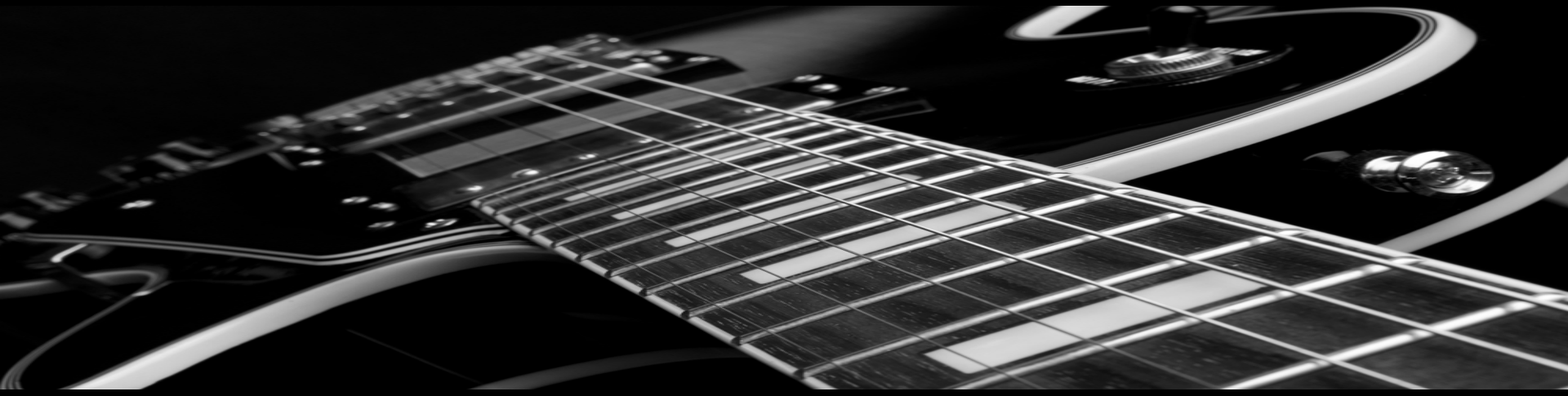
Capturing & Analyzing Requirements with FRET

Dimitra Giannakopoulou, Anastasia Mavridou, Tom Pressburger, Johann Schumann



language used to write reqs

Lockheed Martin Cyber-Physical System Challenge, component FSM:

- Exceeding sensor limits shall latch an autopilot pullup when the pilot is not in control (not standby) and the system is supported without failures (not apfail).

every time these conditions hold or only when they **become** true?

- The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.
- The autopilot shall change states from NOMINAL to MANEUVER when the sensor data is not good.
- The autopilot shall change states from NOMINAL to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from MANEUVER to STANDBY when the pilot is in control (standby) and sensor data is good.
- …

are these requirements consistent? does my model/code satisfy them?

language formal analysis tools understand

```
var autopilot: bool = (not standby) and supported and (not
    apfail);
var pre_autopilot: bool = false -> pre autopilot;
var pre_limits: bool = = false -> pre limits;
guarantee "FSM-001v2" S((((((autopilot and pre_autopilot and
    pre_limits) and (pre ( not (autopilot and pre_autopilot and
    pre_limits)))) or ((autopilot and pre_autopilot and
    pre_limits) and FTP)) => (pullup)) and FTP), ((((autopilot
    and pre_autopilot and pre_limits) and (pre ( not (autopilot
    and pre_autopilot and pre_limits)))) or ((autopilot and
    pre_autopilot and pre_limits) and FTP)) => (pullup)));
```

FRET bridges the gap

- do you speak Fretish?: an extensible grammar defines a restricted natural language with unambiguous semantics
- requirements made up of fields for scope, conditions, component, timing, response; help users think of all aspects
- explanations of the formal semantics in various forms: natural language, diagrams, interactive simulation
- compositional (hence maintainable and extensible) generation of formulas from requirement fields for analysis tools
- checks consistency of requirements and provides feedback
- connects requirements to Simulink models for verification with Cocosim and Simulink Design Verifier

welcome to FRET!

https://github.com/NASA-SW-VnV/fret

FRET

FRET Projects CREATE

Total Projects	Total Requirements	Formalized Requirements	System Components	Requirement Size
4	119	92.44 %	23	10462 bytes

Hierarchical Cluster

Recent Activity

LM_requirements EUL-001
"This requirement is the parent of the EUL-001 subrequirements"

