



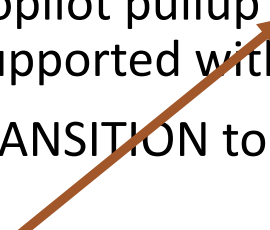
Requirements with FRET

Anastasia Mavridou

KBR / NASA Ames Research Center

how developers write requirements

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

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

- 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?

what formal analysis tools understand

Lockheed Martin Cyber-Physical System Challenge, component FSM:

```
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

- **Captures** requirements with an extensible grammar that defines a restricted natural language with **unambiguous semantics**
- **Explains** formal **semantics** in various forms: natural language, diagrams, interactive simulation
- **Assists** in writing requirements through requirement **templates**
- **Formalizes** requirements in a **compositional** (hence maintainable and extensible) manner
- **Checks consistency** of requirements and provides feedback
- **Connects** with **analysis tools** and **exports verification code**
 - ✓ for model checking Simulink models with CoCoSim
 - ✓ for model checking Lustre code with Kind2
 - ✓ for runtime analysis of C programs with Copilot

☐

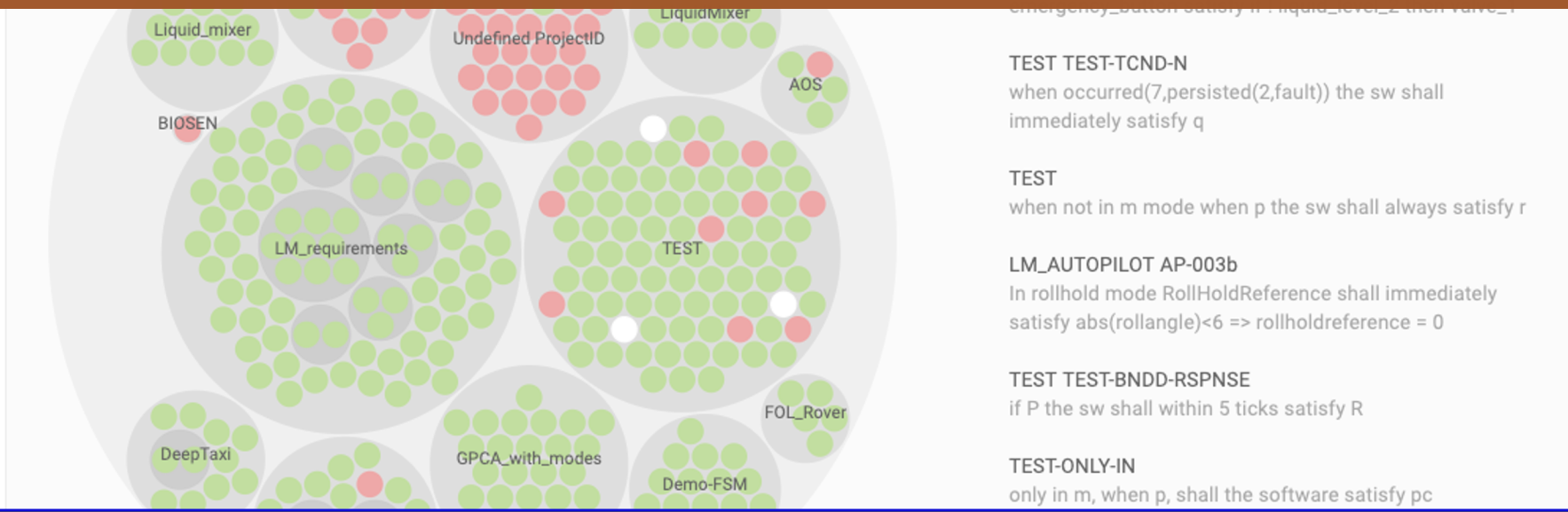
☰

<>

Total Projects 19	Total Requirements 356	Formalized Requirements 80.34 %	System Components 52	Requirement Size 29378 bytes
----------------------	---------------------------	------------------------------------	-------------------------	---------------------------------

Welcome to FRET

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



Team: Dimitra Giannakopoulou, Andreas Katis, Tom Pressburger, Johann Schumann, Khanh Trinh
Alumni: David Bushnell, Tanja DeJong, George Karamanolis, David Kooi, Julian Rhein, Nija Shi

Capturing and explaining requirements

Create Requirement

Requirement ID
Example

Parent Requirement ID

Project
LM_requirements

Rationale and Comments

Rationale

Comments
Target mode: cruising.
Requirement: The altitude hold autopilot shall maintain altitude whenever altitude hold is selected

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.

SCOPECONDITIONSCOMPONENT*SHALL*TIMINGRESPONSES*

when in cruising mode the altitude_hold_autopilot shall

Parse Errors: mismatched input '<EOF>' expecting ('.', 'before', 'after', 'satisfy', 'within', 'for', 'until', 'at', 'immediately', 'eventually', 'always', 'never')

SEMANTICS

CANCELCREATE

Scope (optional)

specifies where the requirement must hold: either globally or in intervals defined with respect to a MODE, e.g.,

- before MODE, only before MODE
- in MODE, not in MODE, only in MODE
- after MODE, only after MODE

