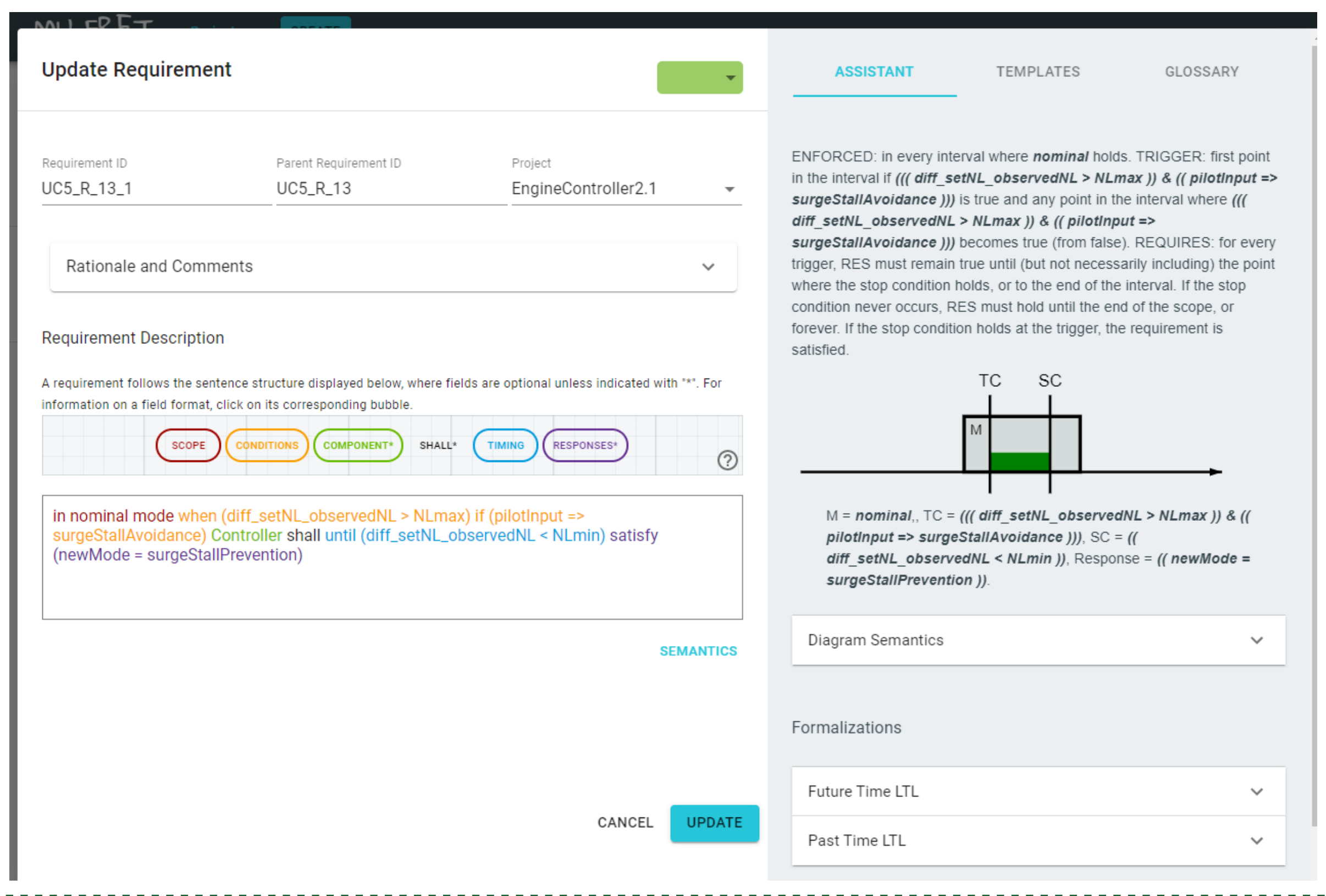
UC5 – Aircraft Engine Controller

Using Mu-FRET for Parent-Child Relationship, and comparisons with EARS

Shi Hao To, Cathal Peelo,
Oisín Sheridan, Rosemary Monahan,
Maynooth University, Ireland

