Towards A Catalogue of Requirement Patterns for Space Robotic Missions

Mahdi Etumi Hazel M. Taylor Marie Farrell
University of Manchester
Manchester, UK
firstname.surname@manchester.ac.uk

In the development of safety and mission-critical systems, including autonomous space robotic missions, complex behaviour is captured during the requirements elicitation phase. Requirements are typically expressed using natural language which is ambiguous and not amenable to formal verification methods that can provide robust guarantees of system behaviour. To support the definition of formal requirements, specification patterns provide reusable, logic-based templates. A suite of robotic specification patterns, along with their formalisation in NASA's Formal Requirements Elicitation Tool (FRET) already exists. These pre-existing requirement patterns are domain agnostic and, in this paper we explore their applicability for space missions. To achieve this we carried out a literature review of existing space missions and formalised their requirements using FRET, contributing a corpus of space mission requirements. We categorised these requirements using pre-existing specification patterns which demonstrated their applicability in space missions. However, not all of the requirements that we formalised corresponded to an existing pattern so we have contributed 5 new requirement specification patterns as well as several variants of the existing and new patterns. We also conducted an expert evaluation of the new patterns, highlighting their benefits and limitations.

1 Introduction

Autonomous space robotics are used and planned to be used in a wide variety of missions. Examples include the ISS-based Astrobee ([5]), the roaming Mars rovers ([31]), and isolated satellites ([27], [35]). These missions help to further our understanding of the universe as well as enable Critical National Infrastructure to function correctly (e.g. communications, navigation systems, etc.). As a result, space robotic missions are viewed as both safety and mission-critical endeavours which require high degrees of assurance prior to and during deployment. Such complex critical systems typically necessitate a robust development process, often involving formal methods, that begins with a requirements engineering phase where developers elicit and formalise properties that capture correct system behaviour [9].

Alongside requirements engineering, system specifications are constructed to enable the analysis of system architectures and behaviour prior to deployment. Requirements are typically expressed as natural language descriptions but formal specification frameworks use formal logics (rather than natural language) to specify and subsequently verify system behaviour. Various tools exist to help bridge the gap between natural language and logical requirements. One notable example is NASA's Formal Requirements Elicitation Tool (FRET) [13], which provides a structured natural language for requirement definition. From this structured natural language, called FRETISH, FRET automatically generates a formal, temporal logic semantics for each requirement that can be used by formal methods.

It has long been recognised that specification is a huge bottleneck in critical systems development, especially when formal methods are used for reliable verification in the development of autonomous systems [29]. Deriving usable specification patterns can provide developers with guidance during these initial project phases and a corpus of specification patterns for robotic missions already exists in [24] which

Matt Luckcuck, Maike Schwammberger & Mengwei Xu (Eds.):
Seventh International Workshop on Formal Methods
for Autonomous Systems (FMAS 2025)
EPTCS 436, 2025, pp. 136–166, doi:10.4204/EPTCS.436.12

was further extended by [34]. These specification patterns are expressed in Linear Temporal Logic (LTL) [24] and formalised using FRET [34]. These existing catalogues of specification patterns are domain-agnostic in nature, which is undoubtedly a strength of the approach. However, it remains to be seen how these patterns can be used in a domain-specific context. In this paper, we recount our explorations about how these pre-existing patterns can be used in space systems. To that end, we contribute:

1. A corpus of requirements for space missions obtained through a detailed literature review.
2. An expanded catalogue of space domain-specific specification patterns that extends [24, 34], including both the LTL and FRET formalisations for each of the newly defined patterns.
3. An expert evaluation where we interviewed several experts with experience in the space domain to provide additional insights into our new patterns and potential future extensions.

The remainder of this paper is structured as follows. We begin by providing an overview of the essential background material (Section 2). Then, Section 3 describes the methodology that we used for our literature review for this study. This is followed by a detailed analysis of our findings in Section 4. Here, we discuss the requirements that we found, novel patterns, metrics and an expert evaluation. We provide a detailed discussion in Section 5 and Section 6 concludes while outlining future research directions.

2 Background

Specification patterns have been proposed in the literature as a way to bridge the gap between natural language descriptions and formal, logical patterns [8]. The ultimate goal is to build a set of useful and usable patterns that can guide engineers during the requirements elicitation and formalisation steps. Specification patterns have been introduced for safety [8], probabilistic [15] and real-time specifications [20]. A comprehensive overview of various patterns has been conducted in [2]. These specification patterns have been studied in various areas, including web services [4] and robotics [24, 34, 23].

All of these patterns provide template requirement specifications that can be used to formally specify suitable properties. In fact, they can be used as a guide for engineers who are not familiar with formal specification to elicit and formalise formally verifiable requirements. These sets of requirement specification patterns generally define generic, domain-agnostic requirement specifications. In this paper, we explore the use of existing (domain agnostic) specification patterns in space applications and identify new patterns from the space domain.

The core related work for this paper is the existing pattern catalogue for robotics in [24, 34]. In these papers, Linear Time Temporal Logic (LTL) [11] is the logic of choice for robotic specification patterns. This is a sensible choice since multiple surveys on formal specification and verification for autonomous robots demonstrated that LTL is the predominant logic used in specifying requirements for robotic systems [21, 3]. As such, we also specify our patterns in LTL. LTL formulas are made up of atomic propositions, logical connectives and temporal operators. We provide the syntax of LTL:

$$\phi := true \mid a \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid X\phi \mid F\phi \mid G\phi \mid \phi_1 \mathscr{U} \phi_2$$

where a denotes an atomic proposition, $\wedge$ denotes logical conjunction and $\neg$ denotes logical negation. Then $X\phi$ specifies that ϕ holds in the *next* state, $F\phi$ that ϕ holds *eventually* in some future state and $G\phi$ that ϕ holds *globally* in all future states. The *until* operator, $\phi_1 \mathscr{U} \phi_2$, specifies that ϕ_1 holds true until a future state where ϕ_2 becomes true, and ϕ_2 does indeed eventually become true. This is commonly known as *strong* until in the literature. FRET (described below) uses the *weak* version of until which omits the condition that ϕ_2 must eventually become true. We use $\phi_1 \mathscr{W} \phi_2$ to denote weak until.

Requirements are typically specified using natural language, which is prone to ambiguity. For formal verification, formal logics are used to specify requirements. Various tools and approaches, including NASA's Formal Requirements Elicitation Tool (FRET) [13], bridge the gap between natural language and formalised, logical requirements. FRET uses a structured natural language, called FRETISH, to give a clear syntax that users specify their requirements in. For each FRETISH requirement, FRET generates a corresponding temporal logic semantics [14]. The FRETISH fields are `scope`; `condition`; `component*`; `shall*`; `probability`; `timing`; and `response*`. Fields marked with an asterisk (*) are mandatory.

The `scope` field specifies the *mode of operation* that is relevant to the component's behaviour. The `condition` field specifies the *conditions* under which the requirement should hold. The `component` field identifies the *component* that the requirement applies to. The `probability` field specifies the *probability* of the timed response, while the `timing` field specifies when the boolean `response` should hold. For each field, there are various options for the keywords that can be used; these are designed to be intuitive to the user. For example, we can write requirements like:

`in FailSafe mode if errorObserved Robot` shall `with probability` $\geq$ `0.99 immediately` satisfy `navigateToSafeArea`

For any FRETISH requirement, both LTL and Probabilistic Computation Tree Logic (PCTL*) semantics are provided by FRET. In this paper, we use the LTL semantics since probabilistic requirements did not appear very often in our dataset but we discuss probabilistic requirements briefly in Section 4.4. To the best of our knowledge, although general guidance, standards and space agency processes exist that are related to requirement specification for space missions [35, 16, 10, 28], we are not aware of any related work that aims to derive formal specification patterns for autonomous robotic space missions.

3 Methodology

In this section, we outline the methodology that we followed when assembling a corpus of space requirements to answer the research questions that we define below:

RQ1: Are existing specification patterns relevant and sufficient for autonomous space robotic missions?

RQ2: Can specification patterns for autonomous space robotic missions be expressed using FRET?

To answer these research questions, we assembled a corpus of 116 requirements for (autonomous) space robotic missions. In total, we reviewed 30 different literature sources to identify and formalise requirements for autonomous space robotics. These sources were found using Google Scholar[1] and the NASA Technical Reports Server[2]. Of these sources, 14 were used to derive requirements. A wide range of topics in autonomous space robotics were encountered, including docking, manoeuvring, robotic systems, satellites, and more. These topics were explored by including the search terms "Requirements OR Verification OR Specification". Snowballing was used wherever possible. This resulted in the identification of 116 requirements.

