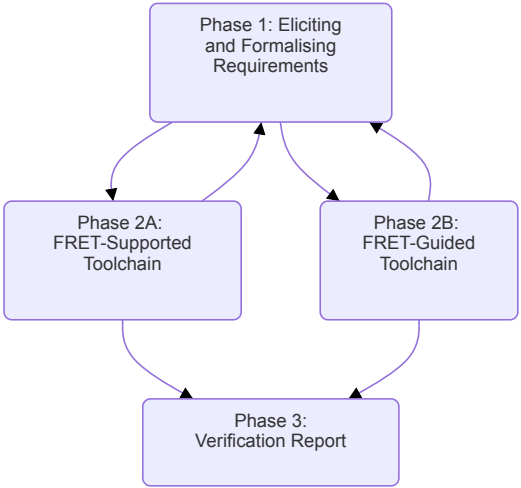
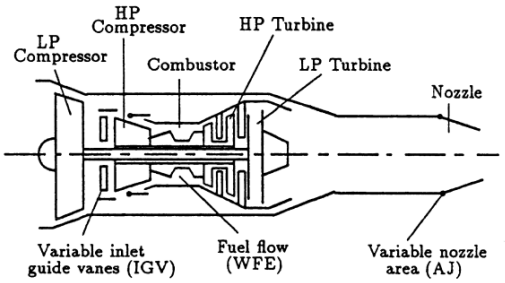
Introduction

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



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

Requirements Elicitation

Requirements

FRETISH Example: Requirement 1

- 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)"*
- 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

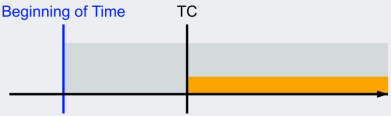
A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "*". For information on a field format, click on its corresponding bubble.

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

Requirements

FRETISH Example: Requirement 13

- Natural-Language 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

Update Requirement

Requirement ID: UC5_R_13

Parent Requirement ID

Project: EngineControllerv1.1

Rationale and Comments

Requirement Description

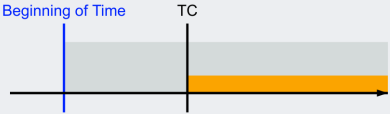
A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "*". For information on a field format, click on its corresponding bubble.

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

Refactoring – MU-FRET

Refactoring Requirements

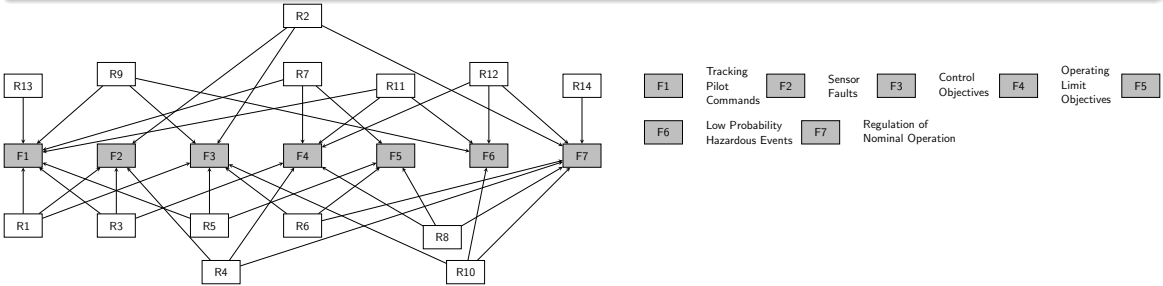
Analysis: Aircraft Engine Controller Requirements

- Traceability: one-to-one mapping in FRETISH

  `scope condition component` shall `timing response`

  UC5_R_1: `if((sensorFaults)&(trackingPilotCommands)) Controller` shall `satisfy (controlObjectives)`.

- Repetition of *fragments*.

