU this project has received funding from Enterprise Ireland under grant agreement No IR20200054.

Use Case 5: Aircraft Engine Controller

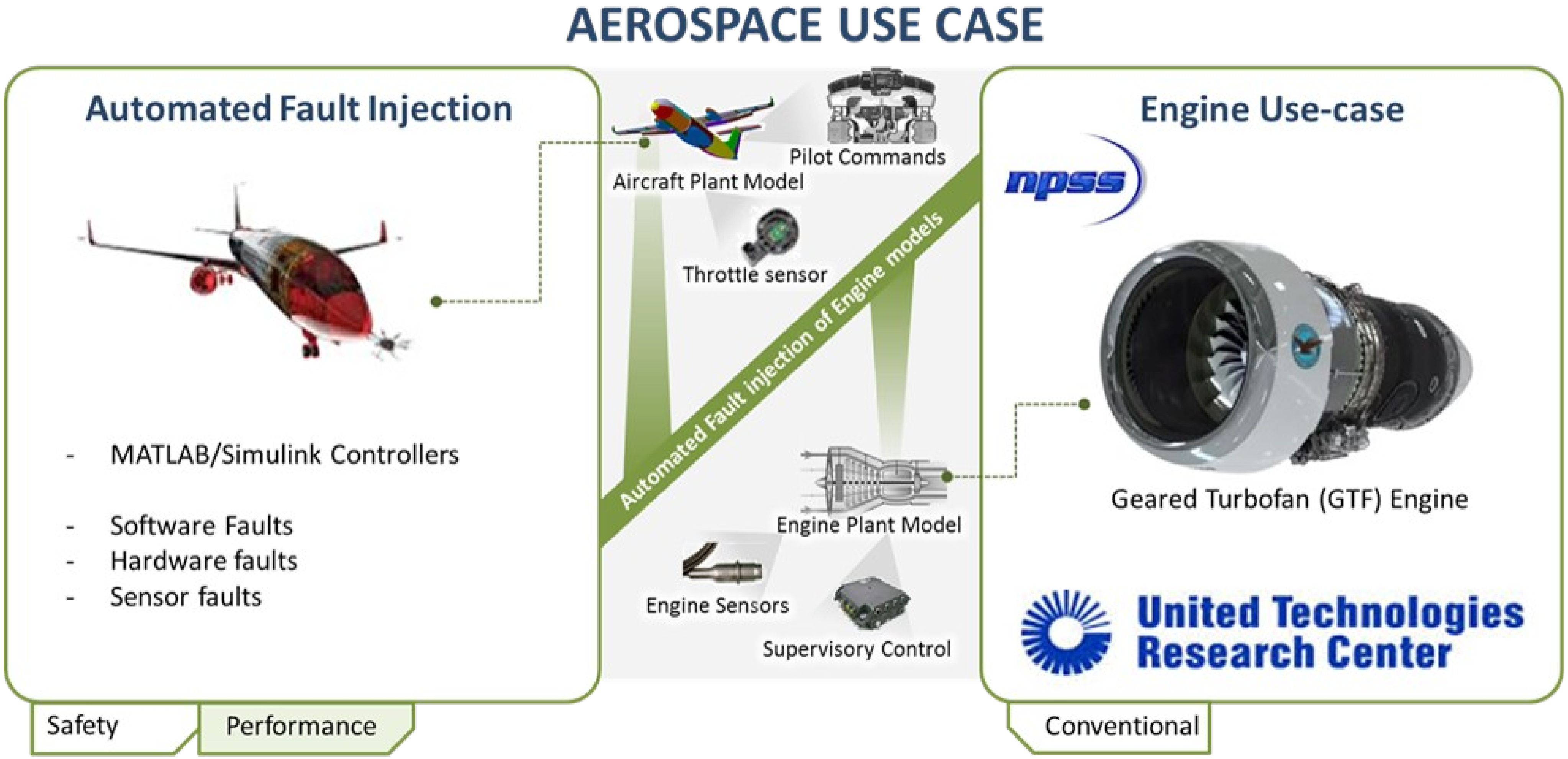
Overview

- Formalising requirements for Use Case 5's Aircraft Engine Controller

Workflow

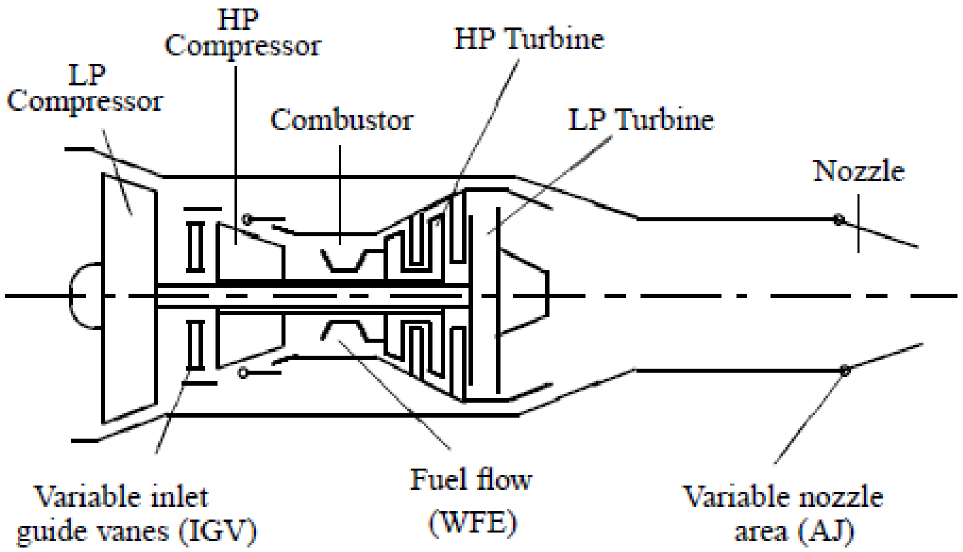
- Using the Formal Requirements Elicitation Tool (FRET)…
- … produce initial formalised requirements
- Refine requirements
  - Adding detail
  - Checking with Use Case Provider
- Decompose Requirements
  - Split up to keep them manageable
  - Identify which elements of Simulink model requirements relate to

Use Case 5: Aircraft Engine Controller



Use Case 5: Aircraft Engine Controller

- FADEC: Full Authority Digital Engine Control
- Monitors and controls the engine e.g. fuel flow.
- Responds to pilot input and sensor data.



Postlethwaite et al., 1995

Formal Verification

- Proving or disproving the correctness of a system with respect to a certain formal specification or property.
- Two broad categories that we are focusing on:

   1. *Model-checkers* exhaustively examine the state space

   Previous VALU3S Tutorial: https://www.youtube.com/watch?v=tU_aOytuqLg&t=450s

   2. *Theorem provers* provide a deductive proof of correctness for the system.

- Particularly useful when a high degree of reliability is sought or to provide robust evidence for regulators.

Formalising Use Case 5 Requirements

Designed with Verification in Mind

Verification is essential, but costly and time-consuming. The large cost and time can be mediated by good design practices:

- Detailed requirements formalisation and elicitation
- Modularity
- Isolate critical components
- Heterogeneous/corroborative verification using multiple techniques

Natural Language Requirements

Example:

"Under sensor faults, while tracking pilot commands, control objectives shall be satisfied (e.g. settling time, overshoot, and steady state error will be within predefined, acceptable limits)."

Ambiguous:

- How do you describe a sensor fault?
- What are the values for settling time, etc.?
- Is this a complete list of the control objectives?
- What does "tracking pilot commands" mean?

Formal Requirements Elicitation Tool: FRET

NASA Requirements Tool: https://github.com/NASA-SW-VnV/fret

- Supports the formalisation, understanding and analysis of requirements
- Graphical interface
- Intuitive diagrammatic explanations of requirement semantics
- Users specify requirements in restricted natural language, called FRETISH, which embodies a temporal logic semantics

Formalising UC5 Requirements using FRET

Update Requirement

Requirement ID
UC5_R_1

Parent Requirement ID

Project
EngineController

Rationale and Comments

Rationale

Under sensor faults, while tracking pilot commands, control objectives shall be satisfied (e.g. settling time, overshoot, and steady state error will be within predefined, acceptable limits)

Comments

Requirement Description

A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "*". For information on a field format, click on its corresponding bubble.