LM_requirements AP-003
"This requirement is the parent requirement that summarizes all 003 reqs"

LM_requirements FSM-006
FSM_Autopilot shall always satisfy (state = ap_maneuver_state & standby & good) => STATE = ap_standby_state

LM_requirements EUL-001H
Euler shall always satisfy DCM321_32 = (- SinPhi * CosPsi) + (CosPhi * SinTheta * SinPsi)

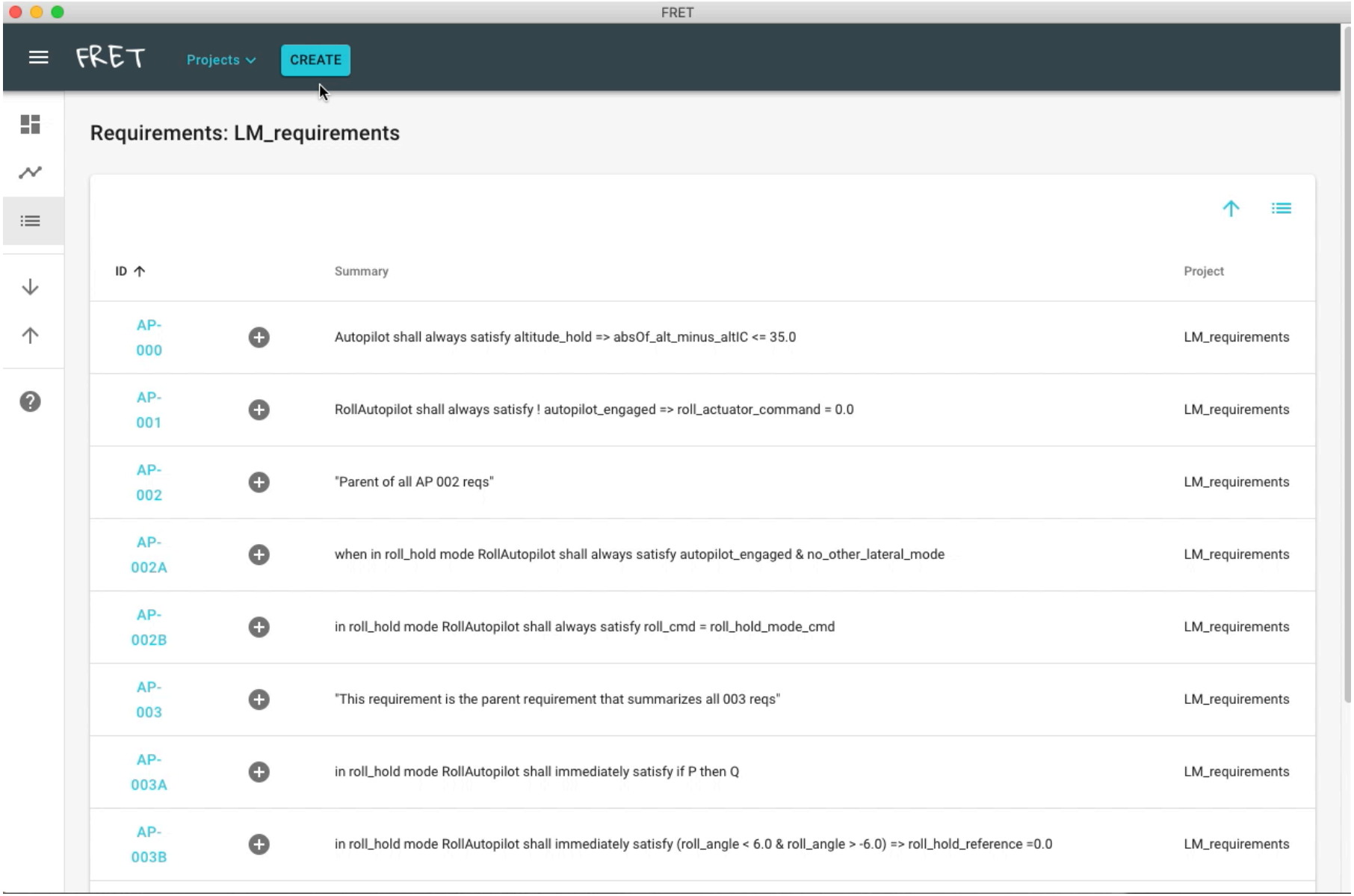
LM_requirements EUL-001B
Euler shall always satisfy DCM321_12 = CosTheta * SinPsi