We provide details about where the in-scope papers were published in Table 1. There is a diverse set of conference, journal, book and technical report publications in this dataset. There is a mix of venue types ranging from more application-oriented venues to formal methods and requirement engineering conferences. There was some duplication as some papers were extended in more detail as NASA technical reports. However, if the same requirement appeared in multiple sources it was only counted once.

[1] `https://scholar.google.com/`
[2] `https://ntrs.nasa.gov/`

Venues	Type	Ref
Acta Astronautica	Journal	[35]
Autonomy Requirements Engineering for Space Missions	Book	[33]
ACM Transactions on Software Engineering and Methodology	Journal	[7]
Revista Tecnología en Marcha	Journal	[27]
Space Operations Conference	Conference	[5]
Technical Report from NASA	Technical Report	[6, 19, 1, 26]
Computer Aided Verification	Conference	[18]
Requirements Engineering: Foundation for Software Quality	Conference	[25]
IEEE Robotics and Automation Magazine	Journal	[17]
Progress in Aerospace Sciences	Journal	[12]
Space Science Reviews	Journal	[31]

Table 1: Publication venues for the in-scope papers that we found.

For each requirement that we found in the literature, we recorded: a natural language description, the component/system name and the literature source. We also determined whether they fit into a pre-existing pattern formulation from [24, 34]. When they did not fit into a pre-existing template, we decided, based on the data, whether a new template should be defined. For all new templates, we provided an LTL and FRETISH formalisation. For all existing templates, we instantiated the pattern using the FRETISH templates from [34].

Several examples of the natural language descriptions that we encountered are shown in Table 2. Many of these were ambiguous and required additional context from the surveyed papers to truly understand. For example, timing was often omitted in the natural language, so it was mostly inferred based on our understanding of the associated mission described in the relevant paper. The full set of the 116 formalised requirements is in the Appendix and available in our repository[3].

4 Requirement Specification Patterns

In this section, we summarise the space missions that we encountered through our survey, define several new patterns using LTL and FRET, explore the distribution of various patterns (both new and old) throughout our dataset and describe the results of an expert evaluation study that we carried out.

4.1 Description of Space Missions Surveyed

We describe several of the robotic space missions that were found during our review. Figure 1 illustrates aspects of these systems. We include an example requirement from each of these missions in Table 2.

Astrobee: This is a free-flying robot system that is designed to assist astronauts on the International Space Station (ISS). The system comprises free-flying robots, a docking station that provides an Ethernet connection and recharging capabilities and a ground control segment. Astrobee can operate autonomously or under remote control. Astrobee aims to reduce crew workload by performing routine monitoring tasks, handling contingencies and improving productivity onboard the ISS [5]. Requirement #60-B in Table 2 for Astrobee describes a requirement related to the crew interface for Astrobee.

[3] https://github.com/mariefarrell/spacepatterns

ID	Natural Language Description	FRETish	Pattern	Source
5	Deploy the parachute using navigated position information once safe parachute deployment velocities have been reached.	if parachutedistance <= safeparachutedistance EDL shall immediately satisfy DeployParachute	TRIGGERED INSTANT REACTION	[31]
9-B	The IM shall be isolated in case of contingency.	In SAFEMODE Whenever CONTINGENCY IM shall immediately satisfy isolated	MODAL INSTANT REACTION	[35]
16	Autonomously release MMO when the polar orbit is reached.	Upon polar_orbit_reached BepiColombo shall immediately satisfy release_MMO	TRIGGERED INSTANT REACTION	[33]
27	If it is required to know the state of the spacecraft, even during the section of the orbit without a communication link with ground segment, store telemetry data.	If StateRequired & !CommunitcationLink CubeSat shall immediately satisfy StoreData	TRIGGERED INSTANT REACTION	[27]
46	The SPHERES satellites, however, triangulate their position using infrared/ultrasonic beacons, preventing them from navigating outside the two-meter cube defined by the fixed beacon locations.	Whenever moving SPHERES shall immediately satisfy x<2 & y<2 & z<2	STAY-IN-PERIMETER	[17]
60-B	A blue "Aud" light tells the crew that the microphone is on.	In AudioRecording Astrobee shall always satisfy BlueAudLED	MODAL MAINTAIN SAFE SPACE	[5]
77-B	Furthermore, when the Int-Ball2 automatically detects that the remaining battery power is low, it returns to the DS for recharging.	Whenever IntBall2Power <= SafeBattery IntBall2 shall at the next timepoint satisfy RechargeMode	PROMPT REACTION	[17]

Table 2: Examples of requirements from the space missions that are described in Section 4.1.

Int-Ball2: Like Astrobee, Int-Ball2 is a free-flying robotic system that is designed to assist astronauts on the ISS. It was developed to overcome the shortcomings of the first Int-Ball with enhanced propulsion, markerless navigation and a docking station for autonomous charging. Int-Ball aims to assist astronauts by capturing photos and videos, reducing the need for crew-operated cameras [17]. Requirement #77-B in Table 2 describes a requirement related to when Int-Ball2 should autonomously recharge its battery.

Mars 2020: This mission saw the deployment of a rover named '*Perseverance*' and a helicopter called '*Ingenuity*' to a chosen Martian crater called Jezero. The mission had 2 goals: (1) finding evidence of microbial life and (2) collecting samples for analysis on Earth. The design of Perseverance builds on the successful design of the *Curiosity* rover with new science instruments for improved analysis and collection. Perseverance also tests new technologies to support future exploration of Mars [31]. Requirement #5 in Table 2 is specific to the Entry, Descent and Landing (EDL) subsystem of the Perseverance rover and specifies when the parachute should be deployed.

Inflatable Module: This is a case study that presents an in-orbit validation mission of an Inflatable Module (IM). The mission involves launching a spacecraft that docks with the ISS, delivering

supplies and testing the IM before disposal through a controlled destructive re-entry. The system operates in five modes across two variants and ten mission phases [35]. Requirement #9-B in Table 2 specifies how the IM should be isolated while operating under specific conditions in safe mode.

BepiColombo: This is a mission to study and research Mercury. The mission is carried out by two satellites: Mercury Planetary Orbiter (MPO) and Mercury Magnetospheric Orbiter (MMO). These two satellites are joined together by the BepiColombo transfer module. The BepiColombo module will provide propulsion, power, and communications while travelling to Mercury. Upon reaching Mercury, the transfer module will release the MPO and the MMO, allowing their independent observation of Mercury's surface, exosphere and composition [33]. Requirement #16 in Table 2 describes a requirement on the MMO release operation of BepiColombo.

CubeSat: CubeSat satellites are a class of miniaturised satellites with the dimensions 10cm x 10cm x 10cm or 1U. These small satellites can be configured with one another to create larger CubeSats; for example, a 2U CubeSat would be 10cm x 10cm x 20cm, and a 3U CubeSat would be 10cm x 10cm x 30cm. They provide an accessible and low-cost platform for research and monitoring in space, making them widely used by universities, startups, and space agencies [27]. In Table 2, requirement #27 describes a communication specific behaviour of the CubeSat mission.

SPHERES: SPHERES (Synchronized Position Hold Engage and Reorient Experimental Satellites) is a free-flying system deployed in the ISS. Mainly used as a zero-g research platform, SPHERES has been used in the validation of metrology, formation flight, and autonomy algorithms. However, its design is outdated, relying on CO_2 for propulsion and requiring significant support from the ISS crew to operate [30][5]. Requirement #46 in Table 2 captures a property related to localisation (that we will discuss in Section 4.2).

This section provided an overview of the main types of space systems that were explored in our literature review. Included in the literature review were some more generic, relevant aerospace systems; they were omitted here for brevity but they are included in the full list in the Appendix and our repository.

4.2 New Patterns with LTL and FRET Formalisation

Based on our analysis of the literature, we identified 5 new patterns (PHASES, TRANSMIT, RECONNECT, STAY-IN-PERIMETER, KEEP-OUT-ZONE) that fall into three distinct categories: **Mode Sequencing**, **Communication** and **Localisation**. These are illustrated in orange in Figure 2. We also found several variations of the existing patterns from [24, 34] that we discuss in Section 4.3.

Mode Sequencing is an important aspect of many cyber-physical systems and this was also apparent in the space applications that we explored. In this category we identified and formalised one pattern as discussed below.