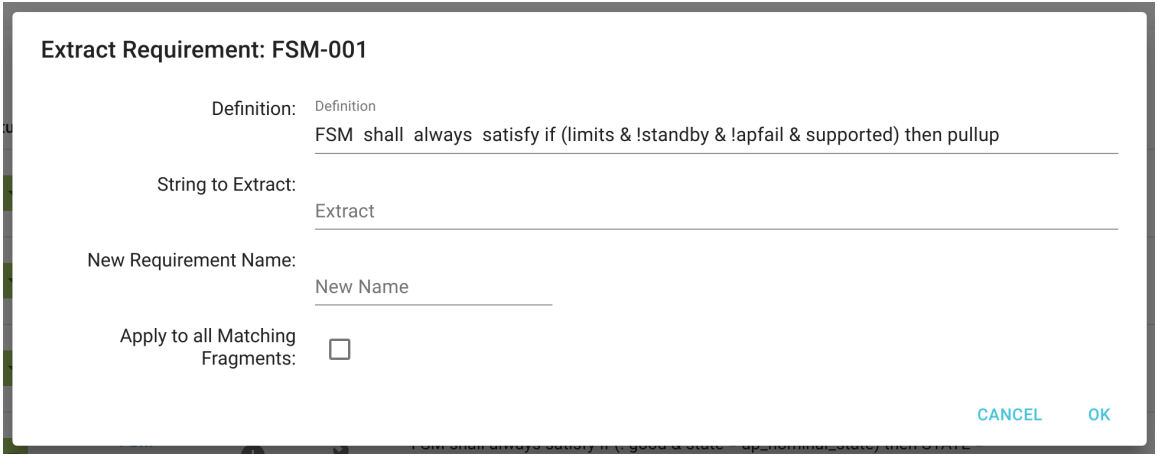


Refactoring Requirements

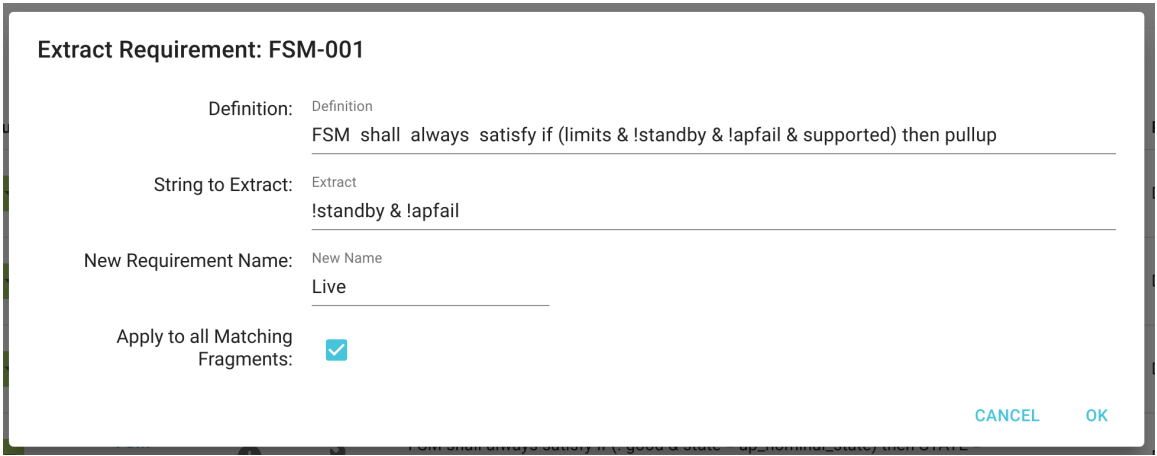
Requirements: Demo-FSM

Status	ID			Summary	Project
	FSM-001			FSM shall always satisfy if (limits & !standby & !apfail & supported) then pullup	Demo-FSM
	FSM-002			FSM shall always satisfy if (standby & state = ap_transition_state) then STATE = ap_standby_state	Demo-FSM
	FSM-003			FSM shall always satisfy if (state = ap_transition_state & good & supported) then STATE = ap_nominal_state	Demo-FSM

Refactoring Requirements



Refactoring Requirements



Refactoring Requirements

Refactoring Requirements

Status	ID ↑			Summary
	FSM-001			FSM shall always satisfy if (limits & Live & supported) then pullup