LM_requirements AP-003C
in roll_hold mode RollAutopilot shall immediately satisfy abs_roll_angle >= 30.0 => roll_hold_reference = 30.0 * sign(roll_angle)

LM_requirements AP-008B
in hdg_hold mode RollAutopilot shall always satisfy roll_cmd = hdg_hold_mode_cmd

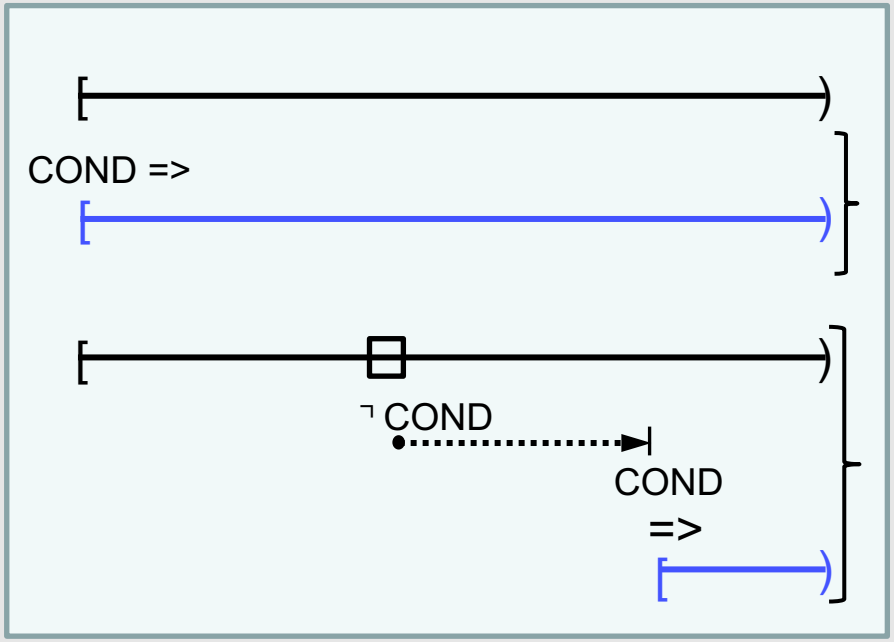
LM_requirements EUL-002B
Euler shall always satisfy R2_21 = VI_1 * R_21 + VI_2 * R_22 + VI_3 * R_23

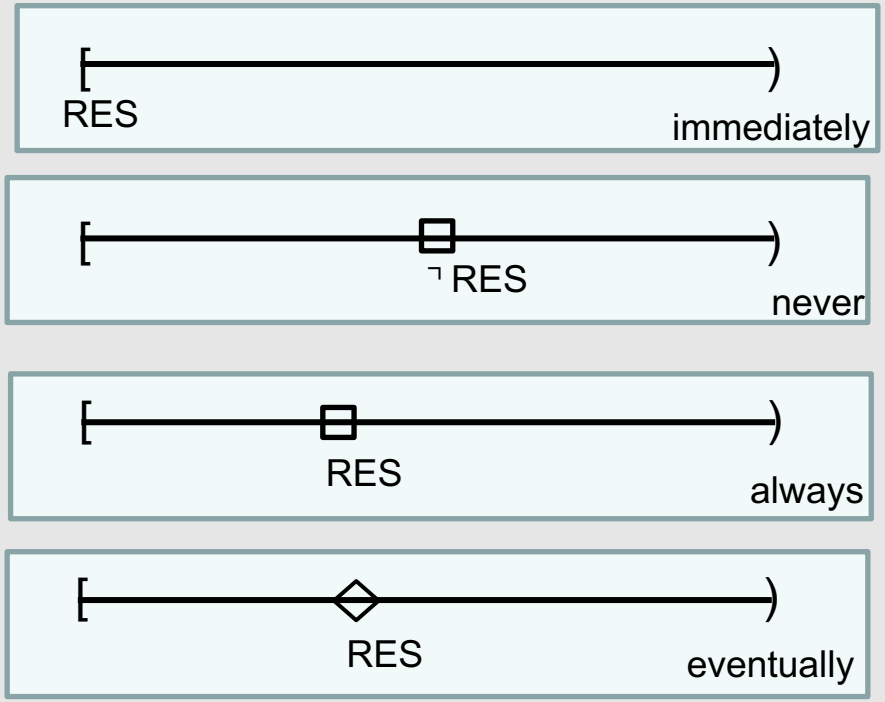
let’s speak Fretish



FRET is rigorous and extensible

Everything is built in a modular / compositional way based on the requirement fields: semantics, formulas, diagrams, test oracles





Recently added timing options until and before, as well as a different semantic notion of triggering conditions.

