Sharper Specs for Smarter Drones: Formalising Requirements with FRET

Oisín Sheridan Leandro Buss Becker Marie Farrell Matt Luckcuck Rosemary Monahan

Department of Computer Science, Maynooth University/Hamilton Institute, Maynooth, Ireland

Automation and Systems Department, Federal University of Santa Catarina, Florianópolis, Brazil

Department of Computer Science, The University of Manchester, Manchester, UK

School of Computer Science, University of Nottingham, Nottingham, UK

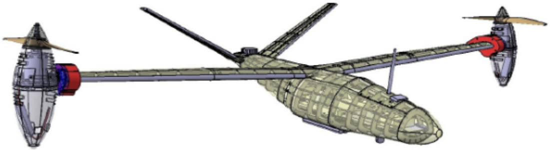
Introduction

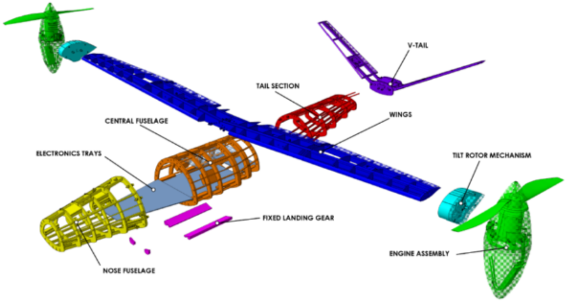
Overview

- We describe the process of formalising the natural-language requirements for a tilt-rotor drone using the Formal Requirements Elicitation Tool (FRET).
- This requirements set evolved over four distinct versions as new information was elicited and incorporated into the FRETish specification.
- Our two concrete outputs are the formalised requirement set, which we will use in our ongoing development and verification of ProVANT; and metrics about the requirements.
- We present guidance for requirements elicitation and formalisation with FRET. We highlight situations where it was difficult to formalise these requirements and describe potential improvements to FRET to address these difficulties.

Case Study – ProVANT Emergentia Tilt-Rotor Drone

Case Study – Tilt-Rotor Drone





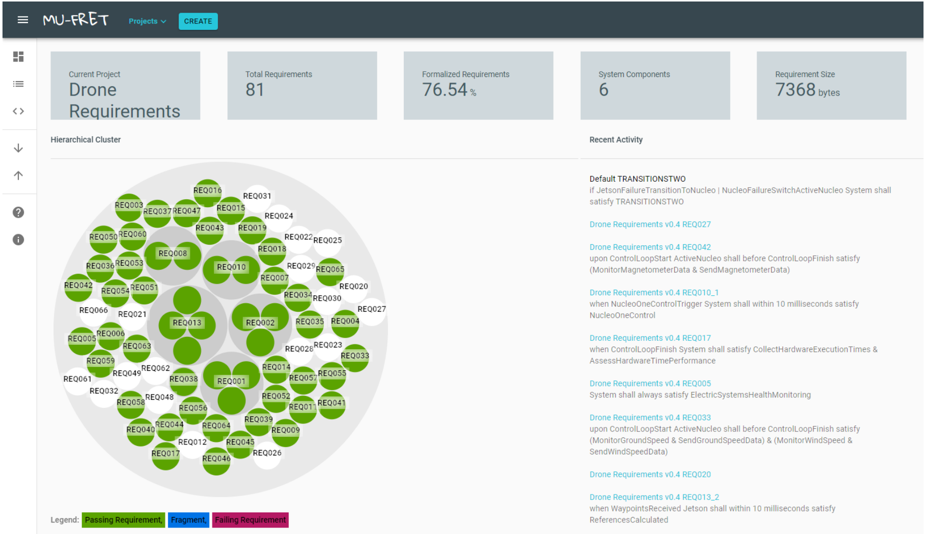
- At the latter stage of development under the ProVANT Emergantia project
- The project is a collaboration among two Brazilian universities, Federal University of Minas Gerais (UFMG) and Federal University of Santa Catarina (UFSC), along with the University of Seville, Spain.