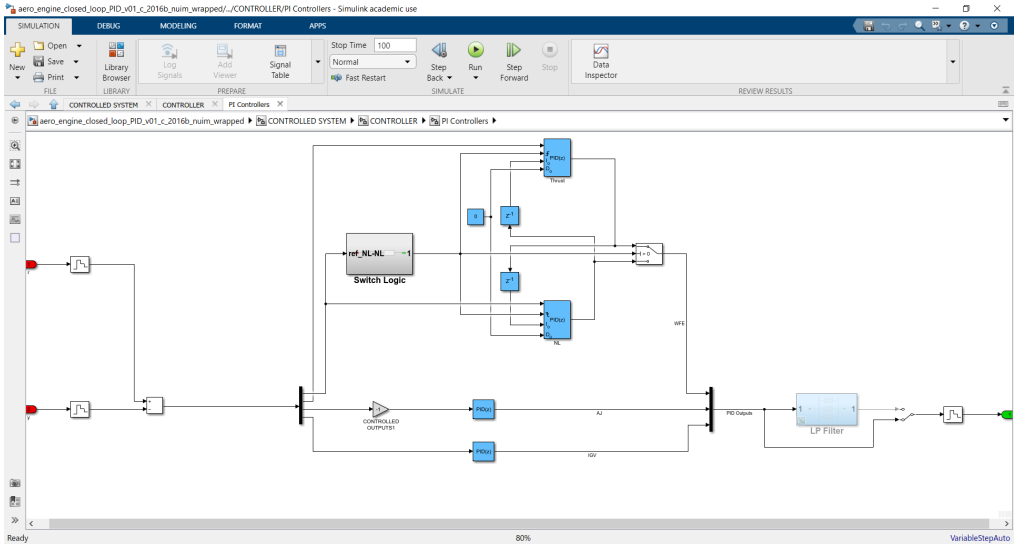
Refactoring Requirements

ID	Fragment Name	№ of (Re)Definitions	
		Before Refactoring	After Refactoring
F1	*Sensor Faults*	8	1
F2	*Tracking Pilot Commands*	13	1
F3	*Control Objectives*	18	1
F4	*Regulation Of Nominal Operation*	14	1
F5	*Operating Limit Objectives*	6	1
F6	*Mechanical Fatigue*	8	1
F7	*Low Probability Hazardous Events*	8	1
F8	*Active*	28	1
F9	*Not Active*	28	1
Total (Re)Definitions		132	9

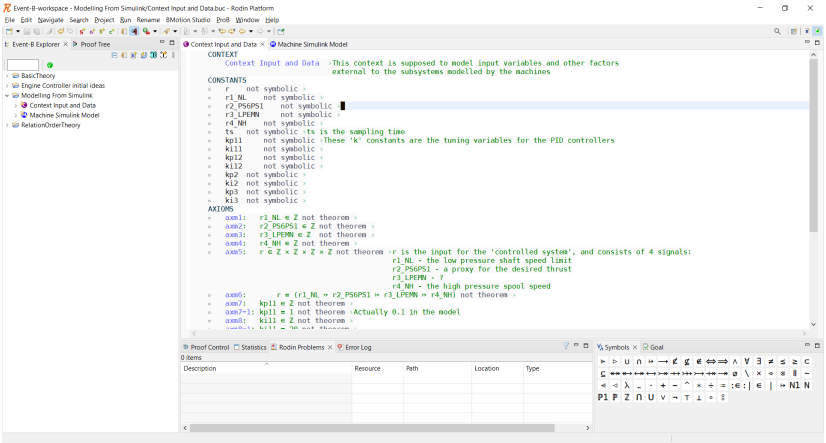
Table 1: The number of times each fragment's definition occurs in a child requirement.