MODE is a string identifier starting with an upper- or lower-case letter, followed by letters, digits or underscores '_'. The **only** modes mean that when the system is not in the specified relationship to the mode (i.e., the system is not **in/after/before** the mode) the response must not occur. Examples:

- global The system shall always satisfy count >= 0
- After boot mode the system shall immediately satisfy prompt_for_password
- Only after arming mode shall the system eventually satisfy fired
- In landing mode the system shall eventually satisfy decrease_speed
- When not in initialization mode the system shall always satisfy commands_accepted
- Only in landing mode shall the system eventually satisfy landing_gear_down
- Before energized mode the system shall always satisfy energized_indicator_off
- Only before energized mode shall the system eventually satisfy manually_touchable

Bubble Color :

Requirement 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.

Requirement 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.

Requirement templates

≡

FRET

Projects ▾

CREATE

❏

⤴

☰

⬇

⬆

?

Create Requirement

Requirement ID

FSM 002

Parent Requirement ID

Project

LM_requirements

▾

Rationale and Comments

⤴

Rationale

Comments

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

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*

?

component

 shall always satisfy if (

input_state

 &

condition

) then

output_state

SEMANTICS

ASSISTANT

TEMPLATES

Template

Change State

▾

Choose a predefined template

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

Examples:

FSM_Autopilot

 shall always satisfy if (

state = ap_standby_state

 &

! standby & ! apfail

) then

STATE = ap_transition_state

Checking consistency

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.

Checking consistency

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.

State: **TRANSITION**

Checking consistency

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.

State: **TRANSITION**

Condition 1: pilot is in control

Condition 2: system is supported
sensor data is good

Checking consistency

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.

State: **TRANSITION** ✓
Condition 1: pilot is in control ✓
Condition 2: system is supported ✓
 sensor data is good ✓

Checking consistency

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.

State: **TRANSITION** ✓
Condition 1: pilot is in control ✓
Condition 2: system is supported ✓
 sensor data is good ✓
Next state 1: **STANDBY** 
Next state 2: **NOMINAL** 

Checking consistency

VARIABLE MAPPING

REALIZABILITY

System Component *

FSM

☒ Compositional ☐ Monolithic

Timeout (seconds)
900

CHECK

DIAGNOSE

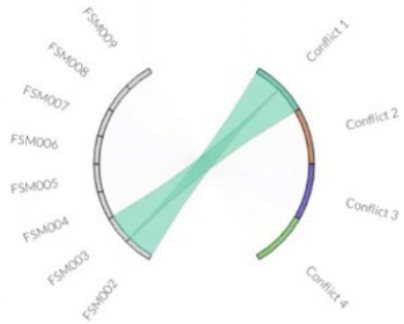
