MANCHESTER 1824
The University of Manchester

Royal Academy of Engineering

Strong Software Reliability for Autonomous Space Robotics

Marie Farrell

March 31, 2023

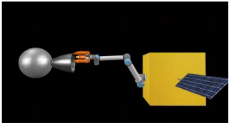
Background: Me

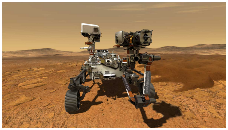
- PhD at Maynooth University: 2013–2017
- Postdoc at Universities of Liverpool and Manchester: 2018–2020 (FAIR-SPACE Hub)
- Postdoc at Maynooth University: 2020–2022 (VALU3S Project)
- Royal Academy of Engineering Research Fellow at The University of Manchester: 2022–2027

IRISH RESEARCH COUNCIL An Chomhairle um Thaighde in Éirinn   FAIR-SPACE   VALU3S   Royal Academy of Engineering

Moon Village

Strong Software Reliability for Autonomous Space Robotics

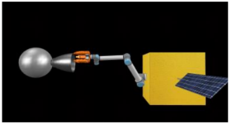


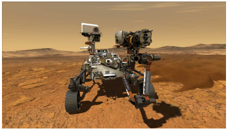


Problem:

- Space exploration necessitates the use of **autonomous** robotic systems.
- Current **verification** approaches are **not sufficient** to accurately specify the required autonomy.
- **Autonomy** introduces a unique set of concerns related to **verification** and **assurance**.

Strong Software Reliability for Autonomous Space Robotics

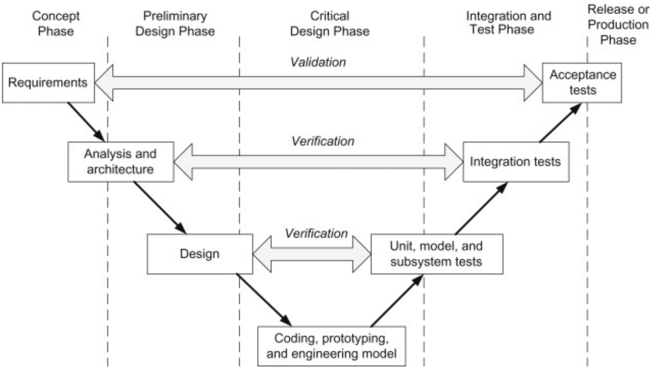


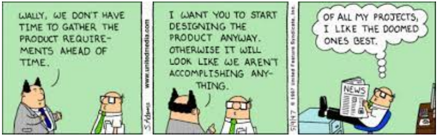


Aim:

- Devise new ways of **describing**, **analysing** and **assuring** the correct autonomous behaviour of robotic space systems.

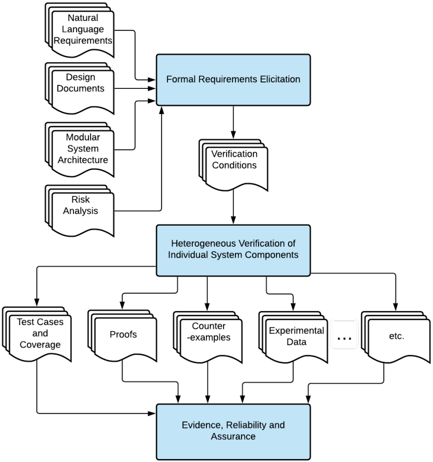
Software Development: V-Model





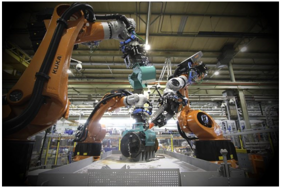
Integrating Multiple Heterogeneous Verification Techniques

- True autonomy requires a step change in **understanding** and **verification**.
- Issue of autonomous systems **assurance** remains unsolved.
- Adopt a sophisticated and **complementary** combination of robust V&V methods to support deployment in space.
- Provide formalisation of requirements for **Machine Learning** components.



Impact

This will ultimately lead to more **reliable**, more **usable** and more **effective** autonomous robots being deployed more **confidently** across a **range of sectors**.



But...

What Should I Verify?

