Capturing and Analyzing Requirements with FRET

Anastasia Mavridou
KBR Inc. / NASA Ames Research Center
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).

- 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.

- ...
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).

• 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.

• …

Every time these conditions hold or only when they become true?
What analysis tools understand

```javascript
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));
```
What analysis tools understand

**Temporal logics**

```plaintext

PAST TIME FORMULA
(H ((Lin_roll_hold & (! FTP)) -> (Y (((! (steady_state)) S (((steady_state)) & Fin_roll_hold)) | ((abs Roller_err <= 1.0) S ((abs Roller_err <= 1.0) & ((steady_state)) & ((Y (! (steady_state))) | Fin_roll_hold))))) S (((! (steady_state)) S ((( steady_state)) & Fin_roll_hold)) | ((abs Roller_err <= 1.0) S ((abs Roller_err <= 1.0) & (steady_state) & ((Y (! (steady_state))) | Fin_roll_hold)))))) & (((! Lin_roll_hold) S ((( Lin_roll_hold) & Fin_roll_hold)) -> (((! (steady_state)) S ((( steady_state)) & Fin_roll_hold)) | ((abs Roller_err <= 1.0) S ((abs Roller_err <= 1.0) & (steady_state) & ((Y (! (steady_state))) | Fin_roll_hold)))))) S (((((! (steady_state)) S ((( steady_state)) & Fin_roll_hold)) | ((abs Roller_err <= 1.0) S ((abs Roller_err <= 1.0) & (steady_state) & ((Y (! (steady_state))) | Fin_roll_hold)))))) & Fin_roll_hold)))
```
What types of bugs are found in models and code?

What types of bugs are found in models and code?

Requirements: incomplete, ambiguous/misunderstood, contradictory

Type of bugs

- In models
- In auto-generated code

How do we bridge this gap?

Natural language requirements

Rigorous formalizations
welcome to FRET!

https://github.com/NASA-SW-VnV/fret
The Formal Requirements Elicitation Tool (FRET) Team

Dimitra Giannakopoulou
Tom Pressburger
Khanh Trinh
Andreas Katis
Anastasia Mavridou
Johann Schumann
The Formal Requirements Elicitation Tool (FRET)

Captures requirements

when in cruising mode, the altitude hold is
The Formal Requirements Elicitation Tool (FRET)

Captures requirements
Explains their meaning
The Formal Requirements Elicitation Tool (FRET)

Captures requirements
Explains their meaning
Produces formalizations

Future Time LTL

$$\text{\texttt{L}ast \ \forall \ (\texttt{cruising} \rightarrow (\texttt{altitude\_hold} \rightarrow \texttt{maintain\_altitude}))}$$

Target: \texttt{altitude\_hold\_autopilot} component.

Past Time LTL
The Formal Requirements Elicitation Tool (FRET)

Captures requirements
Explains their meaning
 Produces formalizations

Connects with analysis tools

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

Future Time LTL

\( \text{LAST} V (\text{cruising} \rightarrow (\text{altitude\_hold} \rightarrow \text{maintain\_altitude})) \)

Target: altitude\_hold\_autopilot component.

Past Time LTL

\( \text{contract FSMSpec(apfail:bool; sensorLimits:bool; standby:bool; supported:bool; ) returns ( pullup: bool; ) ; let } \)
\( \text{var autopilot:bool=supported and not apfail and not standby; } \)
\( \text{guarantee } "\text{FSM001}" \ H ((\text{sensorLimits and autopilot}) => (pullup)); \)
The Formal Requirements Elicitation Tool (FRET)

Captures requirements
Explains their meaning
Produces formalizations

Future Time LTL

\[ \text{LAST \ U (cruising } \rightarrow (\text{altitude\_hold } \rightarrow \text{maintain\_altitude})) \]

Target: altitude\_hold\_autopilot component.

Past Time LTL

Connects with analysis tools

```
contract FSMSpec(apfail:bool; sensorLimits:bool; standby:bool; supported:bool; ) returns (pullup: bool; )
let
var autopilot:bool=supported and not apfail and not standby;
guarantee "FSM001" { (sensorLimits and autopilot) => (pullup));
```
A set of industrial benchmarks. Each challenge includes:

- Natural language requirements
- A Simulink model
- A set of parameters (in .mat format) for simulating the model
A set of industrial benchmarks. Each challenge includes:

- Natural language requirements
- A Simulink model
- A set of parameters (in .mat format) for simulating the model

The challenges are:

- Representative of flight-critical systems
- Publicly available

https://github.com/hbourbouh/lm_challenges
[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.
Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

Step 0 involves understanding the requirement and making it precise. We start by identifying the variables involved.
Step 0: Requirement Elicitation

Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: The *roll hold reference* shall be set to 30 degrees in the same direction as the actual *roll angle* if the actual roll angle is greater than 30 degrees at the time of *roll hold mode engagement*.

