

## Software Verification



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

Shi Hao To, Cathal Peelo, Oisín Sheridan, Prof. Rosemary Monahan Department of Computer Science, shi.to.2021@mumail.ie, Oisin.Sheridan.2019@mumail.ie, Cathal.Peelo@mu.ie, Rosemary.Monahan@mu.ie

## **MU-FRET** Update Requirement UC5\_R\_13\_1 Rationale and Comments nce) Controller shall until (diff\_setNL\_observedNL < NLmin) satisf pilotInput => surgeStallAvoidance))), SC = ((diff setNL observedNL < NLmin )), Response = (( newMode surgeStallPrevention )) Future Time LTL

#### What is FRET?

- > Stands for Formal Requirement **Eliciation Tool**
- A tool/framework for elicitation, requirements, refactoring, and understanding the requirements.

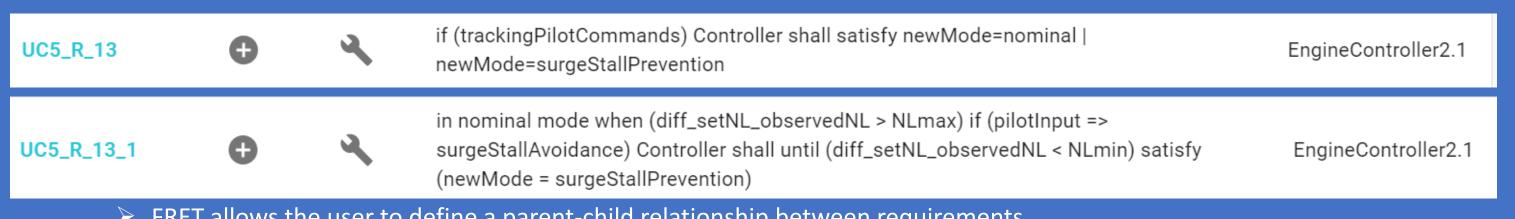
#### What is MU-FRET?

- 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 for MU-FRET**

- 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 fretelectron.
- 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**



- > 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 resultant requirements are easy to read.
- Provides structured guidance for authors to write high quality textual requirements.
- Lightweight, little training, and no specialist tools required.

#### **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

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

#### **Evaluation**

- > EARS provides better yet easy looking syntax compared to Mu-FRET.
- > FRET has better requirement/ verification testing.
- Compared to EARS, Mu-FRET has functionality of refactoring
- > EARS do not need any tools, but FRET requires tool for requirements.
- > FRETish has less wordiness than EARS, in representing the requirements.

#### References

- https://repo.valu3s.eu/tools/improved-developedtool/mu-fret
- https://alistairmavin.com/ears/
- https://www.researchgate.net/publication/224079416 **Easy\_approach\_to\_requirements\_syntax\_EARS**
- https://www.iaria.org/conferences2013/filesICCGI13/I CCGI 2013 Tutorial Terzakis.pdf

#### **Acknowledgements**

I would like to thank my supervisor Rosemary, PhD student Oisin, and research assistant Cathal for giving me the opportunity to work together on this six-week SPUR project.

#### Involved VALU3S Partners