Requirements Engineering

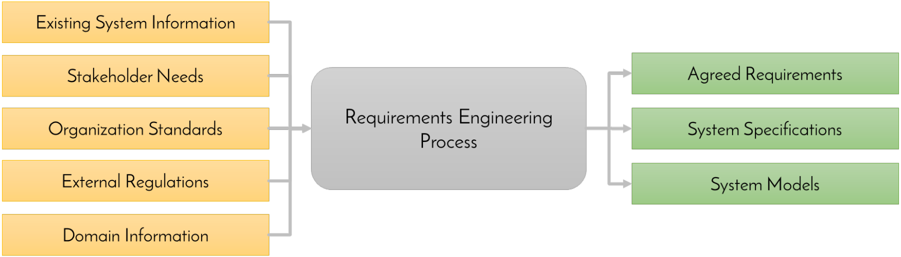
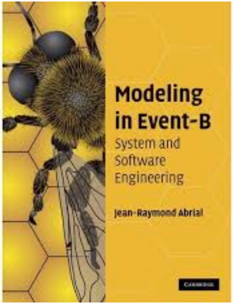


Illustration: The overall requirements engineering process

Moon Village

Requirement: Robots should always maintain a safe distance from astronauts.



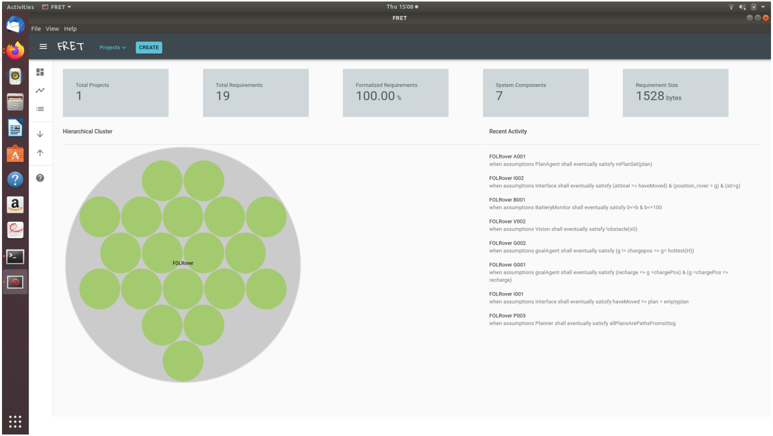
What Should I Verify?

Natural Language Requirements $\neq$ Formal Properties

Natural-Language Requirements: Aircraft Engine Software Controller

During regulation of nominal system operation (no change in pilot input), controller operating mode shall appropriately switch between nominal and surge/stall prevention operating state.

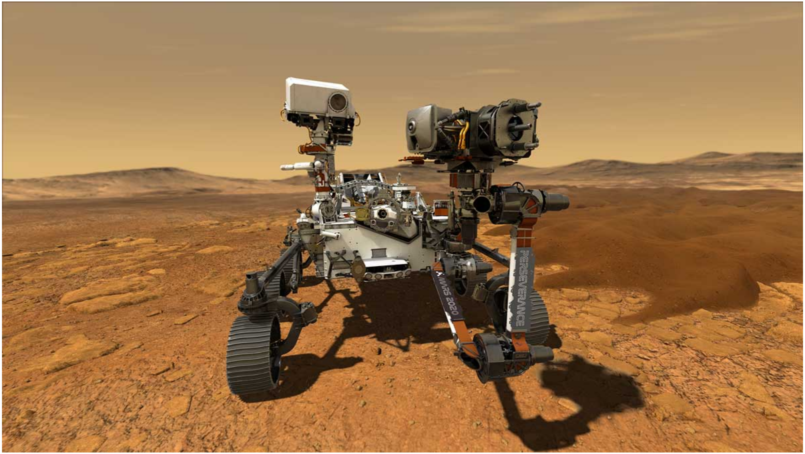
Formal Requirements Elicitation Tool (FRET)



Formal Requirements Elicitation Tool (FRET)