FRET-Guided Toolchain

Modelling in Event-B



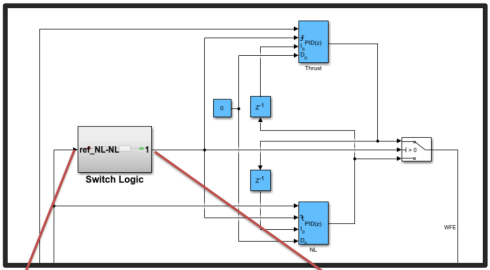
Modelling in Event-B

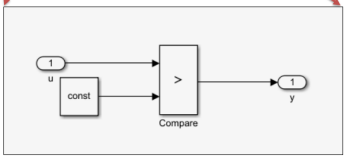


Event-B

- Set-based modelling language
- Development done using the Rodin platform

Modelling in Event-B






```
MACHINE
    Simulink Machine
INVARIANTS
  inv1:   rl_NL ∈ ℤ not theorem
  inv3:   yl_NL ∈ ℤ not theorem
  inv7:   ul_WFE ∈ ℤ not theorem
  inv10:  diff1 ∈ ℤ not theorem
  inv14:  Switch_Th ∈ ℤ not theorem
  inv15:  Switch_Value ∈ BOOL not theorem

  inv21:  S_diffs_summed ∈ BOOL not theorem
  inv22:  S_Switch_set ∈ BOOL not theorem
  inv18:  S_WFE_set ∈ BOOL not theorem
EVENTS
  INITIALISATION:    not extended ordinary
    END
```

```
Switch_Logic_True: not extended ordinary
WHERE
  grd1:   S_diffs_summed = TRUE not theorem
  grd2:   S_Switch_set = FALSE not theorem
  grd3:   S_WFE_set = FALSE not theorem

  grd4:   diff1 > Switch_Th not theorem
THEN
  act1:   Switch_Value ≔ TRUE

  act3:   S_Switch_set ≔ TRUE
END

Switch_Logic_False:    not extended ordinary
WHERE
  grd1:   S_diffs_summed = TRUE not theorem
  grd2:   S_Switch_set = FALSE not theorem
  grd3:   S_WFE_set = FALSE not theorem

  grd4:   diff1 ≤ Switch_Th not theorem
THEN
  act1:   Switch_Value ≔ FALSE

  act3:   S_Switch_set ≔ TRUE
END
```

FRET-Supported Toolchain

CoCoSim and CoPilot

CoCoSim: Contract based **Co**mpositional verification of **Sim**ulink models.

- FRET can generate CoCoSpec assume-guarantee contracts for Simulink blocks.
- CoCoSpec contracts are added to the Simulink diagram.
- Contracts are checked during simulations of the diagram using the Kind2 model checker.

CoPilot: Runtime monitoring framework.

- Copilot allows users to specify monitors and compile them to hard real-time C code.
- Supported by the Ogma tool, FRET users can automatically generate CoPilot monitors.

CoCoSpec Variable Mapping

Controller EXPORT

Corresponding Model Component

aero_engine_closed_loop_PID_v01_c_2016b/PI Controllers IMPORT

FRET Variable Name ↑	Model Variable Name	Variable Type	Data Type	Description
abs		Internal	integer	
changeMode	Out1	Output	double	
controlObjectives		Internal	boolean	
diff		Internal	integer	
E		Internal	integer	
e		Internal	integer	
i		Internal	integer	

Generated CoCoSpec Contract

UC5_R_3

```
(* Req text: if (sensorfaults) & (trackingPilotCommands)
   Controller shall satisfy (operatingLimitObjectives)  *)

guarantee ‘‘UC5_R_3’’ ((H( not (( sensorfaults ) and ( trackingPilotCommands ))))
   or ( not (SI( ((( sensorfaults ) and ( trackingPilotCommands ))
   and ((pre ( not (( sensorfaults ) and ( trackingPilotCommands )))) or FTP)),
   ( not (( operatingLimitObjectives ))) ))));
```

LTL and CoCoSpec Operators in UC5_R_3

- `H`: historically
- `SI`: since inclusive
- `FTP`: first time point of trace
- `pre`: previous value

Attach Contract to Simulink Model