Mu-FRET



Update Requirement

Requirement ID: UC5_R_13_1
Parent Requirement ID: UC5_R_13
Project: EngineController2.1

Rationale and 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*

in nominal mode when (diff_setNL_observedNL > NLmax) if (pilotInput => surgeStallAvoidance) Controller shall until (diff_setNL_observedNL < NLmin) satisfy (newMode = surgeStallPrevention)

SEMANTICS

CANCEL UPDATE

ASSISTANT TEMPLATES GLOSSARY

ENFORCED: in every interval where ***nominal*** holds. TRIGGER: first point in the interval if ***(((diff_setNL_observedNL > NLmax)) & ((pilotInput => surgeStallAvoidance)))*** is true and any point in the interval where ***(((diff_setNL_observedNL > NLmax)) & ((pilotInput => surgeStallAvoidance)))*** becomes true (from false). REQUIRES: for every trigger, RES must remain true until (but not necessarily including) the point where the stop condition holds, or to the end of the interval. If the stop condition never occurs, RES must hold until the end of the scope, or forever. If the stop condition holds at the trigger, the requirement is satisfied.

M = ***nominal***,, TC = ***(((diff_setNL_observedNL > NLmax)) & ((pilotInput => surgeStallAvoidance)))***, SC = ***((diff_setNL_observedNL < NLmin))***, Response = ***((newMode = surgeStallPrevention))***.

Diagram Semantics

Formalizations

Future Time LTL

Past Time LTL

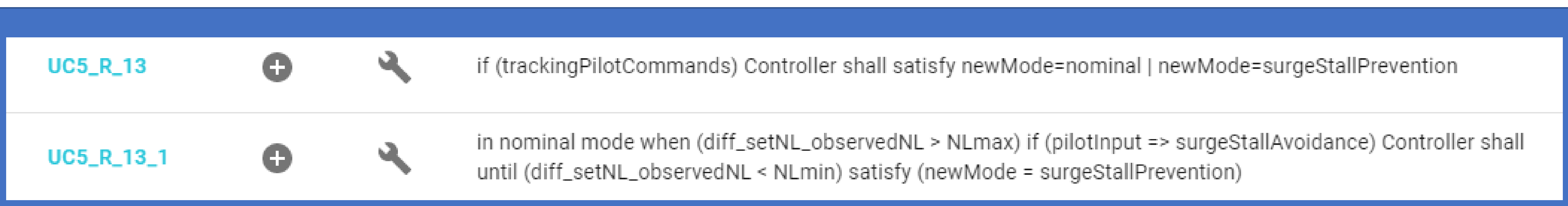
Overview

- ➢ A tool/framework for elicitation, requirements, refactoring, and understanding the requirements.
- ➢ A fork of FRET from NASA.
- ➢ Extends FRET by adding refactoring feature.
- ➢ Enables to extract requirements to a new requirement.
- ➢ The language for Mu-FRET is FRETish.

Installation

- ➢ Install NuSMV and make sure it is on the system's path.
- ➢ Install NodeJS
- ➢ Install python 2.7.18
- ➢ Open a terminal (cmd) in the fret-electron.
- ➢ Run npm run fret-install, or npm run fret-reinstall if FRET is already installed.
- ➢ For more detailed instruction, see the Mu-FRET GitHub.

Using Mu-FRET for Parent-Child relationship



UC5_R_13 — if (trackingPilotCommands) Controller shall satisfy newMode=nominal | newMode=surgeStallPrevention