PATTERN 1 (PHASES): We found several space applications such as Mars 2020 [31] and BepiColumbo [33], that incorporate phases during operating modes. Intuitively, the phase evolution requirements that we observed in the literature review described the linear transition from one phase to another as individual phases complete. In the work that we considered, there was normally a terminal phase in these requirements. We describe this using LTL as follows. Let $P = p_1, \ldots, p_n$ be a set of phases and $C = c_1, \ldots, c_n$ be a set of conditions that are caused by phases and cause phases to begin. Each c_i is produced by phase p_i and will trigger the start of phase p_{i+1}.

$$((G(((!p_1) \wedge (Xp_1)) \rightarrow (X(Fc_1)))) \wedge (p_1 \rightarrow (Fc_1)))$$
$$\wedge ((G(((!c_1) \wedge (Xc_1)) \rightarrow (X(Xp_2)))) \wedge (c_1 \rightarrow (Xp_2))) \ldots$$

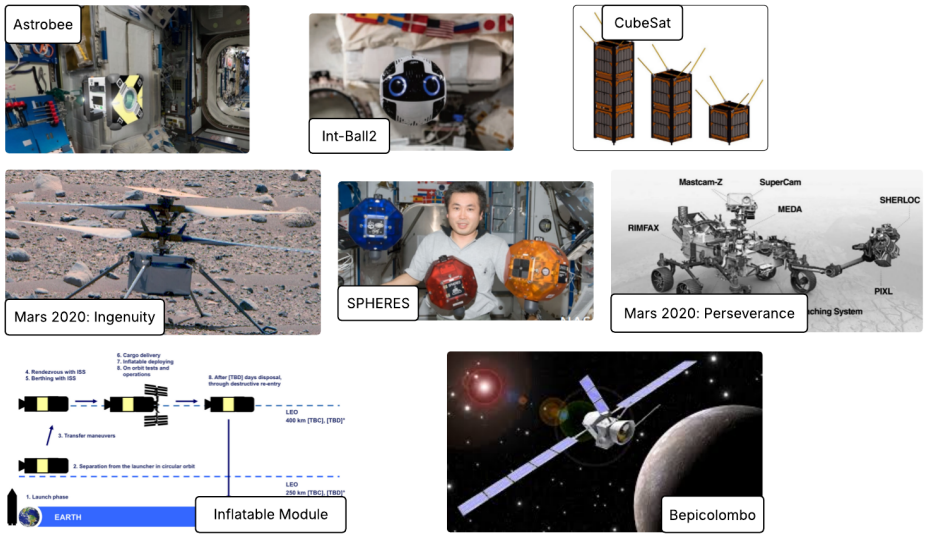

Figure 1: The space missions that are described in Section 4.1. These images are taken from the surveyed literature, NASA, ESA and JAXA sources.

We use multiple[4] FRETISH requirements to describe this kind of phase evolution as follows:

upon p1 System shall eventually satisfy c1
+ upon c1 System shall at the next timepoint satisfy p2
+ upon p2 System shall eventually satisfy c2

We can instantiate this pattern with part of a requirement from the Inflatable case study (also illustrated in Figure 1) [35]:

> *'The launch phase begins with lift-off and ends at burn out. The Separation phase begins with burn out, leading to transfer orbit insertion. During transfer, the spacecraft moves toward the Cygnus arrival near the ISS. Finally, the rendezvous phase covers the approach and capture by the robotic arm."*

upon LaunchPhase System shall eventually satisfy burnout
+ upon burnout System shall at the next timepoint satisfy SeperationPhase
+ upon SeperationPhase System shall eventually satisfy transferorbit

This pattern captures the concept of phases that is ubiquitous in many space systems. It ensures that higher level requirements such as phases or steps flow correctly one after the other. This pattern is quite novel when compared to the pre-existing catalogue of 28 requirements [24, 34]. Structurally, it shares some similarities with the LTL from the SEQUENCED VISIT patterns in [24]. However, the intent of phase evolution is significantly different from visiting specific locations.

Communication is an essential element of space systems since they operate at vast distances away from physical human contact. We found the following two patterns that fit into this category.

PATTERN 2 (TRANSMIT): Data can be transmitted when a systems' connection requirements are met. Let c_i represent the necessary connections, d_i represents the groups of data to be transmitted and T denotes the transmitting protocol. If all of the necessary connections are present and there is data that needs

[4]Following [34], we use '+' to indicate the logical conjunction of multiple requirements.

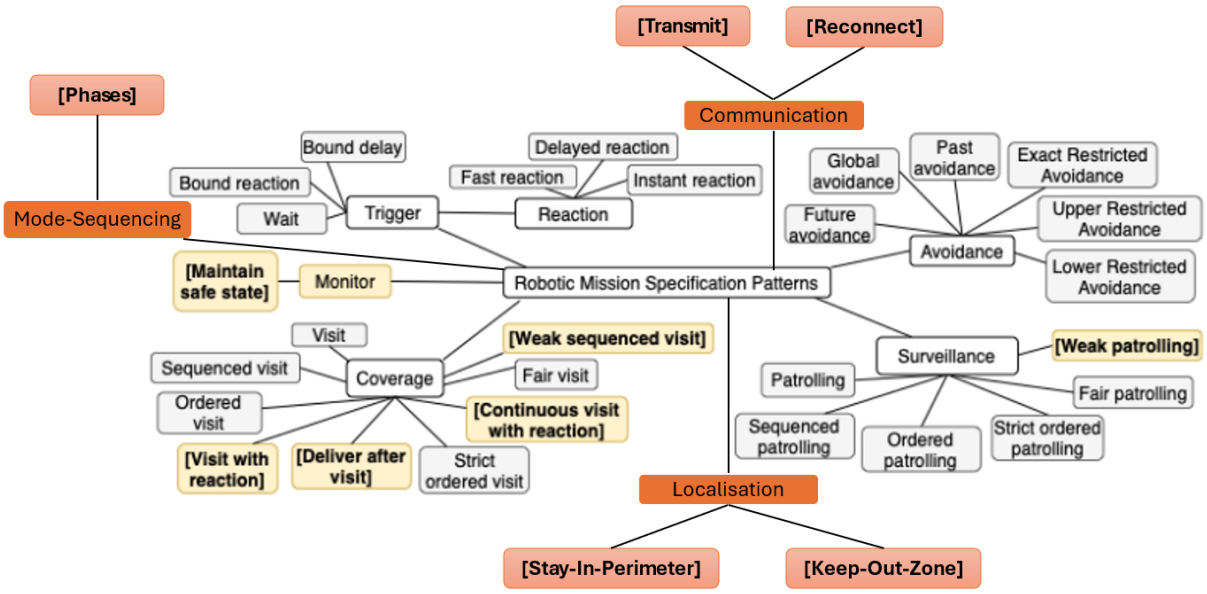

Figure 2: Robotic mission patterns, those with a white/grey background were proposed in [24], those with a yellow background were proposed in [34] and our new patterns are shown with an orange background.

to be sent, the transmission will continue until all of the data is sent. We express this in LTL as follows

$$G\big(((\bigwedge_{i=1}^{n} c_i) \wedge (\bigwedge_{i=1}^{n} \neg d_i)) \rightarrow (T \mathbin{\mathscr{W}} \bigwedge_{i=1}^{n} d_i))\big)$$

Many space systems transmit collections of data to ground control or other systems. Transmission is necessary for communication and receiving/sending mission critical information. Some systems rely on the connection of more than one device hence why this pattern covers multiple connections. This pattern incorporates part of the WAIT pattern from the 2019 catalogue as the structure of the consequent [24].

We express this pattern in FRETISH as follows.

`Whenever c & !d System` shall `until d satisfy T`

We can instantiate this pattern with a requirement from Astrobee [5]:

> *"After a sortie, Astrobee transfers large files through a hard-wired Ethernet connection with its dock."*

`Whenever Ethernet & ISSConnection & !LargeFiles Astrobee` shall
`until LargeFiles satisfy transmit`

Note that the ISSConnection condition was implicit in the requirement, it was derived from the documentation.

PATTERN 3 (RECONNECT): Connection can be lost and it is essential to restore that connection. Here, k_i represents connections and R represents the reconnection protocol. If one necessary connection is down then reconnection will occur until all necessary connections are formed.

$$G((\bigvee_{i=1}^{n} \neg k_i) \rightarrow (\mathrm{R} \mathbin{\mathscr{W}} \bigwedge_{i=1}^{n} k_i))$$

Due to the nature of connections in space, it is common for loss of signal to occur. Many of those systems that deal with transmitting data are bound to need to reconnect. As a result, this pattern is widely applicable to many systems. Examples of connections: ISS, Ethernet, satellites and ground control. Similar to the TRANSMIT pattern, this one also uses part of the WAIT pattern from the 2019 catalogue as the structure of the consequent [24]. We express this using FRET as follows.