EXPORT

HELP

CC0

CC1

CC2



Counterexample For
Conflict 1

Variable name	Variable type	Step 0
good	bool	true
standby	bool	true
state	real	0
supported	bool	true
STATE	real	2

ID ↑	Summary
FSM002	FSM shall always satisfy (standby & state = ap_transition_state) => STATE = ap_standby_state
FSM003	FSM shall always satisfy (state = ap_transition_state & good & supported) => STATE = ap_nominal_state
ECM001	ECM shall always satisfy (limits & standby & fail & supported) => null

Connection with analysis tools

FRET

Projects

CREATE

Requirement Variables to Model Mapping: Demo-FSM

Export Language *

Autopilot

EXPORT

FSM

EXPORT

Corresponding Model Component

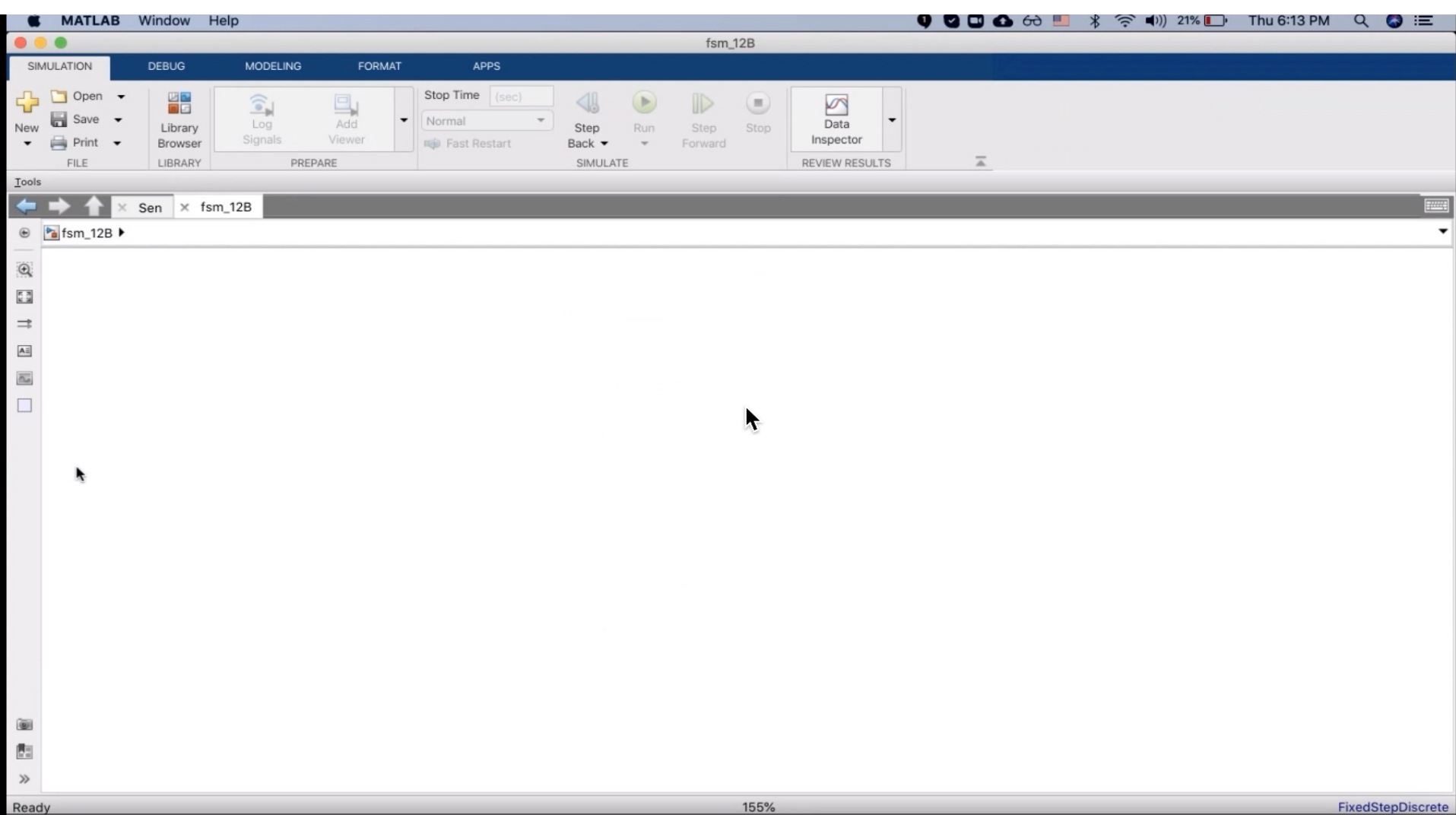
FSM_model

IMPORT

FRET Variable Name ↑	Model Variable Name	Variable Type	Data Type	Description
AP_MANEUVER_STATE		Internal	double	value 2
AP_NOMINAL_STATE		Internal	double	value 1
AP_STANDBY_STATE		Internal	double	value 3
AP_TRANSITION_STATE		Internal	double	value 0
APFAIL	apfail	Input	boolean	
GOOD	good	Input	boolean	
LIMITS	limits	Input	boolean	
PULLUP	pullup	Output	boolean	
REQUEST	request	Input	boolean	
SEN_FAULT_STATE		Internal	double	value 2

Rows per page: 10 1-10 of 18

Connection with analysis tools



Ready for FRETish?



FRET's mission is to provide an intuitive platform for capturing precise requirements, to serve as a portal to a variety of analysis tools, and to support requirements repair based on analysis feedback.

FRET v2.0 coming soon

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

Anastasia Mavridou, Andreas Katis, Dimitra Giannakopoulou, David Kooi, Thomas Pressburger, Michael W. Whalen: *From Partial to Global Assume-Guarantee Contracts: Compositional Realizability Analysis in FRET*. FM 2021.

Dimitra Giannakopoulou, Thomas Pressburger, Anastasia Mavridou, Johann Schumann: *Automated Formalization of Structured Natural Language Requirements*. IST Journal, 2021.

Aaron Dutle, César A. Muñoz, Esther Conrad, Alwyn Goodloe, Laura Titolo, Iván Pérez, Swee Balachandran, Dimitra Giannakopoulou, Anastasia Mavridou, Thomas Pressburger: *From Requirements to Autonomous Flight: An Overview of the Monitoring ICAROUS Project*. FMAS 2020.

Anastasia Mavridou, Hamza Bourbouh, Dimitra Giannakopoulou, Thomas Pressburger, Mohammad Hejase, P-Loïc Garoche, Johann Schumann: *The Ten Lockheed Martin Cyber-Physical Challenges: Formalized, Analyzed, and Explained*. RE 2020.

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

Anastasia Mavridou, Hamza Bourbouh, Pierre-Loïc Garoche, Dimitra Giannakopoulou, Thomas Pressburger, Johann Schumann: *Bridging the Gap Between Requirements and Simulink Model Analysis*. REFSQ 2020.

Thank you