Step 0 involves understanding the requirement and making it precise.

We start by identifying the *variables* involved.

Now we are ready to write the requirement in the *FRETish* language!
Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

FRETish fields:

Scope, condition, component*, shall*, timing, response*
Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

**FRETish** version:

If abs(roll_angle) >30 & roll_hold_mode_mode_engagement,

**FRETish** fields:

Scope, condition, component*, shall*, timing, response*
Requirement from the 6 DoF Dehavenland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

**FRETish** version:

If abs(roll_angle) > 30 & roll_hold_mode_engagement, Autopilot shall

**FRETish** fields:

Scope, condition, component*, shall*, timing, response*
Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

**FRETish** version:

If abs(roll_angle) >30 & roll_hold_mode_engagement, Autopilot shall immediately

**FRETish** fields:

Scope, condition, component*, shall*, timing, response*
Step 0: Requirement Elicitation

Requirement from the 6 DoF Dehavilland Beaver Autopilot LMCPS challenge

Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

**FRETish** version:

If abs(roll_angle) >30 & roll_hold_mode_engagement, Autopilot shall immediately satisfy roll_hold_reference = 30*sign(roll_angle)

**FRETish** fields:

Scope, condition, component*, shall*, timing, response*
Step 0: Requirement Elicitation

**Natural Language requirement:**

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

**FRETish version:**

[AP-003c-v1]: If abs(roll\_angle) >30 & roll\_hold\_mode\_engagement, Autopilot shall immediately satisfy roll\_hold\_reference = 30*sign(roll\_angle)

**FRETish fields:**
Scope, condition, component*, shall*, timing, response*
Natural Language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

FRETish version:

[AP-003c-v1]: If $\text{abs(roll\_angle)} > 30$ & roll\_hold\_mode\_engagement, Autopilot shall immediately satisfy roll\_hold\_reference = 30*sign(roll\_angle)

FRETish fields:
Scope, condition, component*, shall*, timing, response*
Step 0: Requirement Elicitation

Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

FRETish version:

[AP-003c-v1]: If abs(roll_angle) > 30 & roll_hold_mode_engagement, Autopilot shall immediately satisfy roll_hold_reference = 30*sign(roll_angle)

FRETish fields:
Scope, condition, component*, shall*, timing, response*
Step 0: Requirement Elicitation

Natural language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

FRETish version:

[AP-003c-v2]: **Autopilot shall immediately satisfy** if (abs(roll_angle) > 30 & roll_hold_mode_engagement) then (roll_hold_reference = 30*sign(roll_angle))

FRETish fields:

Scope, condition, component*, shall*, timing, response*
Natural Language requirement:

[AP-003c]: *The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.*

**FRETish** version:

[AP-003c-v2]: *Autopilot shall immediately satisfy if (abs(roll_angle) > 30 & roll_hold_mode_engagement) then (roll_hold_reference = 30*sign(roll_angle))*

FRETish fields:
Scope, condition, component*, shall*, timing, response*
Natural Language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

FRETish version:

[AP-003c-v2]: Autopilot shall immediately satisfy if \((\text{abs}(\text{roll}_\text{angle}) > 30 \ \& \ \text{roll}_\text{hold}_\text{mode}_\text{engagement})\) then \((\text{roll}_\text{hold}_\text{reference} = 30 \times \text{sign}(\text{roll}_\text{angle}))\)

FRETish fields:
Scope, condition, component*, shall*, timing, response*
Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

FRETish version:

[AP-003c-v3]: when in roll_hold mode Autopilot shall immediately satisfy if (abs(roll_angle) >30) then (roll_hold_reference = 30*sign(roll_angle))

FRETish fields:

Scope, condition, component*, shall*, timing, response*
Natural Language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

FRETish version:

[AP-003c-v3]: when in roll_hold mode Autopilot shall immediately satisfy if (abs(roll_angle) > 30) then (roll_hold_reference = 30*sign(roll_angle))
Step 1: Requirement Formalization

Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

FRETish version:

[AP-003c-v3]: when in roll_hold mode Autopilot shall immediately satisfy (abs(roll_angle) >30) => (roll_hold_reference = 30*sign(roll_angle))

Past Time Linear Temporal Logic formula:

\[ H(\text{roll\_hold} \& (\neg \text{FTP} \mid (Y(\neg \text{roll\_hold}))) \Rightarrow \text{abs}(\text{roll\_angle} > 30) \Rightarrow \text{roll\_hold\_reference} = 30 \ast \text{sign}(\text{roll\_angle})) \]

where FTP is a predicate that holds at the First Time Point of an execution
Step 2: Generation of Analysis Code