Whenever !k1 | !k2 System shall until (k1 & k2) satisfy R

We can instantiate this pattern with a requirement from Astobee [5]:

"The space-to-ground network is subject to frequent losses of signal. After loss-of-signal (LOS) Astrobee shall attempt to reconnect."

Whenever !GroundSignal | !ISSConnection Astrobee shall
until (ISSConnection & GroundSignal) satisfy Reconnect

We note that both **Communication** patterns use weak, rather than strong, until. This makes implicit the ability for the communication to fail, as is common in space applications. Specifically, if we had used strong until then, we would require that the transmission (resp. reconnection) protocols eventually succeed, which may not be possible in practice.

Localisation is difficult for space applications since they can't simply rely on GPS and the environment within which they operate doesn't always contain easily recognisable features such as landmarks. That said, operating within safe boundaries was a concern in the papers that we reviewed. As a result, we identified the following two patterns.

PATTERN 4 (STAY-IN-PERIMETER): This pattern is used in systems that are required to stay within certain boundaries when specific actions are executed. In the LTL formula below, a represents actions such as movement, approach and velocity matching. Then, l_i represents the areas/boundaries that the system must remain within.

$$G(a \rightarrow (\bigwedge_{i=1}^{n} l_i))$$

During the movement/actions of some autonomous space robotics, robots are restricted to only moving/acting inside the perimeter. These safety restrictions are necessary for dynamic or uncontrolled environments for many moving systems. This pattern is structurally similar to the INSTANT REACTION pattern with a similar intent to the MAINTAIN SAFE SPACE pattern from the previous catalogue [34]. We express this using FRET as follows.

Whenever a System shall immediately satisfy l1 & l2

We can instantiate this pattern with a requirement from Int-Ball [17]:

"Int-Ball cannot operate without a direct line-of-sight (LOS) to its markers."

Whenever operating IntBall2 shall immediately satisfy LOS1 & LOS2

PATTERN 5 (KEEP-OUT-ZONE): This is essentially the dual of the previous pattern. We found several cases where systems should avoid certain areas while executing certain actions. As before, a represents actions and l_i represents the areas that the system must avoid.

$$G(a \rightarrow (\bigwedge_{i=1}^{n} \neg l_i))$$

Unlike the previous pattern, this one focuses on avoiding areas, rather than remaining in certain areas. This is essential for avoiding dangerous areas or critically important areas. We formalise this pattern in FRET as follows.

Whenever a System shall immediately satisfy !l1 & !l2

We can instantiate this pattern with a requirement from Astrobee [5]:

"Astrobee's navigation and control systems understand the concept of a keep-out zone (KOZ). KOZs are defined as areas where Astrobee is not allowed to fly."

Whenever moving Astrobee shall immediately satisfy !KOZ1 & !KOZ2

This pattern is very similar to the Future Avoidance pattern from [24] which is expressed in FRETISH [34] as, for example "whenever a Robot shall never satisfy l1". This gives the LTL formula $G(a \rightarrow$

$(G(\neg l_1)))$. Our version is more strict than this and the LTL formula that we use actually encompasses the original FUTURE AVOIDANCE pattern from [24]. In their pattern, the robot is not prohibited from entering l_1 at the same time instant that a is true. Our pattern prohibits this behaviour which is what, we believe, was the original intention of the requirements that we saw in this category.

4.3 Requirement Pattern Metrics and Analysis

We include a representative set of requirements in Table 3 for discussion purposes. The full list of requirements is in the Appendix for the interested reader. We summarise the distribution of the requirement pattern categories that were used in Table 4. We use a white background for patterns that were present in the pre-existing sets from [24, 34], an orange background for our newly defined patterns, a blue background for variants of the pre-existing and new patterns, and a green background for potential new patterns that require further exploration.

Of our 116 requirements, 41 used pre-existing patterns. The majority, 34, of these used different kinds of reaction patterns: INSTANT REACTION, PROMPT REACTION, DELAYED REACTION and VISIT WITH REACTION. The other existing patterns that were used were the MAINTAIN SAFE SPACE and WAIT patterns. Examples of these instantiated requirement patterns can be seen in requirements #3 (PROMPT REACTION) and #56 (INSTANT REACTION) in Table 3.

Most of our requirements (52 out of 116) used variations of pre-existing patterns. These variations were not seen in [24, 34]. We also had 1 instance of a variation of our new TRANSMIT pattern. In total, we identified 12 distinct pattern variations and most of these (7 out of 12) were variations that involved system modes through the scope field in FRETISH. For example, we see modal versions of several patterns in Table 3, including #2 (MODAL DELAYED REACTION) and #11-B (MODAL TRANSMIT).

Other pattern variations in Table 3 include what we label as TRIGGERED versions of existing patterns. For example, requirement #7 in Table 3 shows an instance of the TRIGGERED PROMPT REACTION pattern. The idea of trigger conditions is part of the formalisation in FRETISH where conditions can either be triggers (using `if`, `upon`, etc.) or holding (using `whenever`). We saw a mix of these condition types throughout our formalisation process and FRET forced us to consider which condition should be used for each requirement. The difference is that for a trigger, the response must hold `upon` a condition becoming true. A holding condition means that the response holds `whenever` the condition is true. Here, the use of FRET helped us to consider subtle aspects of the requirements.

We identified 3 potential new patterns that require further investigation. The first of these, MAINTAIN MODE IN HIERARCHY, was found in one of the more generic aerospace requirements for the Lift Plus Cruise study [26]. Essentially, the system must maintain a mode within another system mode (requirement #41 in Table 3). We didn't see this in other system requirements however, we did have phase updates within system modes as part of the IM mission [35]. We encoded these using the PHASES pattern but we would need to extend this further to capture the behaviour of MAINTAIN MODE IN HIERARCHY. Since we only found one instance of it and the natural language was ambiguous, we decided to leave this as a potential pattern that might be returned to in an expanded literature review to determine whether it constitutes a pattern in its own right.

The potential pattern called MODAL SCHEDULING was also found in the IM system requirements [35]. Requirement #10-A in Table 3 specifies that various checks should be carried out *before* the system tests are executed. We had not seen this pattern elsewhere but we do think that with an expanded literature review that it would present itself as a pattern in the future. Similarly, we found 2 instances of a SEMI-AUTONOMOUS pattern in the more general aerospace use case in [6]. These requirements specify which side of the aircraft is the pilot flying side, which is autonomously controlled and how to switch between

ID	Natural Language Description	FRETISH	Pattern	Source
2	PIXL's hexapod can compensate for X-Y drift if it is found to exceed a predefined threshold.	In Experiment whenever !inthreshold Rover shall eventually satisfy inthreshold	MODAL DELAYED REACTION	[31]
3	In order to ensure that PIXL's XRF and OFS subsystems are behaving in the expected manner, the instrument's performance is periodically checked by measuring the onboard calibration target, and then comparing the results of those measurements against pre-flight measurements of those standards.	whenever timetocheck=1 PIXL shall at the next timepoint satisfy check	PROMPT REACTION	[31]
7	Only once the rover is safely on the martian surface will flight software command the preparation and downlink of EDL Camera images and microphone data.	Upon SafeLanding EDL shall at the next timepoint satisfy Preparation & DownLink	TRIGGERED PROMPT REACTION	[31]
10-A	Check mode all components necessary to check system's health before starting the tests are active.	In CHECKMODE IMCOMPONENT shall before StartingTests satisfy necessary_check_components_-active	MODAL SCHEDULING	[35]
11-B	Data are transmitted to SM to be elaborated, then transmitted to ISS and eventually to Ground Segment.	In NOMINALTESTINGMODE whenever SMConnection & ISSConnection & GroundSegment & !files IM shall until files satisfy transmit	MODAL TRANSMIT	[35]
29	At least one side shall be the pilot flying side.	FGS shall always satisfy PilotFlying <= 1	SEMI-AUTONOMOUS	[6]
36	The probability that the aircraft leaves the taxiway, i.e., |cte| >8 meters, shall be extremely low.	Aircraft shall with probability <= 0.001 eventually satisfy absReal(cte) >8	PROBABILISTIC MAINTAIN SAFE SPACE	[1]
41	The vehicle remains in the thrust-borne mode (TB) as long as kgs <= 20.0 knots and Hover Control (HC) mode is selected.	In HCmode whenever TBMode & Kgs <= 20 LPC shall always satisfy TBmode	MAINTAIN MODE IN HIERARCHY	[26]
56	If multiple Astrobees are active, the Control Station displays the positions of all of the Astrobees so that the operators are aware of the other activities and can avoid collisions.	Whenever numberofAtrobees > 1 CS shall immediately satisfy DisplayALL	INSTANT REACTION	[5]
77-A	However, if the vSLAM output remains unavailable for an extended period, the robot rotates in place until the feature points detected in the current view align with those in the stored map.	whenever vSLAMUnavailable IntBall2 shall until FeaturePointDetected satisfy RotateProtocol + whenever vSLAMUnavailable IntBall2 shall eventually satisfy FeaturePointDetected	CONDITIONAL WAIT	[17]

Table 3: Examples of requirements that use existing patterns, pattern variants and potential new patterns.