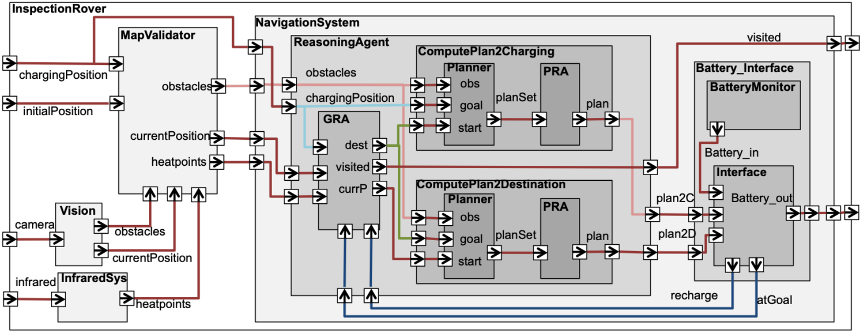
- supports the formalisation, understanding and analysis of requirements user-friendly interface
- intuitive diagrammatic explanations of requirement semantics
- users specify requirements in restricted natural language, called FRETISH, which embodies a temporal logic semantics

Example: Rover Navigation



Bourbouh, H., Farrell, M., Mavridou, A., Sljivo, I., Brat, G., Dennis, L. A., & Fisher, M. *Integrating Formal Verification and Assurance: An Inspection Rover Case Study*. NFM 2021.

Rover Architecture



R1: The rover shall not run out of battery.
R2: The rover shall not collide with an obstacle.
R3: The rover shall visit all reachable points of interest.

Rover Requirement in FRET

Update Requirement

Status ✓

Requirement ID: R1.2

Parent Requirement ID: R1

Project:

Rationale and Comments

Rationale

Charging station shall be selected as the next destination whenever the recharge flag is set to true

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.

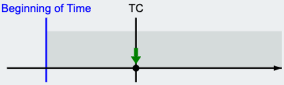
SCOPE CONDITIONS COMPONENT* SHALL* TIMING RESPONSES*

if recharge GRA shall immediately satisfy goal=chargePosition

SEMANTICS

ASSISTANT TEMPLATES

ENFORCED: in the interval defined by the entire execution.
TRIGGER: first point in the interval if ***(recharge)*** is true and any point in the interval where ***(recharge)*** becomes true (from false).
REQUIRES: for every trigger, if trigger holds then RES also holds at the same time point.



TC = ***(recharge)***, Response = ***(goal = chargePosition)***.

Diagram Semantics

Formalizations

Future Time LTL

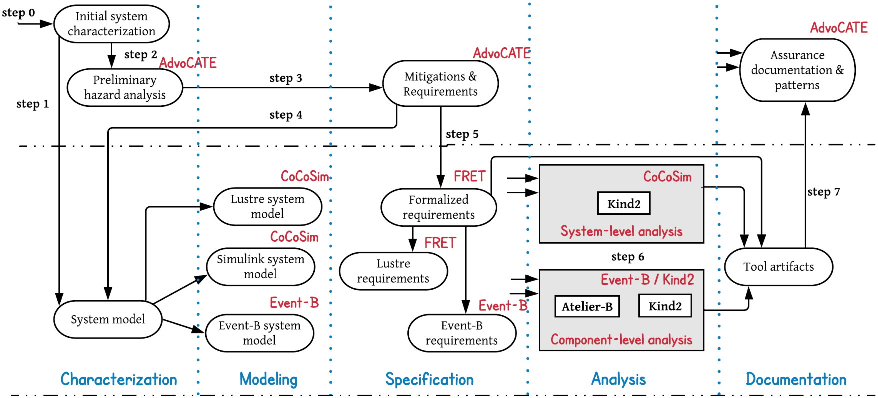
Past Time LTL

```
(H (((recharge) & ((Y (! (recharge))) | FTP))
-> (goal = chargePosition)))
```

Target: ***GRA*** component.