Case Study – Tilt-Rotor Drone

- The drone can perform hovering and Vertical Take-off and Landing (VTOL) manoeuvres, as well as cruise flight as a fixed-wing aircraft.
- The architecture for the Drone's computing system comprises four main components:
  - Raspberry Pi: Gathers sensor data and communicates with the Ground Control Station.
  - Jetson: Processes sensor data and runs the control algorithm.
  - Nucleos: the active nucleo interfaces with the drone's actuators and some sensors. Can also run a backup control algorithm in the case of a failure. There are two nucleos for reliability.
- The set of requirements for the ProVANT Emergentia drone includes aspects related to:
  1. operation features present during simulations and during real executions
  2. remote monitoring configurations
  3. timing constraints associated with the control loop
  4. operation modes under failure conditions

Formalisation with FRET

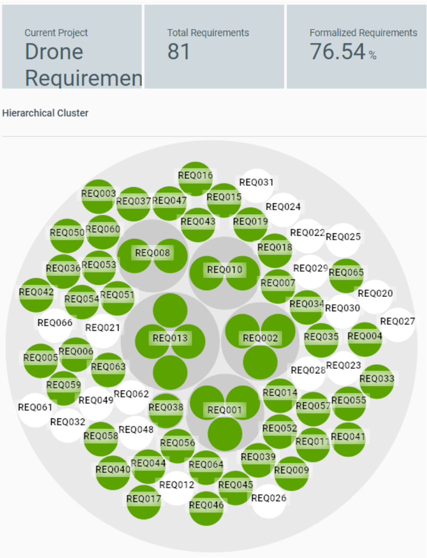
The Formal Requirements Elicitation Tool (FRET)



The Formal Requirements Elicitation Tool (FRET)

FRET

- An open source tool for requirements engineering developed by NASA
- Requirements are written in a structured natural-language called FRETish
- FRET provides automated translations from FRETish to CoCoSpec contracts, which can be verified with the Kind2 model checker, and Copilot runtime monitors
- Formalised requirements are indicated in green, while those in white have not been formalised



Update Requirement

Requirement ID

REQ033

Parent Requirement ID

Project

Drone Requirements v4

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.

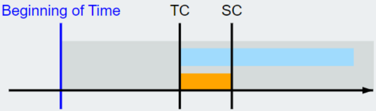
SCOPE CONDITIONS COMPONENT* SHALL* TIMING RESPONSES*

upon ControlLoopStart ActiveNucleo shall before ControlLoopFinish satisfy (MonitorGroundSpeed & SendGroundSpeedData) & (MonitorWindSpeed & SendWindSpeedData)

SEMANTICS

ASSISTANT TEMPLATES GLOSSARY

ENFORCED: in the interval defined by the entire execution. TRIGGER: first point in the interval if ***(ControlLoopStart)*** is true and any point in the interval where ***(ControlLoopStart)*** becomes true (from false). REQUIRES: for every trigger, RES must hold at least once strictly before the state where the stop condition holds. If the stop condition never occurs in the interval, RES does not need to hold. If the stop condition holds at the trigger, the requirement is not satisfied.



TC = ***(ControlLoopStart)***, SC = ***(ControlLoopFinish)***, Response = ***((MonitorGroundSpeed & SendGroundSpeedData) & (MonitorWindSpeed & SendWindSpeedData))***.