Pattern	Frequency	Pattern	Frequency
CONDITIONAL WAIT	1	DELAYED REACTION	9
INSTANT REACTION	14	KEEP-OUT-ZONE	1
MAINTAIN MODE IN HIERARCHY	1	MAINTAIN SAFE SPACE	5
MODAL DELAYED REACTION	15	MODAL INSTANT REACTION	1
MODAL MAINTAIN SAFE SPACE	13	MODAL PROMPT REACTION	1
MODAL REACTION	2	MODAL SCHEDULING	1
MODAL TRANSMIT	1	MODAL TRIGGERED INSTANT REACTION	2
PHASES	11	PROBABILISTIC MAINTAIN SAFE SPACE	2
PROMPT REACTION	9	RECONNECT	1
SEMI-AUTONOMOUS	2	STAY-IN-PERIMETER	2
TRANSMIT	3	TRIGGERED DELAYED REACTION	1
TRIGGERED INSTANT REACTION	12	TRIGGERED PROMPT REACTION	2
VISIT WITH REACTION	2	WAIT	2

Table 4: Distribution of patterns throughout the requirements that we found. We use a white background for patterns that were present in the pre-existing sets from [24, 34], an orange background for our newly defined patterns, a blue background for variants of the pre-existing and new patterns and a green background for potential new patterns that require further exploration.

them, for example requirement #29 in Table 3. We believe that this kind of requirement will become more prevalent as more semi-autonomous systems are deployed. However, we found the natural language to be quite vague, so we have left this one for further exploration.

It was clear that system modes played an important role in space systems that was not as apparent in previous work [24, 34]. In our new patterns, most instances (11 out of 18) corresponded to our newly defined PHASES pattern which is used to specify system phase or mode evolution. Our **Communication** (respectively **Localisation**) patterns accounted for 4 out of 18 (respectively 3 out of 18) instances.

4.4 Expert Evaluation

We conducted an initial evaluation of our newly identified patterns with experts who have experience working on space robotics as well as similar systems in other domains, including nuclear and aerospace. This comprised 1 academic from the University of Manchester, 1 researcher from an industrial robotics engineering company and 3 experts from governmentally funded space organisations (NASA, JAXA and Satellite Applications Catapult).

We interviewed each of these participants using a set of slides to guide the discussion[5]. The slides contained the new patterns, an example of each in LTL and FRETISH and a series of questions for the participants. These questions asked whether they had seen this pattern in use before, were there any variants that we should consider, could they provide us with example uses from their experience and more general feedback. We interviewed most of the participants individually, grouping two of them together to facilitate busy schedules. The pair that were grouped shared a similar experience working on the same project in recent years. One participant provided answers and feedback via email because it was not possible to schedule an interview at the time. They were given the same set of slides as the other participants. We summarise their feedback on a per-pattern basis below and provide overall reflections.

PATTERN 1 (PHASES): In general, the experts had encountered requirements that specified phase or mode transitions, though not usually hierarchical modes and failures, which is what we found in

[5]The slides are available in our repository: `https://github.com/mariefarrell/spacepatterns`

[26]. This pattern had been previously encountered by the experts in aerospace and nuclear scenarios, including unmanned aircraft traffic management[6], (nuclear) operational sequences, Earth observation satellites, satellite manoeuvring and planetary rovers (entry, descent, landing, calibration and connection establishment). One expert remarked that there could be an intermediate "changing phases" step since phase transitions are not always instantaneous in practice. Further, complex systems may have branching phases to enable failure modes, which is not something that we had observed in our dataset but this point was echoed by several experts.

PATTERN 2 (TRANSMIT): The experts had encountered this pattern in communications-related requirements but there are further variants that are worth consideration. Specifically, this requirement does not capture a communication being instantiated, that the data is ready to send or account for the different types of communication that may be used such as Ethernet, radio, etc. As such, there are likely more low-level details that need to be instantiated to use this requirement pattern in practical systems. In a related astronaut-rover use case [36], this pattern would have been split into (1) maintain connection with the astronaut and (2) transmit data. Some experts noticed this requirement in existing use cases related to Earth observation satellites. In these cases, there would likely be more fine-grained transmission timings that might also be included. For example, transmissions would be limited to take a specified number of seconds.

PATTERN 3 (RECONNECT): The experts had seen this in systems involving swarms of robots, satellite constellations and a confined space drone. This pattern does not specify how long reconnection might take but, in practice, the system should probably only try and fail for so long before it resorts to another action. For example, it may need to go into a safe/wait/power-save mode and attempt reconnection again at a later stage. It may even need to autonomously physically reorient or move to a position where it can still harvest solar power to maintain operation, both while it awaits reconnection and as part of its attempt to reconnect. It was noted that this might be split into numerous requirements for each connection type but this would be a domain/application-specific design choice. One expert noted that, in practice, future variants of this pattern may need to capture real-time requirements, which we did not see in our dataset.

PATTERN 4 (STAY-IN-PERIMETER): This pattern was frequently encountered by all of our experts in robotics use cases including a fork-lift use case [22], proximity operations for formation flying satellites, satellites maintaining position, precision targetting, manipulator robots, planetary rovers that must maintain a safe distance from a lander and scenarios that involve robot collectives e.g. swarms, robot soccer, etc. There are also more subtle cases where designers may want to consider a 'gravity-well' attractor force to some position. For example, imagine an Astrobee/SPHERES robot that must maintain its (relative) position in space in/to the ISS, so it is commanded to move to that position; if necessary, it might have to move out of the way to avoid an object of space debris, but then return to its position once clear.

PATTERN 5 (KEEP-OUT-ZONE): Although this is very similar to the previous pattern, it was agreed that the intention was different enough to merit it being a pattern in its own right. This was identified as an especially important pattern for safety systems. This pattern had been observed by experts in transfer vehicles delivering consumables to the ISS, planetary (lunar) rovers that must avoid deep craters, unmanned aircraft traffic management and nuclear scenarios. One expert remarked that it could potentially be adapted to dynamically account for trajectory or even adversary avoidance. In manipulator robotics, there is a possible variant to include a "slow-down-zone" for

[6] https://store.astm.org/f3548-21.html

when the robot gets close to a keep-out-zone. This variant was not present in our dataset. Several experts also remarked that, although these dangerous zones should be avoided, it is feasible that a robot may enter such a zone for a short, tolerable period of time before returning to the safe zones. This would require an ability to express real-time requirements, which we do not have in standard LTL or the current version of FRET.

Overall Comments: Our experts observed that requirements are rarely completely deterministic in practice, likely incorporating probabilistic uncertainty. Probabilistic requirements can be expressed in FRET but we only encountered a small number of probabilistic requirements in our dataset (2 out of 116). One of these probabilistic requirements is contained in Table 3 (requirement #36) which is a probabilistic version of the MAINTAIN SAFE SPACE pattern. Of course, any of the other requirement patterns could be easily made probabilistic, so treat this as a variant of existing rather than a new pattern in Table 4. A comparison of related work on probabilistic specification patterns from [23] comprises an interesting future research direction for this work. Related to uncertainty, several experts noted the lack of a specification or any assumptions about the dynamic and unpredictable environment within which the systems are operating in our patterns.

A limitation of LTL for these specifications that was observed by multiple experts is the likely need for first-order logical quantifiers to accurately express requirements that are related to multi-robot systems such as swarms, satellite constellations and planetary robot teams. For example, we may want to be able to specify requirements such as all swarm robots must maintain communication between all other swarm robots. Requirement #56 in Table 3 is of this spirit and would likely be more precise if we were able to use quantifiers in FRET. That said, extending FRET in this way would be a significant (but worthwhile) undertaking as the move to first-order temporal logic would need to be reflected in both the underlying framework and the linked analysis tools.

