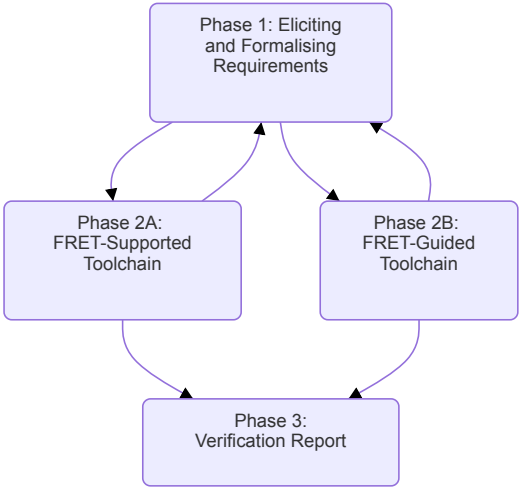
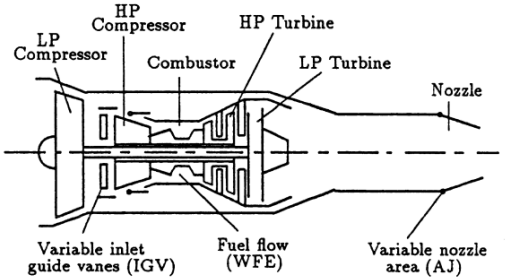
# Introduction

## 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

## 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)"*

## 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

## Update Requirement

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

Rationale and Comments

Requirement Description

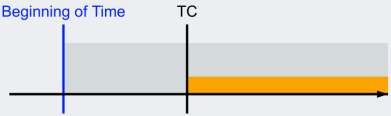
A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "*". For information on a field format, click on its corresponding bubble.

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

## Natural-Language Requirement: 13

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

## 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

## Update Requirement

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

Rationale and Comments

Requirement Description

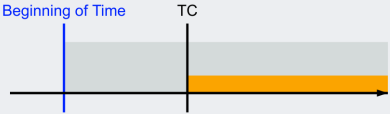
A requirement follows the sentence structure displayed below, where fields are optional unless indicated with "*". For information on a field format, click on its corresponding bubble.

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 Requirements

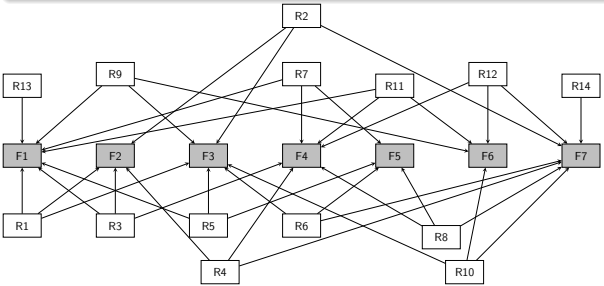
## Analysis: Aircraft Engine Controller Requirements

- Traceability: one-to-one mapping in FRETISH

  `scope condition component` shall `timing response`

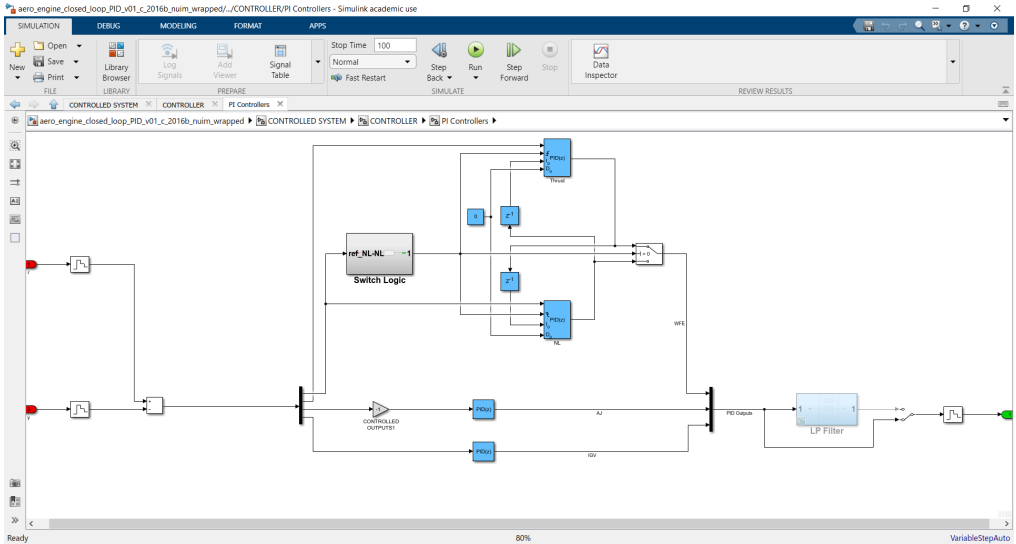
  UC5_R_1: `if((sensorFaults)&(trackingPilotCommands)) Controller` shall satisfy `(controlObjectives)`.

- Repetition of *fragments*.




F1 Tracking Pilot Commands
F2 Sensor Faults
F3 Control Objectives
F4 Operating Limit Objectives
F5
F6 Low Probability Hazardous Events
F7 Regulation of Nominal Operation

# Modelling in Event-B



# Modelling in Event-B

Context Input and Data | Machine Simulink Model ×

```
sum_diff:   not extended ordinary ›Simulates the add block
WHERE
  grd1:   signal_controller_Start = TRUE not theorem ›
  grd2:   signal_diffs_summed = FALSE not theorem ›
THEN
  act1:   diff1 ≔ r1_NL − y1_NL ›
  act2:   diff2 ≔ r2_PS6PS1 − y2_PS6PS1 ›
  act3:   diff3 ≔ r3_LPEMN − y3_LPEMN ›
  act4:   diff4 ≔ r4_NH − y4_NH ›
  act5:   signal_controller_Start ≔ FALSE ›
  act6:   signal_diffs_summed ≔ TRUE ›
END

Switch Logic_True: not extended ordinary ›
WHERE
  grd1:   signal_diffs_summed = TRUE not theorem ›
  grd2:   signal_Switch_Value_set = FALSE not theorem ›
  grd3:   signal_WFE_set = FALSE not theorem ›

  grd4:   diff1 > 1 not theorem ›
THEN
  act1:   Switch_Value ≔ TRUE ›

  act3:   signal_Switch_Value_set ≔ TRUE ›
END

Switch Logic_False:    not extended ordinary ›
WHERE
  grd1:   signal_diffs_summed = TRUE not theorem ›
  grd2:   signal_Switch_Value_set = FALSE not theorem ›
  grd3:   signal_WFE_set = FALSE not theorem ›

  grd4:   diff1 ≤ 1 not theorem ›
THEN
  act1:   Switch_Value ≔ FALSE ›
```