SIMULATE

Verifying Rover Navigation



Example: Aircraft Engine Controller

ID	FRETISH
UC5_R_1	if ((sensorfaults) & (trackingPilotCommands)) Controller shall satisfy (controlObjectives)
UC5_R_2	if ((sensorfaults) & (!trackingPilotCommands)) Controller shall satisfy (controlObjectives)
UC5_R_3	if ((sensorfaults) & (trackingPilotCommands)) Controller shall satisfy (operatingLimitObjectives)
UC5_R_4	if ((sensorfaults) & (!trackingPilotCommands)) Controller shall satisfy (operatingLimitObjectives)
UC5_R_5	if ((mechanicalFatigue) & (trackingPilotCommands)) Controller shall satisfy (controlObjectives)
UC5_R_6	if ((mechanicalFatigue) & (!trackingPilotCommands)) Controller shall satisfy (controlObjectives)
UC5_R_7	if ((mechanicalFatigue) & (trackingPilotCommands)) Controller shall satisfy (operatingLimitObjectives)
UC5_R_8	if ((mechanicalFatigue) & (!trackingPilotCommands)) Controller shall satisfy (operatingLimitObjectives)
UC5_R_9	if ((lowProbabilityHazardousEvents) & (trackingPilotCommands)) Controller shall satisfy (controlObjectives)
UC5_R_10	if ((lowProbabilityHazardousEvents) & (!trackingPilotCommands)) Controller shall satisfy (controlObjectives)
UC5_R_11	if ((lowProbabilityHazardousEvents) & (trackingPilotCommands)) Controller shall satisfy (operatingLimitObjectives)
UC5_R_12	if ((lowProbabilityHazardousEvents) & (!trackingPilotCommands)) Controller shall satisfy (operatingLimitObjectives)
UC5_R_13	if (trackingPilotCommands) Controller shall satisfy (changeMode(nominal)) | (changeMode(surgeStallPrevention))
UC5_R_14	if (!trackingPilotCommands) Controller shall satisfy (changeMode(nominal)) | (changeMode(surgeStallPrevention))

Farrell, M., Luckcuck, M., Sheridan, O., & Monahan, R. *FRETting about requirements: formalised requirements for an aircraft engine controller.* In REFSQ 2022.

Formalised Requirements for an Aircraft Engine Controller

ID	Parent	FRETISH
UC5_R_1.1	UC5_R_1	when (diff(r(i),y(i)) > E) if((sensorValue(S) > nominalValue + R) | (sensorValue(S) < nominalValue - R) | (sensorValue(S) = null) & (pilotInput => setThrust = V2) & (observedThrust = V1)) Controller shall until (diff(r(i),y(i)) < e) satisfy (settlingTime >= 0) & (settlingTime <= settlingTimeMax) & (observedThrust = V2)
UC5_R_1.2	UC5_R_1	when (diff(r(i),y(i)) > E) if((sensorValue(S) > nominalValue + R) | (sensorValue(S) < nominalValue - R) | (sensorValue(S) = null)& (pilotInput => setThrust = V2) & (observedThrust = V1)) Controller shall until (diff(r(i),y(i)) < e) satisfy (overshoot >= 0) & (overshoot <= overshootMax) & (observedThrust = V2)
UC5_R_1.3	UC5_R_1	when (diff(r(i),y(i)) > E) if((sensorValue(S) > nominalValue + R) | (sensorValue(S) < nominalValue - R) | (sensorValue(S) = null)& (pilotInput => setThrust = V2)& (observedThrust = V1)) Controller shall until (diff(r(i),y(i)) < e) satisfy (steadyStateError >= 0) & (steadyStateError <= steadyStateErrorMax) & (observedThrust = V2)