Almost all of the experts that we interviewed remarked that we did not have patterns that captured error handling, fault tolerance and/or failure recovery. One suggestion was to include patterns related to Failure Mode and Effects Analysis (FMEA) [32] as a future extension. Related to this would be requirement patterns related to resource management, such as battery consumption. This should be particularly relevant for space applications. We did find one requirement related to the IntBall mission (Requirement #77-B in the Table 3) that specifies that the robot should automatically recharge when its battery power is low. However, we did not find other such requirements so we have not designated a separate pattern for this; in fact, it fits the Prompt Reaction pattern from [24].

Another interesting point is that we had originally used the `whenever` condition in the PHASES pattern and this was the version that we presented to the experts. After much discussion, it was agreed that a triggered condition more accurately captured our desired semantics so we changed this to an `upon` condition in the pattern shown in Section 4. The main reason for this is that phase changes must be triggered (become true from false) at a specific time instant rather than continually holding.

Although we examined these requirements in the context of space missions, several experts remarked that they could be useful in other domains with significant crossover in nuclear applications that they had seen. This is somewhat ironic since, at the start of this work, we set out to find space domain-specific requirements. However, it is useful that we found patterns that can apply across distinct domains.

5 Discussion: Answering the Research Questions

We now return to the two research questions that this project started with. We begin by discussing our first research question:

RQ1: Are existing specification patterns relevant and sufficient for autonomous space robotic missions?

We address this research question through the metrics that we presented in Section 4.3. Many of the pre-existing patterns from [24, 34] were used in our dataset, exhibiting their usefulness for space applications. So for the first part of the question, the answer is yes. The existing patterns are indeed relevant for space applications. However, for the second part of the question, they are not quite sufficient. We needed to define several variants of the existing patterns and 5 new patterns. We also found a small number of requirements that did not clearly fit into any of the patterns that already existed or were defined in this paper so further analysis is required as future work to determine if the potential patterns that we identified become patterns in their own right. This was also reflected in the expert feedback that we received, where the experts identified many variants of the new patterns as important based on their experience in the space sector. These variants included probabilistic, real-time and first-order temporal logic versions of patterns. This area clearly needs more work and an even more detailed study, including pattern use in the development of future space systems, is needed to further expand and more thoroughly evaluate which patterns are most useful for and needed in space applications. In addition, a consideration of how patterns are related would provide an interesting avenue of future work. This would examine whether a hierarchy of patterns would be useful and provide guidance on how patterns could be composed in, for example, a compositional verification framework.

RQ2: Can specification patterns for autonomous space robotic missions be expressed using FRET?

For the second research question, we followed the approach in [34] and expressed our specification patterns as requirements in FRET. The patterns that we identified were all straightforwardly expressed in FRETISH using FRET. However, many of the useful variants that were identified by the experts are not currently expressible in FRETISH. These include first-order temporal formulas as well as those incorporating real-time aspects. These are potential and significant extensions to FRET, which would require further detailed research to enable. Interestingly, the FRETISH structured natural language influenced our thought process when formalising the requirements, primarily through the use of the `scope` and `condition` fields. Specifically, the `scope` field provided us with a natural way to express many of the requirements that referred to system operating modes. Further, we felt that having to distinguish between trigger and holding conditions forced us to be more precise during formalisation. However, as many of the natural-language requirements were quite vague, it is possible that a domain expert might have opted for the other condition type in some cases. Notably, FRET's use of weak, rather than strong, until was a very positive point as weak until more closely captured the intention of the **Communication** patterns that we identified. It was useful that this was natural in FRETISH.

6 Conclusion and Future Work

As space missions become ever more sophisticated and reliant on autonomous functionality, ensuring correctness through suitable, logically expressed, requirements is essential. In this paper, we sought to examine whether existing robotic specification patterns were applicable in space missions and whether new patterns were necessary for this domain. We answered these questions by providing a literature review which contributed a corpus of 116 formalised requirements from existing space missions. Based on these 116 requirements, we demonstrated that existing patterns were indeed applicable in the space domain. However, we also contributed five new patterns and several pattern variants to adequately capture the requirements that we found. We also identified a number of potential new patterns that require

further research. We provide detailed insight into the new patterns through our expert evaluation study which highlighted the utility and limitations of the new patterns that were identified. Our contributions provide a baseline set of space requirements and associated patterns that have spawned several interesting directions of future work including further analysis of potential patterns, real-time and first-order extensions to FRETISH, and a large-scale validation of these patterns in a space mission.

Notably, our current baseline requirement patterns for autonomous space robotics could be expanded by exploring space missions in general across multiple space agencies. This broader analysis of space missions would provide new architectural patterns to execute a mission as the patterns used by a given space agency may not be the only way to specify a mission. Moreover, future work could focus on more application-driven user case studies where domain-specific experts would be introduced to the patterns, subsequently apply them using FRET and finally implement the system using e.g. ROS to provide traceability from requirement patterns through to implementations. Such studies would not only validate the patterns in a more practical setting but also encourage adoption by the wider engineering community.

Acknowledgements: We thank our expert evaluators for their insightful discussion and feedback: Michael Fisher (The University of Manchester), John Brotherhood (Amentum), Tsutomu Kobayashi (JAXA), Andreas Katis (KBR Inc./NASA Ames Research Center) and Dimitris Xydas (Satellite Applications Catapult). This work was supported by the Summer 2025 Google DeepMind Research Ready internship through a partnership with the Royal Academy of Engineering, Google DeepMind and the Hg Foundation. It was also supported by a Royal Academy of Engineering Research Fellowship.

References

[1] Adrian Agogino, Guillaume Brat, Yuning He, Daniel Hulse, Rory Lipkis, Thomas Pressburger, Divya Gopinath, Lukman Irshad, Andreas Katis, Anastasia Mavridou et al. (2024): *Recommendations on evidence and process for certification of learning-enabled components in aerospace systems*. Technical Report.

[2] Marco Autili, Lars Grunske, Markus Lumpe, Patrizio Pelliccione & Antony Tang (2015): *Aligning qualitative, real-time, and probabilistic property specification patterns using a structured english grammar*. *IEEE Transactions on Software Engineering* 41(7), pp. 620–638, doi:10.1109/TSE.2015.2398877.

[3] Atef Azaiez, David A Anisi, Marie Farrell & Matt Luckcuck (2025): *Revisiting Formal Methods for Autonomous Robots: A Structured Survey*. In: *Towards Autonomous Robotic Systems*, Springer, pp. 338–352, doi:10.1007/978-3-032-01486-3_26.

[4] Domenico Bianculli, Carlo Ghezzi, Cesare Pautasso & Patrick Senti (2012): *Specification patterns from research to industry: a case study in service-based applications*. In: *International Conference on Software Engineering*, IEEE, pp. 968–976, doi:10.1109/ICSE.2012.6227125.

[5] Maria G. Bualat, Trey Smith, Ernest E. Smith, Terrence Fong, D. W. Wheeler & the Astrobee Team (2018): *Astrobee: A New Tool for ISS Operations*. In: *Proceedings of SpaceOps (AIAA 2018-2517)*, doi:10.2514/6.2018-2517.

[6] Darren Cofer & Steven P. Miller (2023): *Formal Methods Case Studies for DO-333*. NASA.

[7] Judith Crow & Ben Di Vito (1998): *Formalizing Space Shuttle Software Requirements: Four case studies*. *ACM Transactions on Software Engineering and Methodology* 7(3), pp. 296–332, doi:10.1145/287000.287023.

[8] Matthew B Dwyer, George S Avrunin & James C Corbett (1999): *Patterns in Property Specifications for Finite-State Verification*. In: *International Conference on Software Engineering*, pp. 411–420, doi:10.1145/302405.302672.

[9] Mike Hinchey Emil Vassev (2021): *Autonomy Requirements Engineering for Space Missions*. Springer, doi:10.1007/978-3-319-09816-6.

[10] Marie Farrell, Matt Luckcuck, Laura Pullum, Michael Fisher, Ali Hessami, Danit Gal, Zvikomborero Murahwi & Ken Wallace (2021): *Evolution of the IEEE P7009 standard: Towards fail-safe design of autonomous systems*. In: International Symposium on Software Reliability Engineering Workshops, IEEE, pp. 401–406, doi:10.1109/ISSREW53611.2021.00109.

[11] Michael Fisher (2011): *An introduction to practical formal methods using temporal logic*. John Wiley & Sons, doi:10.1002/9781119991472.

[12] Angel Flores-Abad, Ou Ma, Khanh Pham & Steve Ulrich (2014): *A Review of Robotics Technologies for On-Orbit Services*. Progress in Aerospace Sciences 68, pp. 1–22, doi:10.1016/j.paerosci.2014.03.002.