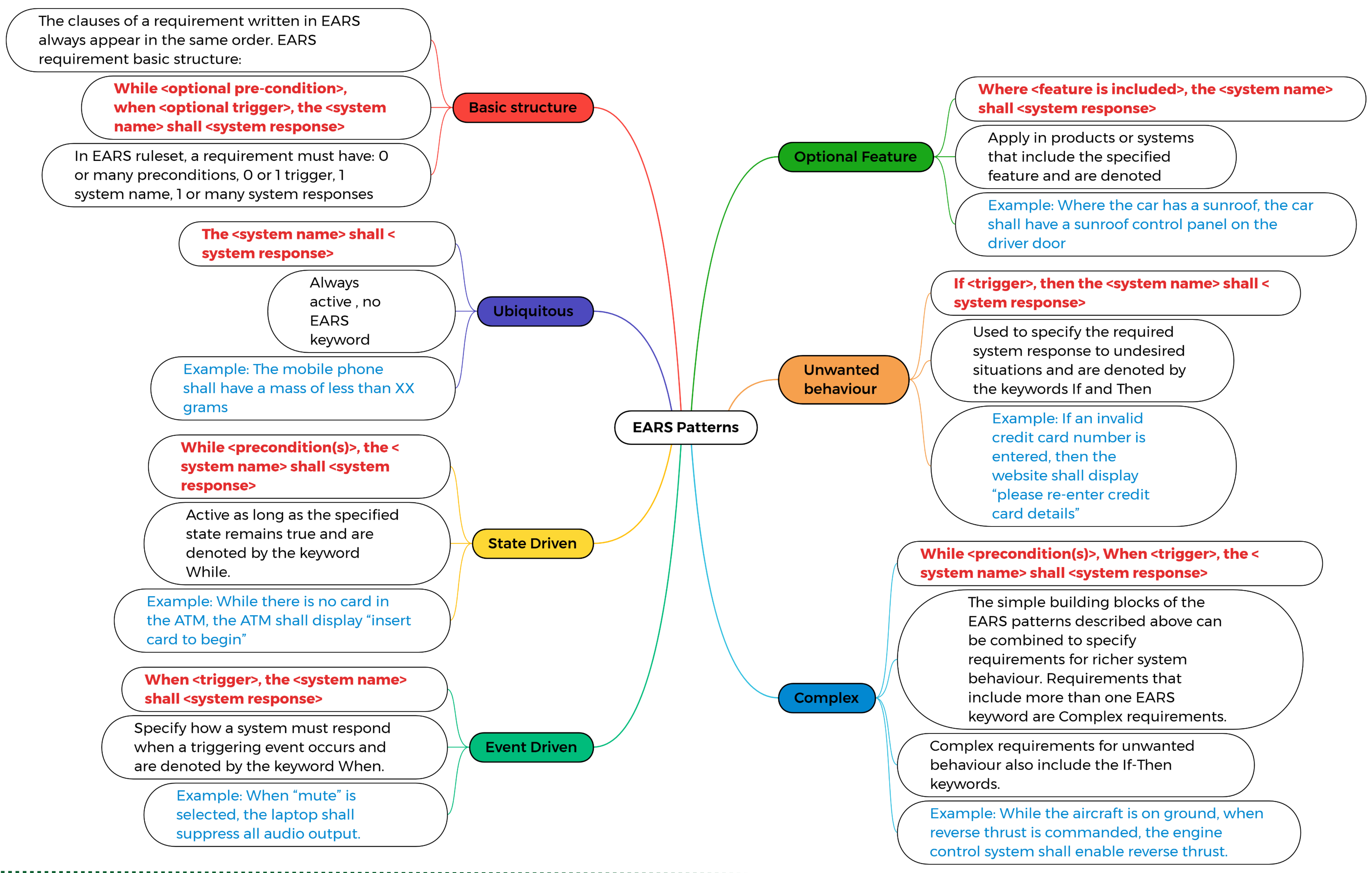
UC5_R_13_1 — in nominal mode when (diff_setNL_observedNL > NLmax) if (pilotInput => surgeStallAvoidance) Controller shall until (diff_setNL_observedNL < NLmin) satisfy (newMode = surgeStallPrevention)

- ➢ FRET allows the user to define a parent-child relationship between requirements.
- ➢ For the VALU3S use case, this relationship is analogous to formal refinement where a child requirement acts as a more concrete version of its parent, with details closer to the implementation of the system.
- ➢ The exact semantics of the relationship isn't prescriptively defined, which gives flexibility to the user when creating a hierarchy among the requirements.