Dimitra Giannakopoulou, Thomas Pressburger, Anastasia Mavridou, Johann Schumann: **Generation of Formal Requirements from Structured Natural Language.** REFSQ 2020

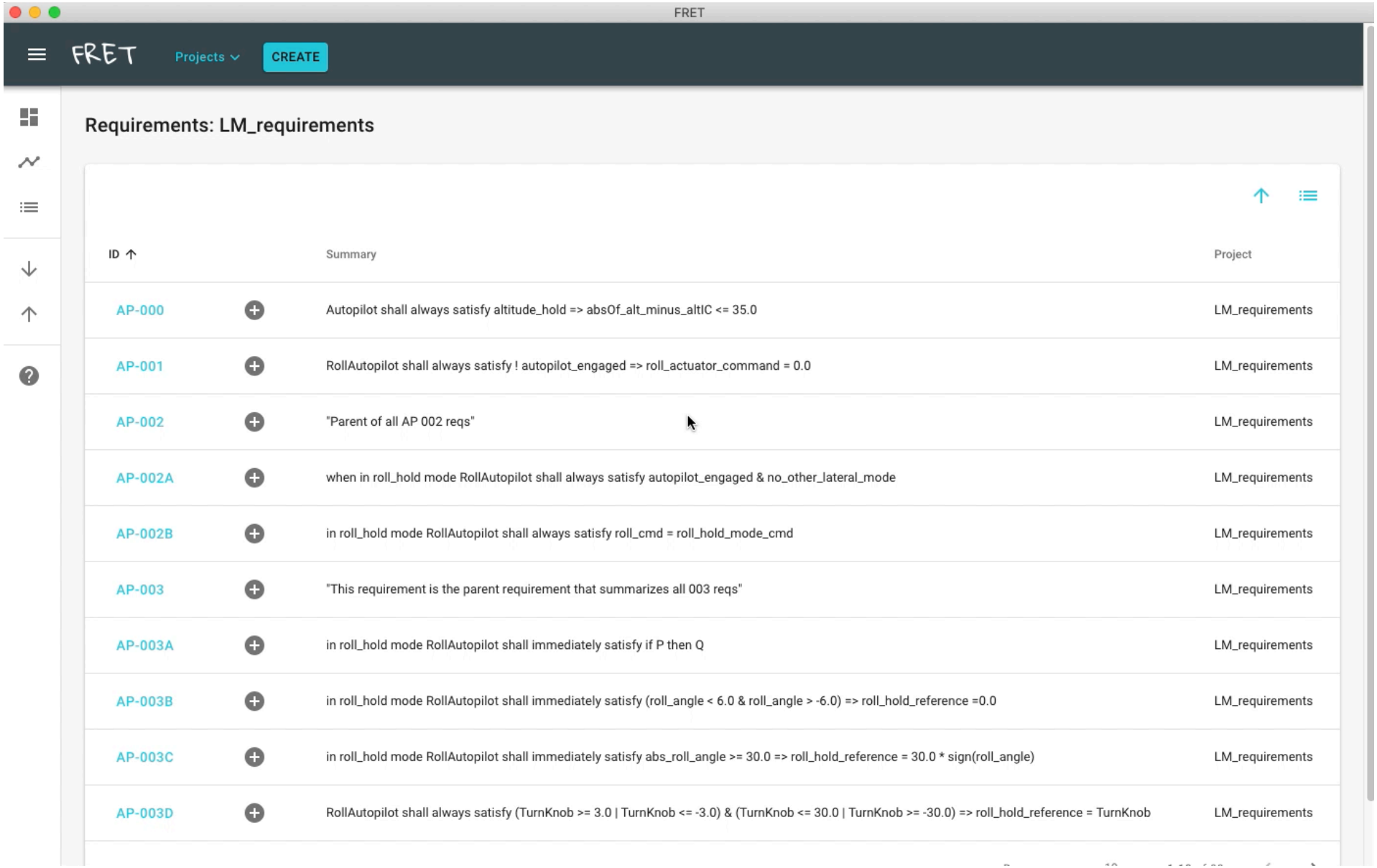
templates

templates

Lockheed Martin Cyber-Physical System Challenge, component FSM:

- The autopilot shall change states from TRANSITION to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.
- The autopilot shall change states from NOMINAL to MANEUVER when the sensor data is not good.
- The autopilot shall change states from NOMINAL to STANDBY when the pilot is in control (standby).
- The autopilot shall change states from MANEUVER to STANDBY when the pilot is in control (standby) and sensor data is good.
- …

brand new feature



template definition (currently)

```
//============ CHANGE STATE ============
ed.newTemplate ("template-change-state","Change State");
ed.templateSummary ("This template describes how the state of a finite-state-machine component changes. \
It describes the input state and some conditions based on which the change must occur. \
The corresponding output state must reflect the required change. \
The input and output states have a pre - post- relationship")
ed.templateStructure('[component] shall always satisfy if ([input_state] & [condition]) then [output_state]')

ed.fieldDescription('component', "Specifies the component of the system that the requirement applies to.")
ed.addOption('component', 'component',"Replace the text by the name of the target component")

ed.fieldDescription('input_state', "Specifies the value of the input state that may need to change.")
ed.addOption('input_state', 'state = value',"The input state value is determined")

ed.fieldDescription('condition', "The condition under which the change is triggered. \
Usually expressed in terms of a predicate, the negation of a predicate, or a conjunction. ")
ed.addOption('condition', 'predicate',"Predicate is described by name")
ed.addOption('condition', '! predicate', "Predicate should not hold")
ed.addOption('condition', 'predicate1 & predicate2', "Conjunction")

ed.fieldDescription('output_state', "Specifies the value of the output state, reflecting \
the new value of the input state .")
ed.addOption('output_state', 'STATE = value',"The output state value is determined")

ed.addExample("[FSM_Autopilot] shall always satisfy \
if ([state = ap_standby_state] & [! standby & ! apfail]) then [STATE = ap_transition_state]")
```

analysis

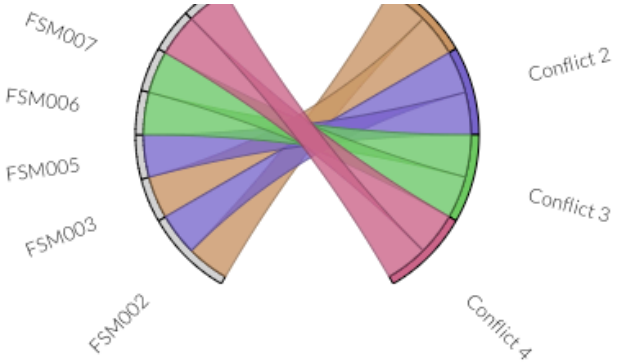
requirements consistency (realizability)

FSM_Autopilot

The autopilot shall change states from TRANSITION to STANDBY when the pilot is in control (standby).

The autopilot shall change states from TRANSITION to NOMINAL when the system is supported and sensor data is good.

inputs: standby=true; supported=true; good=true; output: state=nominal



CLOSE

FRET Variable Name | Model Variable Name | Variable Type* | Data Type* | Description