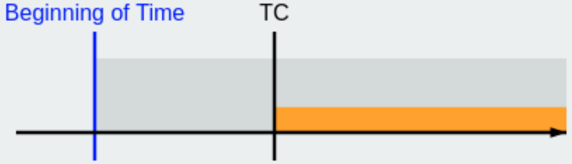
SCOPE CONDITIONS COMPONENT* SHALL* TIMING RESPONSES*

if ((sensorfaults) & (trackingPilotCommands)) Controller shall satisfy (controlObjectives)

SEMANTICS

ASSISTANT TEMPLATES

ENFORCED: in the interval defined by the entire execution. TRIGGER: first point in the interval if *(((sensorfaults) & (trackingPilotCommands)))* is true and any point in the interval where *(((sensorfaults) & (trackingPilotCommands)))* becomes true (from false). REQUIRES: for every trigger, RES must hold at some time point between (and including) the trigger and the end of the interval.



TC = *(((sensorfaults) & (trackingPilotCommands)))*, Response = *((controlObjectives))*.

Diagram Semantics

Formalizations

Future Time LTL

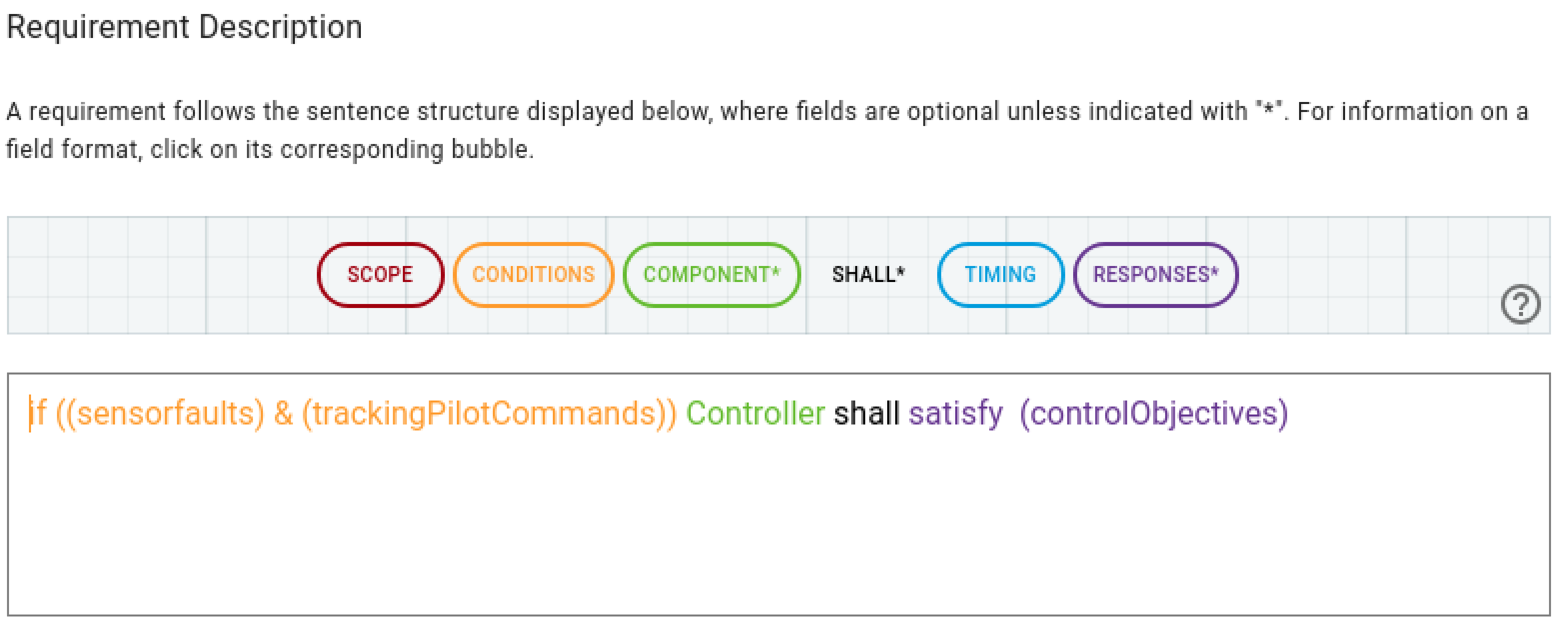
Past Time LTL

SIMULATE

Formalising Requirements

Example:

"Under sensor faults, while tracking pilot commands, control objectives shall be satisfied (e.g. settling time, overshoot, and steady state error will be within predefined, acceptable limits)."