MU-FRET on GitHub:

FRETish and EARS

- ➢ EARS stands for **Easy Approach Requirement Syntax**.
- ➢ Created by Alistair Mavin and his colleagues from Rolls-Royce.
- ➢ The first notation was published in 2009.
- ➢ Reduces/ eliminates common problems found in natural language, and the resulting requirements are easy to read.
- ➢ Provides structured guidance for authors to write high quality textual requirements.
- ➢ Lightweight, little training, and no specialist tools required.


EARS Patterns
Basic structure
The clauses of a requirement written in EARS always appear in the same order. EARS requirement basic structure:
While <optional pre-condition>, when <optional trigger>, the <system name> shall <system response>
In EARS ruleset, a requirement must have: 0 or many preconditions, 0 or 1 trigger, 1 system name, 1 or many system responses
Ubiquitous
The <system name> shall < system response>
Always active , no EARS keyword
Example: The mobile phone shall have a mass of less than XX grams
State Driven
While <precondition(s)>, the < system name> shall <system response>
Active as long as the specified state remains true and are denoted by the keyword While.
Example: While there is no card in the ATM, the ATM shall display "insert card to begin"
Event Driven
When <trigger>, the <system name> shall <system response>
Specify how a system must respond when a triggering event occurs and are denoted by the keyword When.
Example: When "mute" is selected, the laptop shall suppress all audio output.
Optional Feature
Where <feature is included>, the <system name> shall <system response>
Apply in products or systems that include the specified feature and are denoted
Example: Where the car has a sunroof, the car shall have a sunroof control panel on the driver door
Unwanted behaviour
If <trigger>, then the <system name> shall < system response>
Used to specify the required system response to undesired situations and are denoted by the keywords If and Then
Example: If an invalid credit card number is entered, then the website shall display "please re-enter credit card details"
Complex
While <precondition(s)>, When <trigger>, the < system name> shall <system response>
The simple building blocks of the EARS patterns described above can be combined to specify requirements for richer system behaviour. Requirements that include more than one EARS keyword are Complex requirements.
Complex requirements for unwanted behaviour also include the If-Then keywords.
Example: While the aircraft is on ground, when reverse thrust is commanded, the engine control system shall enable reverse thrust.

FRET and EARS Comparison

EARS:

While in nominal mode and the difference between the set NL and the observed NL is less than the minimum NL, when the difference between the set NL and the observed NL is greater than the maximum NL, and the pilot's input implies surge stall avoidance, the controller shall prevent a surge stall

FRETISH:

In nominal mode when (diff_setNL_observedNL > NLmax) if (pilotInput => surgeStallAvoidance) Controller shall until (diff_setNL_observedNL < NLmin) satisfy (newMode = surgeStallPrevention

Evaluation

- ➢ The syntax of EARS is closer to natural language than FRETish, while FRETish is more structured and compact.
- ➢ EARS is not bound to a particular tool.
- ➢ FRET provides a verification environment, which includes a structured language to express requirements.
- ➢ The FRET tool provides requirement verification via model checking.
- ➢ The Mu-FRET tool adds support for refactoring requirements.

References

- https://repo.valu3s.eu/tools/improved-developed-tool/mu-fret
- https://alistairmavin.com/ears/
- https://www.researchgate.net/publication/224079416_Easy_approach_to_requirements_syntax_EARS
- https://www.iaria.org/conferences2013/filesICCGI13/ICCGI_2013_Tutorial_Terzakis.pdf

Involved VALU3S Partners

Leader: Maynooth University National University of Ireland Maynooth

Participating Partners: ENTERPRISE IRELAND Collins Aerospace

VALU3S HAS RECEIVED FUNDING FROM THE ECSEL JOINT UNDERTAKING UNDER GRANT AGREEMENT NO 876852. THE JOINT UNDERTAKING RECEIVES SUPPORT FROM THE EUROPEAN UNION'S HORIZON 2020 RESEARCH AND INNOVATION PROGRAMME AND AUSTRIA, CZECH REPUBLIC, GERMANY, IRELAND, ITALY, PORTUGAL, SPAIN, SWEDEN, TURKEY.

ECSEL Joint Undertaking
Electronic Components and Systems for European Leadership