connecting to Simulink model

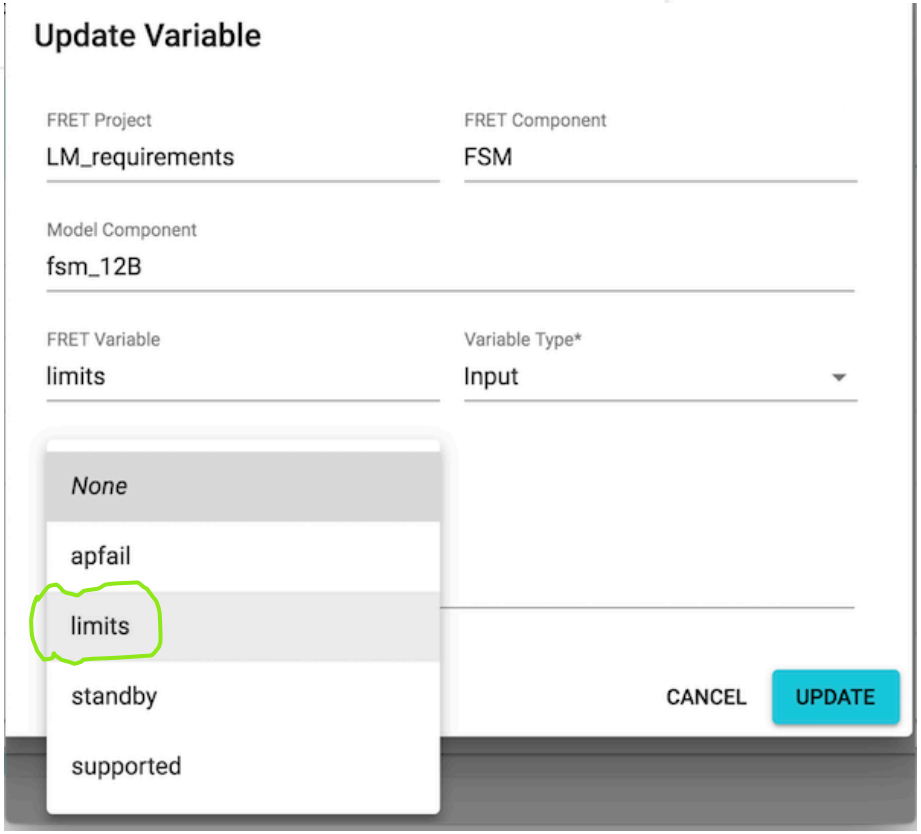
FSM

FRET Component: FSM

Corresponding Model Component: fsm_12B

Simulink component

FRET Variable Name	Model Variable Name	Variable Type*	Data Type*	Description
AUTOPILOT		Internal	boolean	
LIMITS	limits	Input	boolean	
PULLUP	pullup	Output		

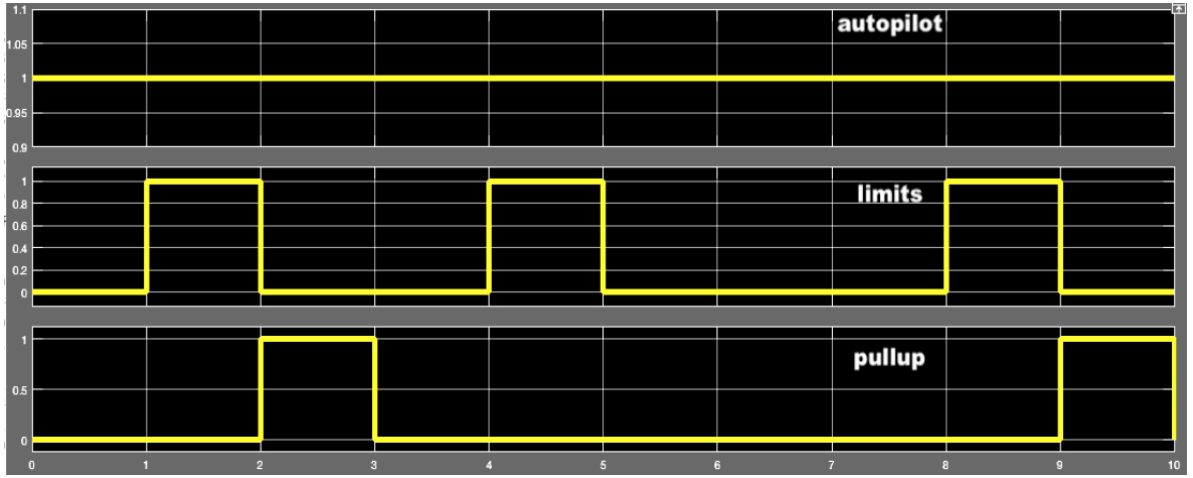


cocosim analysis

verification of Simulink model

1. detection: FSM requirement 1 violation
2. explanation through counterexample and simulation:

“Exceeding sensor limits shall latch an autopilot pullup when the pilot is not in control (not standby) and the system is supported without failures (not apfail).”



A. Mavridou, H. Bourbouh, P.L. Garoche, D. Giannakopoulou, T. Pressburger, J. Schumann: **Bridging the Gap Between Requirements and Simulink Model Analysis.** REFSQ 2020 Posters

summary and future work

- FRET tries to bridge the gap between intuitive capture of requirements and formal languages needed for analysis
- goal: combine formal rigor with usability
- currently in the hands of several projects: Starling, Boeing, GE
- researching into bringing natural language requirements into FRET, providing customization capabilities, providing help with requirements repair
- user feedback is extremely valuable
- available open-source: `https://github.com/NASA-SW-VnV/fret`
- contact: `fret@lists.nasa.gov`

Other contributors: Andreas Katis, David Kooi, Julian Rhein, Nija Shi, Tanja de Jong, Hank Bushnell