Formalising UC5 Requirements using FRET

Update Requirement

Requirement ID: UC5_R_1.1
Parent Requirement ID: UC5_R_1
Project: EngineController

Rationale and Comments

Rationale

Under sensor faults, while tracking pilot commands, control objectives shall be satisfied (e.g. settling time, overshoot, and steady state error will be within predefined, acceptable limits)

From Test Cases UC5_TC_1 and UC5_TC_2:
sensor S value deviates at most +/- R % from nominal value
sensor S value is not available

Comments

settling time

Requirement Description

A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "*". For information on a field format, click on its corresponding bubble.

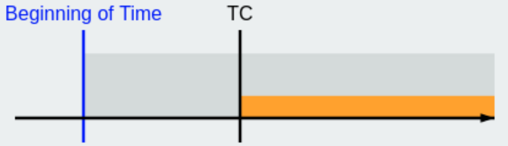
SCOPE CONDITIONS COMPONENT* SHALL* TIMING RESPONSES*

if ((sensorValue > nominalValue + R) | (sensorValue < nominalValue - R) | (sensorValue = null)) & (trackingPilotCommands) Controller shall satisfy (settlingTime >= settlingTimeMin) & (settlingTime <= settlingTimeMax)

SEMANTICS

ASSISTANT TEMPLATES

ENFORCED: in the interval defined by the entire execution. TRIGGER: first point in the interval if *(((sensorValue > nominalValue + R) | (sensorValue < nominalValue - R) | (sensorValue = null)) & (trackingPilotCommands))* is true and any point in the interval where *(((sensorValue > nominalValue + R) | (sensorValue < nominalValue - R) | (sensorValue = null)) & (trackingPilotCommands))* becomes true (from false). REQUIRES: for every trigger, RES must hold at some time point between (and including) the trigger and the end of the interval.



TC = *(((sensorValue > nominalValue + R) | (sensorValue < nominalValue - R) | (sensorValue = null)) & (trackingPilotCommands))*, Response = *((settlingTime >= settlingTimeMin) & (settlingTime <= settlingTimeMax))*.

Diagram Semantics

Formalizations

Future Time LTL

Past Time LTL

SIMULATE

Future Time LTL

```
((LAST V (((! (( ( sensorValue > nominalValue + R ) | ( sensorValue < nominalValue - R ) | ( sensorValue = null ) ) & ( trackingPilotCommands ))) & ((! LAST) & (X (( ( sensorValue > nominalValue + R ) | ( sensorValue < nominalValue - R ) | ( sensorValue = null ) ) & ( trackingPilotCommands ))))) -> (X ((! LAST) U (( settlingTime >= settlingTimeMin ) & ( settlingTime <= settlingTimeMax )))))) & ((( ( sensorValue > nominalValue + R ) | ( sensorValue < nominalValue - R ) | ( sensorValue = null ) ) & ( trackingPilotCommands )) -> ((! LAST) U (( settlingTime >= settlingTimeMin ) & ( settlingTime <= settlingTimeMax ))))
```

Target: *Controller* component.

Past Time LTL

```
((H (! (( ( sensorValue > nominalValue + R ) | ( sensorValue < nominalValue - R ) | ( sensorValue = null ) ) & ( trackingPilotCommands )))) | (! ((! (( settlingTime >= settlingTimeMin ) & ( settlingTime <= settlingTimeMax ))) S ((! (( settlingTime >= settlingTimeMin ) & ( settlingTime <= settlingTimeMax ))) & ((( ( sensorValue > nominalValue + R ) | ( sensorValue < nominalValue - R ) | ( sensorValue = null ) ) & ( trackingPilotCommands )) & ((Y (! (( ( sensorValue > nominalValue + R ) | ( sensorValue < nominalValue - R ) | ( sensorValue = null ) ) & ( trackingPilotCommands )))) | FTP))))))
```

Target: *Controller* component.

