Maynooth University
National University of Ireland Maynooth

FRETing about Requirements:

Formalised Requirements for an Aircraft Engine Controller

Marie Farrell **Matt Luckcuck** Oisín Sheridan Rosemary Monahan

Department of Computer Science, Maynooth University, Ireland

REFSQ 2022
22^{nd} of March 2022

Introduction

Overview

- Our experience using NASA's Formal Requirements Elicitation Tool (FRET)[a]
  - Alongside aerospace industrial partner
  - Natural-language requirements encoded in FRET
- Use Case:
  - Aircraft engine's software controller
  - *Verification and Validation of Automated Systems' Safety and Security* Project[b]
- Bridge communication gap between...
  - Industrial Partner – Aerospace industry specialists
  - Formal methods – Our team

[a] FRET: `https://github.com/NASA-SW-VnV/fret`
[b] VALU3S: `https://valu3s.eu`

Formalising Requirements

- Formal Methods...
  - Broad group of mathematical approaches to software and system development
  - Support rigorous specification, design and verification

Formalising Requirements

- Formal Methods...
  - Broad group of mathematical approaches to software and system development
  - Support rigorous specification, design and verification
- Formal Verification...
  - Proving or disproving the correctness of a system with respect to a certain formal specification or property

Formalising Requirements

- Formal Methods...
  - Broad group of mathematical approaches to software and system development
  - Support rigorous specification, design and verification
- Formal Verification...
  - Proving or disproving the correctness of a system with respect to a certain formal specification or property
- High degree of reliability and robust evidence for regulators

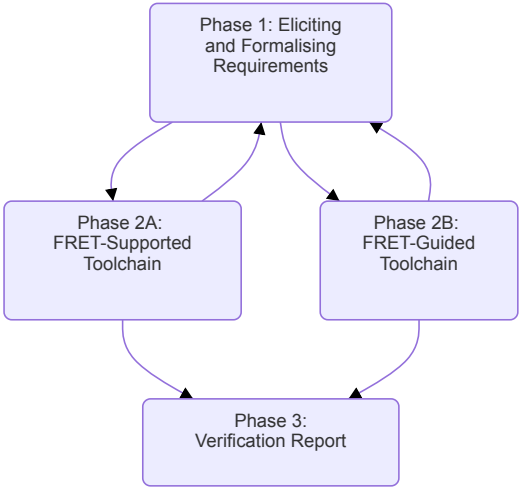
Formalising Requirements

- Formal Methods...
  - Broad group of mathematical approaches to software and system development
  - Support rigorous specification, design and verification
- Formal Verification...
  - Proving or disproving the correctness of a system with respect to a certain formal specification or property
- High degree of reliability and robust evidence for regulators
- There are 2 broad categories of formal method:
  1. Model-checkers exhaustively examine state space
  2. Theorem provers provide deductive proof of correctness

Introduction

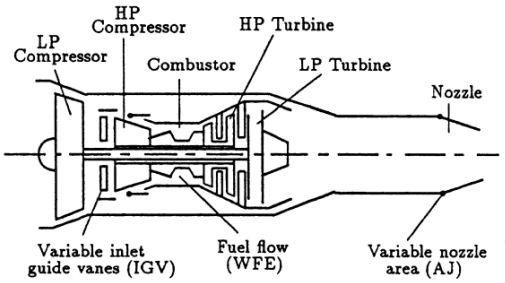
Methodology

- Phase 1: Requirements...
  - Initial requirements
  - Eliciting detail
- Phase 2: Verification...
  - Automatic output from FRET (2A)
  - Guided by requirements in FRET (2B)
- Phase 3: Reporting...
  - Traceability evidence
  - Verification evidence



Requirements for an Aircraft Engine Software Controller

Aircraft Controller



Postlethwaite et al., 1995

Aircraft Engine Software Controller

- FADEC: Full Authority Digital Engine Control
- Responds to pilot input and sensor data
- Monitors and controls the engine...
  - Thrust control
  - Fuel control
  - Power management
  - System health monitoring
  - etc

Aircraft Controller

Requirements of an Aircraft Engine Software Controller

- Industrial partner supplied...
  - 14 English-Language requirements
  - 20 High-level test cases
  - Design in Simulink
- First, manually encoded the requirements into FRET...
- Next, added detail from the test cases...
- Then, regular elicitation meetings with industrial partner

Aircraft Controller

Requirements of an Aircraft Engine Software Controller

- Industrial partner supplied...
  - 14 English-Language requirements
  - 20 High-level test cases
  - Design in Simulink
- First, manually encoded the requirements into FRET...
- Next, added detail from the test cases...
- Then, regular elicitation meetings with industrial partner
- Essential, because we had misunderstood some requirements...

Requirements

Natural-Language Requirement: 1

- *"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)"*

Requirements

Natural-Language Requirement: 1

- *"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)"*
- Questions:
  - 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?

Requirements

Natural-Language Requirement: 13

- *"While tracking pilot commands, controller operating mode shall appropriately switch between nominal and surge/stall prevention operating state "*

Requirements

Natural-Language Requirement: 13

- *"While tracking pilot commands, controller operating mode shall appropriately switch between nominal and surge/stall prevention operating state "*
- Questions:
  - 'Tracking pilot commands' again...
  - What does 'appropriately' mean?
  - What triggers the mode change? Is it related to 'appropriately'?
  - Are these the only operating states?