- 14 natural-language requirements became 42 formalised requirements

Formalised Requirements for an Aircraft Engine Controller

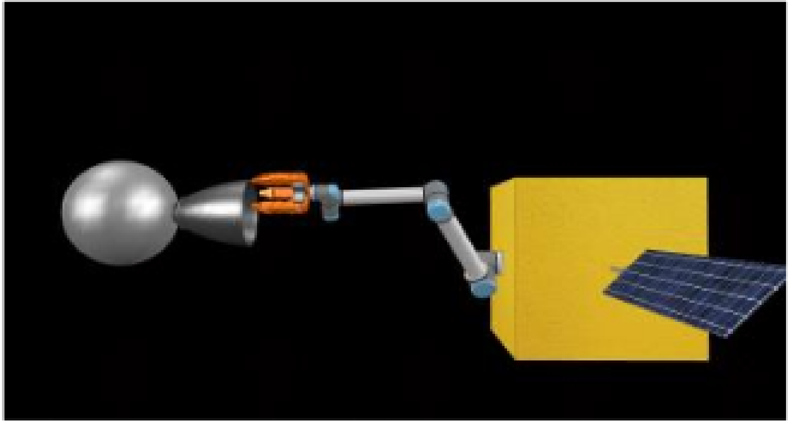
<table>
<tr><th>ID</th><th>Parent</th><th>FRETISH</th></tr>
<tr><td>UC5_R_1.1</td><td>UC5_R_1</td><td>when (diff(r(i),y(i)) > E) if((sensorValue(S) > nominalValue + R) | (sensorValue(S) < nominalValue - R) | (sensorValue(S) = null) & (pilotInput => setThrust = V2) & (observedThrust = V1)) Controller shall until (diff(r(i),y(i)) < e) satisfy (settlingTime >= 0) & (settlingTime <= settlingTimeMax) & (observedThrust = V2)</td></tr>
<tr><td>UC5_R_1.2</td><td>UC5_R_1</td><td>when (diff(r(i),y(i)) > E) if((sensorValue(S) > nominalValue + R) | (sensorValue(S) < nominalValue - R) | (sensorValue(S) = null)& (pilotInput => setThrust = V2) & (observedThrust = V1)) Controller shall until (diff(r(i),y(i)) < e) satisfy (overshoot >= 0) & (overshoot <= overshootMax) & (observedThrust = V2)</td></tr>
<tr><td>UC5_R_1.3</td><td>UC5_R_1</td><td>when (diff(r(i),y(i)) > E) if((sensorValue(S) > nominalValue + R) | (sensorValue(S) < nominalValue - R) | (sensorValue(S) = null)& (pilotInput => setThrust = V2)& (observedThrust = V1)) Controller shall until (diff(r(i),y(i)) < e) satisfy (steadyStateError >= 0) & (steadyStateError <= steadyStateErrorMax) & (observedThrust = V2)</td></tr>
</table>

- 14 natural-language requirements became 42 formalised requirements

What do we mean?

'***it forces you to think about the actual meaning behind the natural-language requirements***'. - *Collins Aerospace/UTRC Ireland*

Example: Active Debris Removal



Autonomous Grasping for Active Debris Removal

ID	**FRET Formalisation**
R1	SV shall satisfy (grasp(TGT, BGP) & closer(SV, TGT))
R1.1	Camera shall satisfy distance(Camera, TGT) $\geq$ 0.5
R1.2	TGT shall satisfy if !contact(SVA, TGT) then motionless(TGT)
R1.3	Camera shall satisfy valid(p)
R1.3.1	Camera shall satisfy maxRes(p) = 1280*720
R1.3.2	Camera shall satisfy length(p) $>$ 0
R1.4	Imagepreprocessing shall satisfy length(filteredimage) $\leq$ length(p) & length(filteredimage) $>$ 0
R1.5	Findoptimalgrasp shall satisfy if exists(BGP) then return(BGP)
R.1.5.1	Findoptimalgrasp shall satisfy offset(BGP, TGT) = 1 & -20 $\leq$ fingersurfaceyaw & fingersurfaceyaw $\leq$ 20
R1.5.2	findoptimallgrasp shall satisfy length(grasps) $\geq$ 0
R1.6	Findoptimalgrasp shall satisfy if !(exists(BGP)) then printerror
R1.7	Controller shall satisfy executeJointTrajectory(SVA, BGP)
R1.8	SVA shall satisfy captured(TGT) $\Rightarrow$ contactpoint(SVA, TGT) = BGP
R1.9	SV shall satisfy totalpullingdistance $\geq$ 0.3 & totalpullingdistance $\leq$ 0.5
R2	SV shall always satisfy !collide(SV, TGT)
R2.1	SV shall always satisfy !(position(SV) = position(TGT))
R2.2	SV shall always satisfy contactpoint(SVG, TGT) = BGP.
R2.2.1	SV shall satisfy if !grasped then contactpoint(SV, TGT) = null
R2.2.2	SV shall satisfy if grasped then contactpoint(SVG,TGT) = BGP + errormargin
R2.3	SVG shall satisfy captured(TGT) $\Rightarrow$ force = 180

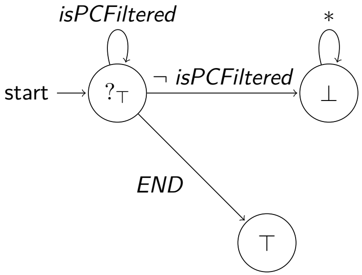
Farrell, M., Mavrakis, N., Ferrando, A., Dixon, C., & Gao, Y. *Formal Modelling and Runtime Verification of Autonomous Grasping for Active Debris Removal.* Frontiers in Robotics and AI. 2022.