Formalising UC5 Requirements using FRET

Rationale

Under sensor faults, while tracking pilot commands, control objectives shall be sa
settling time, overshoot, and steady state error will be within predefined, acceptab

From Test Cases UC5_TC_1 and UC5_TC_2:
sensor S value deviates at most +/- R % from nominal value
sensor S value is not available

Comments

settling time

Requirement Description

A requirement follows the sentence structure displayed below, where fields are optional unless indicated w
information on a field format, click on its corresponding bubble.

SCOPE CONDITIONS COMPONENT* SHALL* TIMING RESPONSES*

if ((sensorValue > nominalValue + R) | (sensorValue < nominalValue - R) | (sensorVal
(trackingPilotCommands) Controller shall satisfy (settlingTime >= settlingTimeMin)
(settlingTime <= settlingTimeMax)

Formalising UC5 Requirements using FRET

Update Requirement

Requirement ID
UC5_R_1.2

Parent Requirement ID
UC5_R_1

Project
EngineController

Rationale and Comments

Rationale
Under sensor faults, while tracking pilot commands, control objectives shall be satisfied (e.g. settling time, overshoot, and steady state error will be within predefined, acceptable limits)

Comments
overshoot

Requirement Description

A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "*". For information on a field format, click on its corresponding bubble.

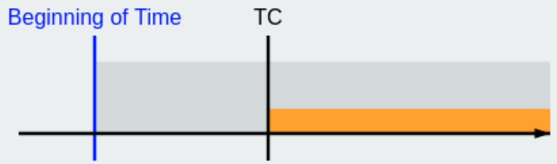
SCOPE CONDITIONS COMPONENT* SHALL* TIMING RESPONSES*

if ((sensorValue > nominalValue + R) | (sensorValue < nominalValue - R) | (sensorValue = null)) & (trackingPilotCommands) Controller shall satisfy (overshoot >= overshootMin) & (overshoot <= overshootMax)

SEMANTICS

ASSISTANT TEMPLATES

ENFORCED: in the interval defined by the entire execution. TRIGGER: first point in the interval if *(((sensorValue > nominalValue + R) | (sensorValue < nominalValue - R) | (sensorValue = null)) & (trackingPilotCommands))* is true and any point in the interval where *(((sensorValue > nominalValue + R) | (sensorValue < nominalValue - R) | (sensorValue = null)) & (trackingPilotCommands))* becomes true (from false). REQUIRES: for every trigger, RES must hold at some time point between (and including) the trigger and the end of the interval.



TC = *(((sensorValue > nominalValue + R) | (sensorValue < nominalValue - R) | (sensorValue = null)) & (trackingPilotCommands))*, Response = *((overshoot >= overshootMin) & (overshoot <= overshootMax))*.