Using Formal Requirements Elicitation Tool (FRET)

Using FRET

FRET

- Graphical User Interface
- Input language – FRETISH
  - Structured natural-language
  - `scope condition component` shall `timing response`
- Textual and Graphical explanations of the requirement
- Translations...
  - Temporal Logic
  - Contracts for Simulink diagrams

FRETISH Example: Requirement 1

- Requirement 1: *"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)"*

FRETISH Example: Requirement 1

- Requirement 1: *"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)"*

- In FRETISH: `if sensorfaults & trackingPilotCommands Controller` shall satisfy `controlObjectives`

Using FRET

FRETISH Example: Requirement 1

- Requirement 1: *"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)"*
- ~~scope~~ condition component shall ~~timing~~ response
- In FRETISH: `if sensorfaults & trackingPilotCommands Controller` shall satisfy `controlObjectives`

Using FRET

Update Requirement

Requirement ID: UC5_R_1 | Parent Requirement ID | Project: EngineControllerv1.1

Rationale and Comments

Requirement Description

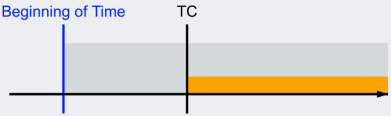
A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "*". For information on a field format, click on its corresponding bubble.

SCOPE | CONDITIONS | COMPONENT* | SHALL* | TIMING | RESPONSES*

if sensorfaults & trackingPilotCommands ControlledSystem shall satisfy controlObjectives

ASSISTANT | TEMPLATES | GLOSSARY

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

FRETISH Example: Requirement 13

- Requirement 13: *"While tracking pilot commands, controller operating mode shall appropriately switch between nominal and surge/stall prevention operating state "*

Using FRET

FRETISH Example: Requirement 13

- Requirement 13: *"While tracking pilot commands, controller operating mode shall appropriately switch between nominal and surge/stall prevention operating state "*

- In FRETISH: `if trackingPilotCommands Controller` shall satisfy `changeMode(nominal) | changeMode(surgeStallPrevention)`

Using FRET

FRETISH Example: Requirement 13

- Requirement 13: *"While tracking pilot commands, controller operating mode shall appropriately switch between nominal and surge/stall prevention operating state "*
- ~~scope~~ condition component shall ~~timing~~ response
- In FRETISH: `if trackingPilotCommands Controller` shall satisfy `changeMode(nominal) | changeMode(surgeStallPrevention)`

Using FRET

Update Requirement

Requirement ID: UC5_R_13 | Parent Requirement ID | Project: EngineControllerv1.1

Rationale and Comments

Requirement Description

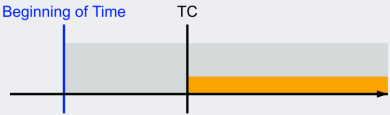
A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "*". For information on a field format, click on its corresponding bubble.

SCOPE | CONDITIONS | COMPONENT* | SHALL* | TIMING | RESPONSES*

if (trackingPilotCommands) Controller shall satisfy (changeMode(nominal)) | (changeMode(surgeStallPrevention))

ASSISTANT | TEMPLATES | GLOSSARY

ENFORCED: in the interval defined by the entire execution. TRIGGER: first point in the interval if ***((trackingPilotCommands))*** is true and any point in the interval where ***((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 = ***((trackingPilotCommands))***, Response = ***((changeMode (nominal)) | (changeMode (surgeStallPrevention)))***.

Diagram Semantics

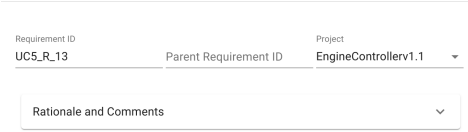
Formalizations

Future Time LTL

Using FRET

What About the Test Cases. . . ?

- FRET can link requirements. . .
  - But no inheritance etc
- Natural-Language Requirements – Parent Requirements
- Test Cases and New Details – Child Requirements
- Details in the paper



Lessons Learnt and Future Improvements

An Industrial Perspective

Industrial Partner Debrief

- Industrial partner was new to FRET...
  - FRETISH requirements were *'much more clear'* than the natural-language requirements
  - *'controlled-natural language with precise semantics is always better than natural-language'*
  - FRET was useful *'because it forces you to think about the actual meaning behind the natural-language requirements'*

Using FRET

Bridging the Communications Gap

- FRETISH provides a stepping-stone between
  - Readable natural-language requirements
  - Fully-formal requirements
- Graphical explanation useful visualisation of requirements...
  - Sanity-checking for both us and our industrial partner
- FRET can also ...
  - Document rationale behind requirement/changes
  - Explainability

Conclusion

Conclusion

Summary

- Formalised requirements from industrial partner...
- via FRETISH as intermediate language
- Elicitation meetings with industrial partner...
  - Clarified requirements
  - Added new information
- Requirements now ready for use in formal verification

Conclusion

Summary

- Formalised requirements from industrial partner...
- via FRETISH as intermediate language
- Elicitation meetings with industrial partner...
  - Clarified requirements
  - Added new information
- Requirements now ready for use in formal verification

Future Work

1. Adding *refactoring* to FRET
2. Adding global types to FRETISH
3. FRET translation to other formal languages

Conclusion

Summary

- Formalised requirements from industrial partner...
- via FRETISH as intermediate language
- Elicitation meetings with industrial partner...
  - Clarified requirements
  - Added new information
- Requirements now ready for use in formal verification

Future Work

1. Adding *refactoring* to FRET
2. Adding global types to FRETISH
3. FRET translation to other formal languages

matt.luckcuck@mu.ie or marie.farrell@mu.ie