[13] Dimitra Giannakopoulou, Anastasia Mavridou, Julian Rhein, Thomas Pressburger, Johann Schumann & Nija Shi (2020): *Formal Requirements Elicitation with FRET*. In: International Working Conference on Requirements Engineering: Foundations for Software Quality, Springer.

[14] Dimitra Giannakopoulou, Thomas Pressburger, Anastasia Mavridou & Johann Schumann (2021): *Automated formalization of structured natural language requirements*. Information and Software Technology 137, p. 106590, doi:10.1016/j.infsof.2021.106590.

[15] Lars Grunske (2008): *Specification patterns for probabilistic quality properties*. In: International Conference on Software Engineering, pp. 31–40, doi:10.1145/1368088.1368094.

[16] Steven R Hirshorn, Linda D Voss & Linda K Bromley (2017): *NASA Systems Engineering Handbook*. Technical Report.

[17] NASA / JAXA (2024): *Int-Ball2: On-Orbit Demonstration of Autonomous Intravehicular Flight and Docking for Image Capturing and Recharging*. IEEE Robotics & Automation Magazine, doi:10.1109/MRA.2024.3505776.

[18] Andreas Katis, Anastasia Mavridou, Dimitra Giannakopoulou, Thomas Pressburger & Johann Schumann (2022): *Capture, analyze, diagnose: Realizability checking of requirements in FRET*. In: International Conference on Computer Aided Verification, Springer, pp. 490–504, doi:10.1007/978-3-031-13188-2_24.

[19] Andreas Katis, Anastasia Mavridou, Dimitra Giannakopoulou, Thomas Pressburger & Johann Schumann (2022): *Realizability checking of requirements in FRET*. Technical Report, NASA, doi:10.1007/978-3-031-13188-2_24.

[20] Sascha Konrad & Betty HC Cheng (2005): *Real-time specification patterns*. In: International Conference on Software Engineering, pp. 372–381, doi:10.1145/1062455.1062526.

[21] Matt Luckcuck, Marie Farrell, Louise A Dennis, Clare Dixon & Michael Fisher (2019): *Formal specification and verification of autonomous robotic systems: A survey*. ACM Computing Surveys 52(5), pp. 1–41, doi:10.1145/3342355.

[22] Shahar Maoz & Jan Oliver Ringert (2016): *Synthesizing a lego forklift controller in GR (1): A case study*. arXiv preprint arXiv:1602.01172, doi:10.48550/arXiv.1602.01172.

[23] Claudio Menghi, Christos Tsigkanos, Mehrnoosh Askarpour, Patrizio Pelliccione, Gricel Vázquez, Radu Calinescu & Sergio García (2022): *Mission specification patterns for mobile robots: Providing support for quantitative properties*. IEEE Transactions on Software Engineering 49(4), pp. 2741–2760, doi:10.1109/TSE.2022.3230059.

[24] Claudio Menghi, Christos Tsigkanos, Patrizio Pelliccione, Carlo Ghezzi & Thorsten Berger (2021): *Specification Patterns for Robotic Missions*. IEEE Transactions on Software Engineering 47(10), pp. 2208–2224, doi:10.1109/TSE.2019.2945329.

[25] Tom Pressburger, Andreas Katis, Aaron Dutle & Anastasia Mavridou (2023): *Authoring, analyzing, and monitoring requirements for a lift-plus-cruise aircraft*. In: International Working Conference on Requirements Engineering: Foundation for Software Quality, Springer, pp. 295–308, doi:10.1007/978-3-031-29786-1_21.

[26] Tom Pressburger, Andreas Katis, Aaron Dutle & Anastasia Mavridou (2023): *Using FRET to Create, Analyze and Monitor Requirements for a Lift Plus Cruise Case Study*. Technical Report NASA/TM–20220017032, NASA Technical Reports Server (NTRS).

[27] Olman D. Quiros-Jimenez & Duncan d'Hemecourt (2019): *Development of a Flight Software Framework for Student CubeSat Missions*. *Tecno- logía en Marcha.*, doi:10.18845/tm.v32i8.4992.

[28] Signe Redfield, Joanna I Olszewska, Kevin Leahy, Zvikomborero Murahwi, Dejanira Araiza-Illan & Michael Fisher (2024): *Verification of autonomous systems: the road ahead*. In: *40th Anniversary of the IEEE International Conference on Robotics and Automation*, IEEE.

[29] Kristin Yvonne Rozier (2016): *Specification: The biggest bottleneck in formal methods and autonomy*. In: *Working Conference on Verified Software: Theories, Tools, and Experiments*, Springer, pp. 8–26, doi:10.1007/978-3-319-48869-1_2.

[30] Alvar Saenz-Otero & David W. Miller (2008): *SPHERES: Development of an ISS Laboratory for Formation Flight and Docking Research*. In: *Spacecraft Platforms and Infrastructure*, *Space Sciences Series of ISSI* 26, Springer, pp. 57–65, doi:10.1007/978-1-4020-6943-7_24.

[31] John W. Smith & Jane A. Doe (2020): *Mars 2020 Mission Overview*. Technical Report JPL D-12345, NASA Jet Propulsion Laboratory. Available at `https://mars.nasa.gov/mars2020/mission/overview/`.

[32] Diomidis H Stamatis (2003): *Failure mode and effect analysis*. Quality Press.

[33] Emil Vassev & Mike Hinchey (2014): *Verification and Validation of Autonomy Requirements*. In: *Autonomy Requirements Engineering for Space Missions*, NASA Monographs in Systems and Software Engineering, pp. 173–184, doi:10.1007/978-3-319-09816-6.

[34] Gricel Vázquez, Anastasia Mavridou, Marie Farrell, Tom Pressburger & Radu Calinescu (2024): *Robotics: A New Mission for FRET Requirements*. In: *NASA Formal Methods*, Springer, p. 8, doi:10.1007/978-3-031-60698-4_22.

[35] Maria Antonietta Viscio, Nicole Viola, Roberta Fusaro & Valter Basso (2015): *Methodology for Requirements Definition of Complex Space Missions and Systems*. *Acta Astronautica* 114, pp. 79–92, doi:10.1016/j.actaastro.2015.04.018.

[36] Matt Webster, Louise A Dennis, Clare Dixon, Michael Fisher, Richard Stocker & Maarten Sierhuis (2020): *Formal verification of astronaut-rover teams for planetary surface operations*. In: *2020 IEEE aerospace conference*, IEEE, pp. 1–8, doi:10.1109/AERO47225.2020.9172303.

A Table of Requirements

ID	Description	FRETISH	Pattern	Source
1	During an experiment: Two context images taken at different times of a PIXL experiment will be compared to detect unplanned movement (drift) of the rover arm likely to arise from temperature changes.	In Experiment PIXL shall eventually satisfy Take2Picture & ComparePicture	MODAL REACTION	[31]
2	PIXL's hexapod can compensate for X-Y drift if it is found to exceed a pre-defined threshold	In Experiment whenever !inthreshold Rover shall eventually satisfy inthreshold	MODAL DELAYED REACTION	[31]
3	In order to ensure that PIXL's XRF and OFS subsystems are behaving in the expected manner, the instrument's performance is periodically checked by measuring the onboard calibration target, and then comparing the results of those measurements against pre-flight measurements of those standards.	whenever timetocheck=1 PIXL shall at the next timepoint satisfy check	PROMPT REACTION	[31]
4	Once a safe target has been selected, the spacecraft adjusts its trajectory in propulsive powered flight to land at the target.	if safetargetlocated shuttle shall immediately satisfy AdjustToLand	TRIGGERED INSTANT REACTION	[31]
5	Deploy the parachute using navigated position information once safe parachute deployment velocities have been reached	if parachutedistance <= safeparachutedistance EDL shall immediately satisfy DeployParachute	TRIGGERED INSTANT REACTION	[31]
6	LVS begins taking pictures at 4.2 km altitude and matching them up to an onboard map.	Whenever Altitude <= 4.2 LVS shall immediately satisfy TakePictures & Match	INSTANT REACTION	[31]
7	Only once the rover is safely on the martian surface will flight software command the preparation and downlink of EDL Camera images and microphone data.	Upon SafeLanding EDL shall at the next timepoint satisfy Preparation & DownLink	TRIGGERED PROMPT REACTION	[31]
8	Stand-by mode: In IM stand-by mode only components necessary to monitor the system and to survive the external environment shall be active	In STANDBYMODE IM shall always satisfy necessary_components_only	MODAL MAINTAIN SAFE SPACE	[35]