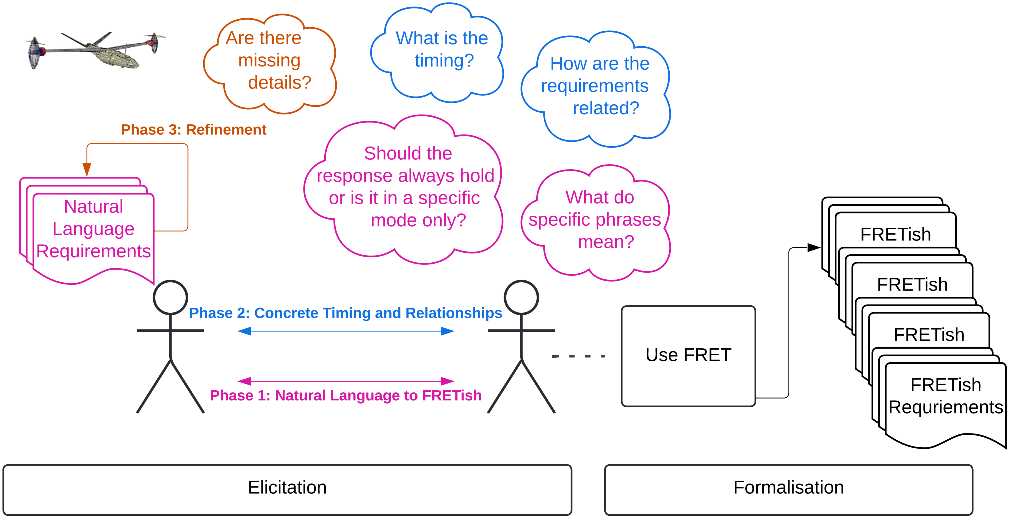
Diagram Semantics

Formalizations

Future Time LTL

Formalising the Requirements

Formalising the Requirements – Methodology



Formalising the Requirements – Starting from Natural Language

ID	Original English text
REQ001	Allow failure simulations between nucleo/jetson and nucleo/nucleo
REQ013	Send references
REQ016	Run each simulation loop within 10ms
REQ018	Present the total time spent
REQ019	Present the time spent in the control algorithm
REQ024	Work on any operational system
REQ033	Monitor linear velocities (ground speed and relative wind speed)

- The formalisation began with a set of 66 natural-language requirements for the tilt-rotor drone.
- Each requirement consists of an ID number and a short description
- Each one also had additional metadata: a *Category* of Functional or Non-Functional, a *Feasibility* ranging from Feasible to Unknown to Unfeasible, and a *Group*.

Formalising the Requirements – FRETish Version 1

- For the first iteration, we mapped the drone's original natural-language requirements one-to-one into FRETISH, matching the vocabulary between both versions where possible.
- Many of these were of a simple form, such as "`System` shall `always/eventually` `satisfy [variableName]`"
- Twenty of the 66 requirements were not translated in this initial set
- This stage gave us a clearer picture of what information we would need for a more robust formalisation

ID	FRETish First Iteration
REQ001	`System` shall `always` `satisfy` `AllowNucleoJetsonSimulation & AllowNucleoNucleoSimulation`
REQ016	`when SimulationLoopStart` `System` shall `within 10 milliseconds` `satisfy SimulationLoopFinish`
REQ033	`System` shall `always` `satisfy MonitorGroundSpeed & MonitorWindSpeed`

Formalising the Requirements – FRETish Version 2

- We discussed the ambiguities that we found and consulted with the use case provider for additional detail, leading to a number of updates.
- The largest update was a distinction between running the system in simulation versus in the real world. We created a `SimulationMode` scope variable in nine requirements.
- 27 requirements gained a scope of "`while MonitoringEnabled`", so that Data Monitoring would be optional when running the system.
- We added the first child requirements to the set: REQ008_1 & _2, and REQ010_1 & _2.

ID	FRETish Second Iteration
REQ001	`in SimulationMode System` shall `eventually satisfy NucleoJetsonFailure|NucleoNucleoFailure`
REQ033	`while MonitoringEnabled System` shall `always satisfy MonitorGroundSpeed & MonitorWindSpeed`

Formalising the Requirements – FRETish Version 2

Child requirements

We use child requirements to express how a requirement should apply to different components or in different situations.
REQ008_1 specifies that the Raspberry Pi should transmit the data to the ground station, while REQ008_2 states that the Jetson should save the data and send it to the active Nucleo for evaluation

ID	**Final FRETish text**
REQ008	Save any desired simulation data `after SimulationMode System` shall `within 100 ticks satisfy SimulationDataSaved`
REQ008_1	`after SimulationMode Raspberry` shall `within 100 ticks satisfy GroundStationReceivedData`
REQ008_2	`after SimulationMode Jetson` shall `within 100 ticks satisfy SimulationDataRecorded & NucleoReceivedData`

Formalising the Requirements – FRETish Version 4

Formalising the Requirements – Beginning and End

ID	Final FRETish text
REQ001	Allow failure simulations between nucleo/jetson and nucleo/nucleo `in SimulationMode whenever SimulateFailureTransitions System` shall `eventually satisfy JetsonFailureTransitionToNucleo | NucleoFailureSwitchActiveNucleo`
REQ018	The control loop will complete within 12 milliseconds `upon ControlLoopStart System` shall `within 12 milliseconds satisfy ControlLoopFinish`
REQ019	The control algorithm will complete within 6 milliseconds `upon ControlAlgorithmStart System` shall `within 6 milliseconds satisfy ControlAlgorithmFinish`
REQ033	Monitor linear velocities (ground speed and relative wind speed) `upon ControlLoopStart ActiveNucleo` shall `before ControlLoopFinish satisfy (MonitorGroundSpeed & SendGroundSpeedData) & (MonitorWindSpeed & SendWindSpeedData)`