What Can We Verify?

Autonomous Grasping for Active Debris Removal

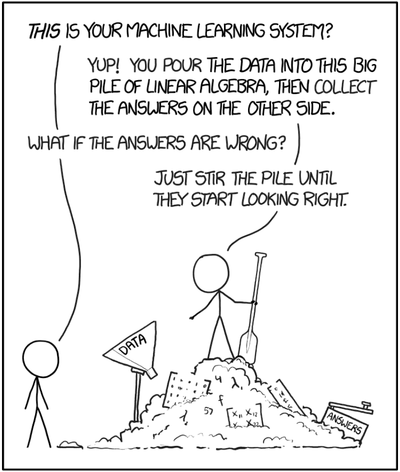
```
method imagepreprocessing(t: real , p: array<Point>, v: real , nb: int , rf: real)
  returns (filteredimage: array<Point>)
requires 0 < p.Length; // R1.3.2
requires v > 0.0;
ensures filteredimage.Length ⩽ p.Length;//R1.4
ensures filteredimage.Length > 0 ; // R1.4
{
 filteredimage := removeDepth(p,t); //remove distant points from p.
 filteredimage := downSample(filteredimage,v); / voxel representation of p.
 filteredimage := filter(filteredimage,nb,rf); //removes noise and speckles.
}
```

R1.4:



Autonomous Systems: ML and Explainability Requirements

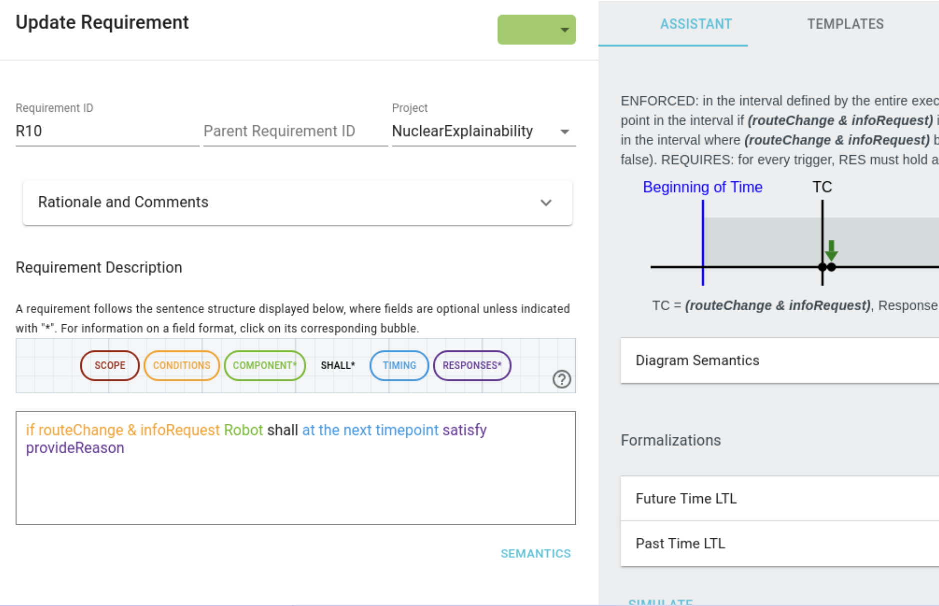
- Very difficult to formalise correct functionality of ML and explanability.
- Very difficult to verify these systems.



Explainabilty Requirements in FRET: Nuclear Robotics

ID ↑		Summary
R1	⊕	Robot shall before go satisfy explain
R10	⊕	if routeChange & infoRequest Robot shall at the next timepoint satisfy provideReason
R11	⊕	if planDeviation Robot shall immediately satisfy returnCurrentLocation
R12	⊕	Robot shall always satisfy computeShortestPath(A,B)
R13	⊕	Robot shall always satisfy (explainCurrentPath => returnCurrentPath) & (explainIntendedPath => returnIntendedPath) & followPath & noDeviations
R14	⊕	Robot shall always satisfy (explainVisited => returnVisited) & (explainMeasurements => returnMeasurementOrigins)
R15	⊕	if notOptimal Robot shall immediately satisfy explain
R16	⊕	if pathImpossible Robot shall immediately satisfy returnAllOptions
R17	⊕	Robot shall always satisfy provideHealthUpdate & selfRecoveryStrategyConfidence > acceptableThreshold
R18A	⊕	Robot shall always satisfy avoidObstacles & minimiseRadRisk