Natural Language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

Past Time Linear Temporal Logic formula:

\[ H(\text{roll\_hold} \& (! \text{FTP} \mid (Y (! \text{roll\_hold}))) => \text{abs}(\text{roll\_angle} > 30) => \text{roll\_hold\_reference} = 30 \times \text{sign}(\text{roll\_angle})) \]

CoCoSpec analysis code:

```plaintext
--AP-003c-v3 requirement in CoCoSpec
guarantee H((roll_hold and (FTP or (pre (not roll_hold))))
=> abs(roll_angle) > 30 =>
roll_hold_reference = 30 * sign(roll_angle))
```
Step 3: Automated Architectural Mapping

Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

FRETish version:

[AP-003c-v3]: when in roll_hold mode Autopilot shall immediately satisfy (abs(roll_angle) > 30) => (roll_hold_reference = 30*sign(roll_angle))
Step 3: Automated Architectural Mapping

Natural Language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

FRETish version:

[AP-003c-v3]: when in roll_hold mode Autopilot shall immediately satisfy (abs(roll_angle) > 30) => (roll_hold_reference = 30*sign(roll_angle))

Generation of traceability data!
Step 4: Generation of Simulink Monitors

Natural language requirement:

[AP-003c]: The roll hold reference shall be set to 30 degrees in the same direction as the actual roll angle if the actual roll angle is greater than 30 degrees at the time of roll hold mode engagement.

CoCoSpec analysis code:

```plaintext
-- AP-003c-v3 requirement in CoCoSpec
guarantee H((roll_hold and (FTP or (pre (not roll_hold))))
    => abs(roll_angle) > 30 =>
    roll_hold_reference = 30 * sign(roll_angle))
```

Simulink monitor:
Step 4: Generation of Simulink Monitors

Simulink monitors automatically attached on the model:
### RollAutopilot_PP/Roll_Autopilot

<table>
<thead>
<tr>
<th>Time</th>
<th>0</th>
<th>0.025</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phi (input)</td>
<td>6</td>
<td>31</td>
</tr>
<tr>
<td>Psi (input)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>p (input)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TAS (input)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>APEng (input)</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>HDGMode (input)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>HDGRef (input)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>TurnKnob (input)</td>
<td>3</td>
<td>31</td>
</tr>
<tr>
<td>PhiRef_cmd (output)</td>
<td>3</td>
<td>31</td>
</tr>
<tr>
<td>RollReference (local)</td>
<td>3</td>
<td>31</td>
</tr>
</tbody>
</table>
Step 6: Simulation

- roll_angle
- roll_hold_reference
- roll_hold
Modular Analysis

- For the Autopilot challenge this proved particularly useful:
  - We were able to analyze all properties that were specified at local level but none of the properties that were specified globally

### TABLE VI: AP Analysis Results with Kind2

<table>
<thead>
<tr>
<th>Reqs</th>
<th>Scope</th>
<th>Kind2 Result</th>
<th>Kind2 Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>[AP-000]</td>
<td>Global</td>
<td>Unsupported</td>
<td></td>
</tr>
<tr>
<td>[AP-001]</td>
<td>Roll AP</td>
<td>Valid</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>[AP-002]</td>
<td>Roll AP</td>
<td>Valid</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>[AP-003a]</td>
<td>Roll AP</td>
<td>Invalid</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>[AP-003b]</td>
<td>Roll AP</td>
<td>Invalid</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>[AP-003c]</td>
<td>Roll AP</td>
<td>Invalid</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>[AP-003d]</td>
<td>Roll AP</td>
<td>Valid</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>[AP-004]</td>
<td>Global</td>
<td>Unsupported</td>
<td></td>
</tr>
<tr>
<td>[AP-005]</td>
<td>Global</td>
<td>Unsupported</td>
<td></td>
</tr>
<tr>
<td>[AP-006]</td>
<td>Global</td>
<td>Unsupported</td>
<td></td>
</tr>
<tr>
<td>[AP-007]</td>
<td>Roll AP</td>
<td>Valid</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>[AP-008]</td>
<td>Roll AP</td>
<td>Valid</td>
<td>&lt; 1 sec</td>
</tr>
<tr>
<td>[AP-010]</td>
<td>Global</td>
<td>Unsupported</td>
<td></td>
</tr>
</tbody>
</table>

Total running time CoCoSim: 40.589s
To summarize

- Using an end-to-end framework significantly simplifies requirements elicitation and model analysis.
- Eliciting requirements with unambiguous and as-intended semantics is not an easy task.
  - Explanations and interactive exploration of requirements help.
- Industrial requirements are complex to analyze.
  - It is important to provide modular analysis.

Our open source tools can be accessed on Github:

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

Thank you!