Diagram Semantics

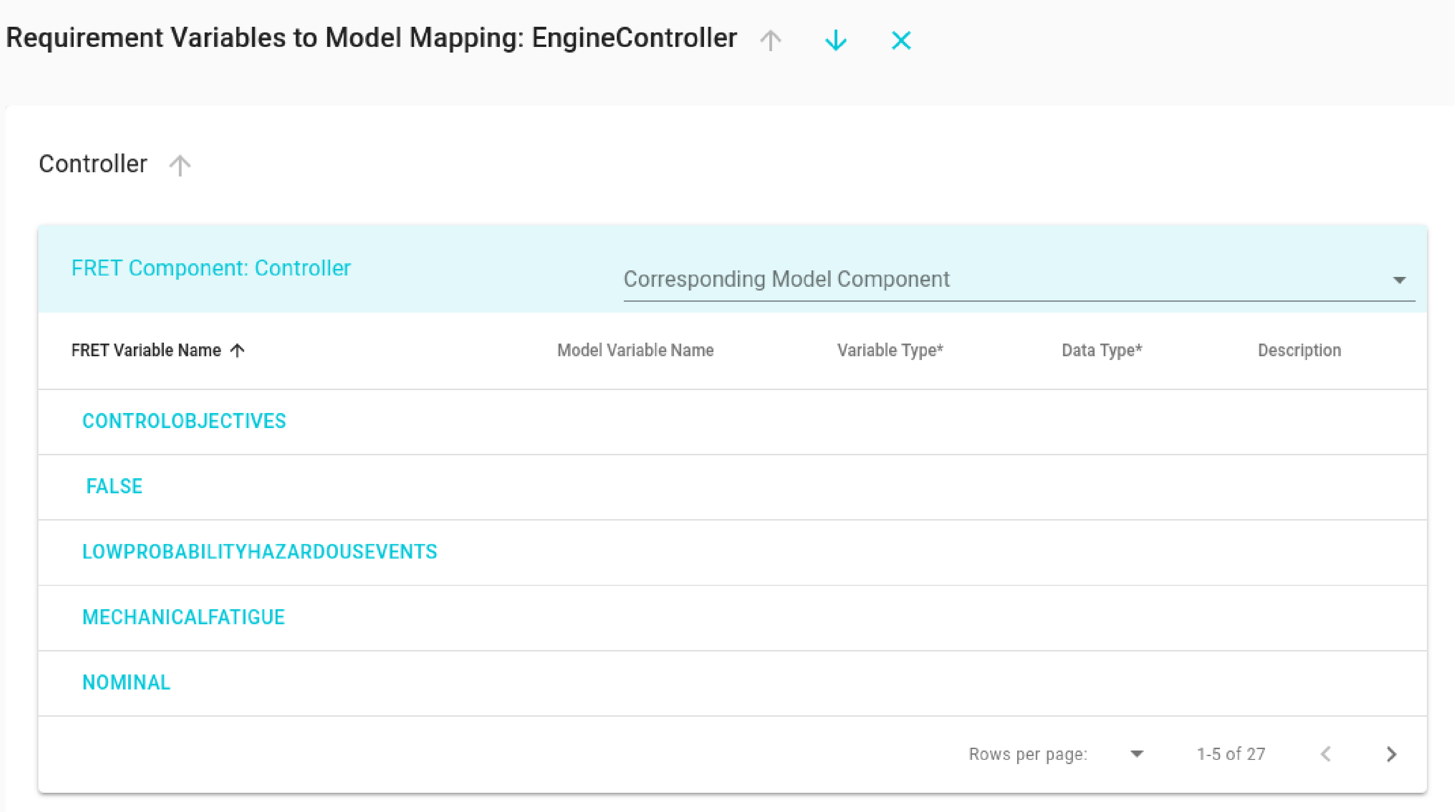
Formalizations

Future Time LTL

Past Time LTL

Using FRET with Other Tools

FRET connects to Simulink models and generates CoCoSim (https://github.com/NASA-SW-VnV/CoCoSim) contracts for model-checking with Kind2.



Using FRET with Other Tools

- The LTL representation can also be used to generate runtime monitors for the implemented system.
- FRET requirements without timing constraints can be used to specify requirements in other formalisms e.g. Event-B.

Formalised Requirements in Development

Methodology: Designed for Verification

Step 0: Characterize initial system.

Step 1: Create initial system model.

Step 2: Perform preliminary hazard analysis.

Step 3: Define mitigations and safety requirements.

Step 4: Refine system model according to mitigations.