Explainabilty Requirements in FRET: Nuclear Robotics



Requirements for Machine Learning

Req ID	Requirement
[RRAV-001]	The neural network shall output the cross track distance error (perpendicular distance from the rover to the centerline.) Error to truth must not exceed X.
[RRAV-002]	Neural network shall output cross track heading error (the angle between the rover heading and the centerline.) Error to truth must not exceed X.
[RRAV-003]	Upon receiving an image, the Neural Network shall output the distance and the angle within X seconds (latency).
[RRAV-004]	Neural network shall output a sensible distance: the value must be between 0 and half the width of the taxiway plus X (buffer X so that it can still report if it is off the taxiway).
[RRAV-005]	Neural network shall output a sensible angle: the value must be between -90 and 90 degrees.
[RRAV-006]	The neural network shall achieve a minimum of X% accuracy on training and Y% accuracy on testing.
[RRAV-007]	(Local robustness) The neural network shall be robust to small perturbations in the image (pixels).
[RRAV-008]	(Semantic variations) The neural network shall be robust to irrelevant variations in the scene.
[RRAV-009]	The neural network shall safely navigate intersections.
[RRAV-010]	The magnitude of the cross track distance error shall drop below X m within T seconds and remain there.
[RRAV-011]	The magnitude of the cross track heading error shall drop below X degrees within T seconds and remain there.

Requirements for Machine Learning

Req ID	Requirement Pattern (source: NASA)
[IC-001]	The sw shall achieve an average PARAMETER value of X.
[IC-002]	The sw shall estimate PARAMETER to within $+-X$ with a $Y\%$ confidence.
[IC-003]	The sw shall estimate the confidence of the PARAMETER estimate.
[IC-004]	The requirement shall be verified by measuring the average of the parameter over N repetitions.
[IC-005]	The sw shall estimate PARAMETER with an $X\%$ confidence interval of no more than $+-Y$.
[IC-006]	The sw shall calculate the PARAMETER confidence interval at an $X\%$ confidence level.
[IC-007]	The sw shall calculate the PARAMETER as a probability distribution.
[IC-008]	The sw shall determine PARAMETER with a high level of confidence.
[IC-009]	The sw shall detect $X\%$ of occurrences of EVENT.
[IC-010]	The risk-ratio requirements shall be verified using a statistically significant set of SCENARIOS.
[IC-011]	The sw shall cause EVENT at a rate less than X times per Y DURATION.
[IC-012]	The sw shall detect CONDITION that implies EVENT is probable.
[IC-013]	The sw shall take action so that the risk ratio thresholds are satisfied.

Patterns derived from 770 mission/industrial requirements.

Farrell, M., Mavridou, A. & Schumann, J. *Exploring Requirements for Software that Learns: A Research Preview.* REFSQ 2023.

Are ML Requirements Special?

- **Confidence, Criticality and Risk Levels**:
  "The sw shall determine PARAMETER with a high level of confidence."
- **Accuracy as a measure of functional correctness**:
  "The neural network shall achieve a minimum of X% accuracy on training and Y% accuracy on testing."
- **Achievement of average value**:
  "The requirement shall be verified by measuring the average of the parameter over N repetitions."
- **Robustness**:
  "The neural network shall be robust to small perturbations in the image (pixels)."

Are ML Requirements Special?

Yes!

- Probability plays a distinguishing role.

Uncertainty in ML Requirements

- Probabilities **within** requirements:
  "The sw shall detect CONDITION that implies EVENT is probable."

- Probabilities **about** requirements:
  "The sw shall estimate PARAMETER to within $+-X$ with a $Y\%$ confidence."

Summary

- Two important questions:
  - What Should I Verify?
  - What Can I verify?

- Tools like FRET can help.

- Requirements engineering for autonomous systems is very difficult.
  - Tools need to be able to capture requirements related to uncertainty.

- Heterogeneous/corroborative verification is the way forward.

Questions?

marie.farrell@manchester.ac.uk