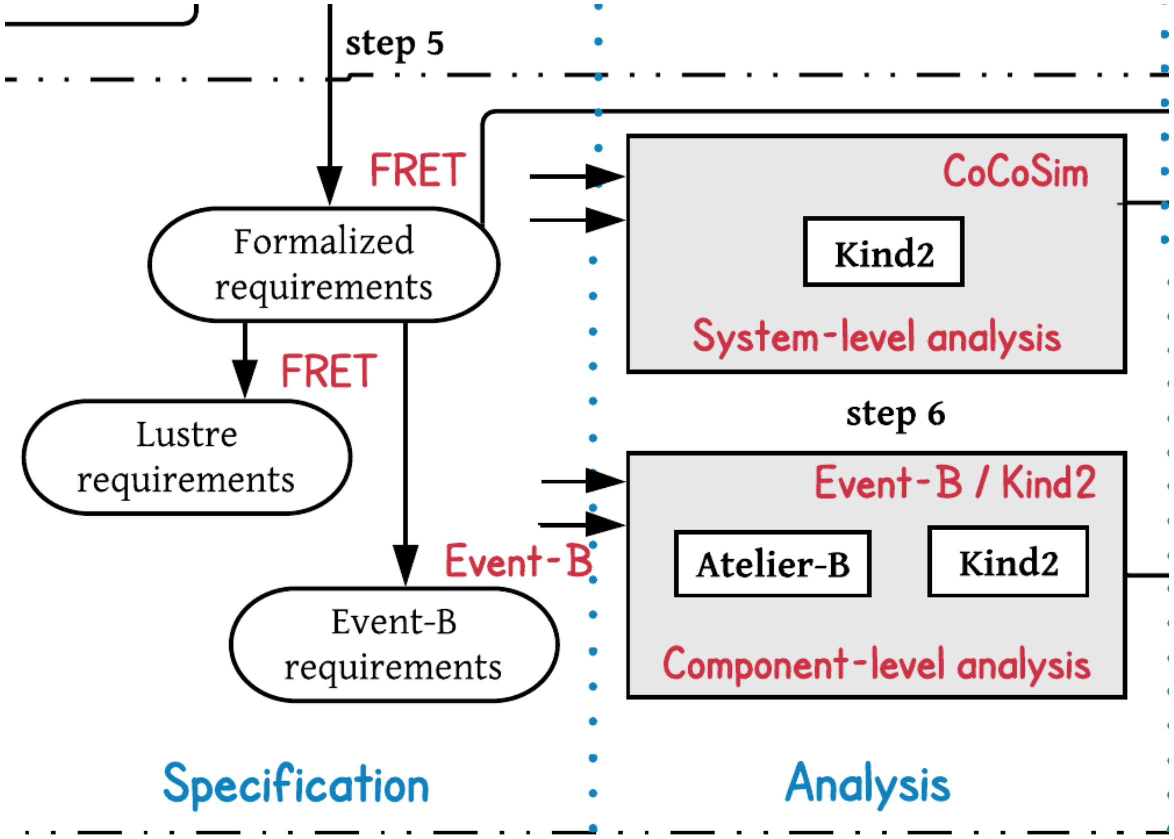
Step 5: Formalize requirements and create formal specification(s).

Step 6: Perform verification and simulation at system-and component-levels.

Step 7: Document verification results and build safety case.

Bourbouh, H., Farrell, M., Mavridou, A., Sljvio, I., Dennis, L.A., Fisher, M., Brat, G. Integrating Formal Verification and Assurance: An Inspection Rover Case Study. NASA Formal Methods Symposium, 2021.

Methodology: Designed for Verification



Bourbouh, H., Farrell, M., Mavridou, A., Sljvio, I., Dennis, L.A., Fisher, M., Brat, G. Integrating Formal Verification and Assurance: An Inspection Rover Case Study. NASA Formal Methods Symposium, 2021.

Conclusions

- Formalised requirements are useful for verification and traceability between verification artefacts
  - Using FRET revealed ambiguities in the natural language requirements that could be made more explicit later.
  - FRET can act as a useful bridge for communication between academic and industrial partners.
  - Formalised requirements simplify formal verification tasks later in the development process.
- Aim to integrate formal methods used for verification and safety analysis of automated systems
  - Developing rigorous methods that are compatible with industry.
  - Supporting interoperability between formalisms.

VALU3S

Questions?

Marie Farrell, Matt Luckcuck, Rosemary Monahan

Email: valu3s@mu.ie

11th June 2021

Dissemination - Public (PU)

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 876852. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Austria, Czech Republic, Germany, Ireland, Italy, Portugal, Spain, Sweden, Turkey. Through ESCEL JU this project has received funding from Enterprise Ireland under grant agreement No IR20200054.