9-A	Safe mode: In IM safe mode all components are activated at limited level (adopted in case of contingency)	In SAFEMODE IM shall always satisfy components_limited_level	MODAL MAINTAIN SAFE SPACE	[35]
9-B	The IM shall be isolated in case of contingency.	In SAFEMODE whenever CONTINGENCY IM shall immediately satisfy isolated	MODAL INSTANT REACTION	[35]
9-C	In case of contingency safe mode is employed	Whenever CONTINGENCY IM shall immediately satisfy SAFEMODE	INSTANT REACTION	[35]
10-A	Check mode all components necessary to check system's health before starting the tests are active	In CHECKMODE IMCOMPONENT shall before StartingTests satisfy necessary_check_components_active	MODAL SCHEDULING	[35]
10-B	If testing is imminent enter check mode	whenever TESTING_IMMINENT IM shall at the next timepoint satisfy Checkmode	PROMPT REACTION	[35]
11-A	Nominal testing mode: all components necessary to perform tests are active	In NOMINALTESTINGMODE IMCOMPONENT shall always satisfy necessary_testing_components_active	MODAL MAINTAIN SAFE SPACE	[35]
11-B	data are transmitted to SM to be elaborated, then transmitted to ISS and eventually to Ground Segment	In NOMINALTESTINGMODE whenever SMConnection & ISSConnection & GroundSegment & !files IM shall until files satisfy transmit	MODAL TRANSMIT	[35]
12	Nominal crew mode: all main functionalities are active and access of the crew to perform visual inspections is allowed.	In NOMINALCREWMODE IM shall always satisfy main_functionalities_active & crew_access	MODAL MAINTAIN SAFE SPACE	[35]
13-A	During the Launch Phase the only mode of operation in use is STANDBY mode which should be done when the IM is in a stowed configuration	In LAUNCHPHASE whenever stowed IM shall eventually satisfy STANDBYMODE	MODAL DELAYED REACTION	[35]
13-B	During the Separation Phase the only mode of operation in use is STANDBY mode which should both be done when the IM is in a stowed configuration	In SEPARATIONPHASE whenever stowed IM shall eventually satisfy STANDBYMODE	MODAL DELAYED REACTION	[35]
13-C	During the Transfer Phase the only modes of operation in use is STANDBY and SAFE mode which should both be done when the IM is in a stowed configuration	In TRANSFERPHASE whenever stowed IM shall eventually satisfy STANDBYMODE & SAFE	MODAL DELAYED REACTION	[35]

13-D	During the Rendezvous Phase the only modes of operation in use is STANDBY and SAFE mode which should both be done when the IM is in a stowed configuration	In RENDEZVOUSPHASE whenever stowed IM shall eventually satisfy STANDBYMODE & SAFEMODE	MODAL DELAYED REACTION	[35]
13-E	During the Berthing Phase the only modes of operation in use is STANDBY and SAFE mode which should both be done when the IM is in a stowed configuration	In BERTHINGPHASE whenever stowed IM shall eventually satisfy STANDBYMODE & SAFEMODE	MODAL DELAYED REACTION	[35]
13-F	During the Cargo delivery Phase the only modes of operation in use is STANDBY and SAFE mode which should both be done when the IM is in a stowed configuration	In CARGODELIVERYPHASE whenever stowed IM shall eventually satisfy STANDBYMODE & SAFEMODE	MODAL DELAYED REACTION	[35]
13-G	During the Inflatable deploying Phase the only modes of operation in use is CHECK, SAFE, NOMINAL TESTING mode which should be done when the IM is in a stowed or deployed configuration	In INFLATABLEPHASE whenever (stowed|deployed) IM shall eventually satisfy SAFEMODE & CHECKMODE & NOMINALTESTINGMODE	MODAL DELAYED REACTION	[35]
13-H	During the On orbit tests and ops Phase the only modes of operation in use is CHECK, SAFE, NOMINAL TESTING and NOMINAL CREW mode which should be done when the IM is in a deployed configuration	In ONORBITSPHASE whenever deployed IM shall eventually satisfy SAFEMODE & CHECKMODE & NOMINALCREWMODE & NOMINALTESTINGMODE	MODAL DELAYED REACTION	[35]
13-I	During the Undocking delivery Phase a mode of operation in use is STANDBY which should both be done when the IM is in a deployed configuration	In UNDOCKINGPHASE whenever deployed IM shall eventually satisfy STANDBY	MODAL DELAYED REACTION	[35]
13-J	During the Undocking delivery Phase a mode of operation in use is SAFE which should both be done when the IM is in a deployed or stowed configuration	In UNDOCKINGPHASE whenever (stowed|deployed) IM shall eventually satisfy SAFE	MODAL DELAYED REACTION	[35]
13-K	During the Destructive re-entry a mode of operation in use is STANDBY which should both be done when the IM is in a deployed configuration	In DESTRUCTIVEPHASE whenever deployed IM shall eventually satisfy STANDBY	MODAL DELAYED REACTION	[35]

14-A	The launch phase begins ends at burn out.	Upon LaunchPhase System shall eventually satisfy burnout	PHASES	[35]
14-B	The Separation phase begins with burn out.	Upon burnout System shall at the next timepoint satisfy SeparationPhase	PHASES	[35]
14-C	The Separation phase ends with transfer orbit insertion.	Upon SeparationPhase System shall eventually satisfy orbitinsertion	PHASES	[35]
14-D	Orbit insertion leads to the beginning of the transfer phase.	Upon orbitinsertion System shall at the next timepoint satisfy TransferPhase	PHASES	[35]
14-E	During transfer, the spacecraft moves toward the Cygnus arrival near the ISS	Upon TransferPhase System shall eventually satisfy cyngusarriaval	PHASES	[35]
14-F	Finally, the rendezvous phase covers the approach.	Upon cyngusarriaval System shall at the next timepoint satisfy RendezvousPhase	PHASES	[35]
14-G	Finally, the rendezvous phase covers the approach and capture by the robotic arm	Upon RendezvousPhase System shall eventually satisfy captureroboticarm	PHASES	[35]
15	Autonomously release the SEPM when the right jettison attitude is reached	Upon Currentattitude <= Rightattitude ReleaseBepiColombo shall immediately satisfy release_SEPM	TRIGGERED INSTANT REACTION	[33]
16	Autonomously release MMO when the polar orbit is reached	Upon polar_orbit_reached BepiColombo shall immediately satisfy release_MMO	TRIGGERED INSTANT REACTION	[33]
17	Autonomously determine a steering law	whenever operating BepiColombo shall eventually satisfy determine_steering_law	PROMPT REACTION	[33]
18	Use low thrust to achieve capture around Mercury	whenever capturing BepiColombo shall immediately satisfy low_-thrust	INSTANT REACTION	[33]
19	Autonomously acquire the escape procedure and use it to leave Mercury if necessary	Upon need_mercury_escape BepiColmbo shall immediately satisfy acquire_escape_procedure & escape	TRIGGERED INSTANT REACTION	[33]
20-A	Autonomously detect the presence of high solar irradiation	whenever high_solar BepiColombo shall eventually satisfy detect	DELAYED REACTION	[33]
20-B	In case of presence of high solar irradiation the system will be able to shield the electronics by turning them off	Whenever solar_irradiation >Normal_solar_radiation BepiColmbo shall immediately satisfy turn_off_electronics	INSTANT REACTION	[33]

20-C	In case of presence of high solar irradiation the system will be able to shield the electronics	Whenever solar_irradiation >Normal_solar_radiation BepiColmbo shall immediately satisfy shield_electronics	INSTANT REACTION	[33]
20-D	Autonomously detect the presence of high solar irradiation and get away if possible, by using chemical propulsion.	Whenever solar_irradiation >Normal_solar_radiation BepiColmbo shall immediately satisfy get_away_chemically	INSTANT REACTION	[33]
21	Autonomously maintain the onboard equipment and the spacecraft structure in proper temperature range.	BepiColmbo shall always satisfy MaintainEquipment & MaintainTemperature	MAINTAIN SAFE SPACE	[33]
22	The algorithm first selects the vernier jet or the group of primary jets whose acceleration has the largest scalar (dot) product with the desired rotational acceleration vector.	SRC shall at the next timepoint satisfy (SelectFirstJet | SelectPrimaryJets) & !(SelectFirstJet & SelectPrimaryJets)	PROMPT REACTION	[7]
23	If second and third jets are required, they are similarly selected on the basis of the second and third largest scalar products.	Whenever SecondJet | ThirdJet SRC shall immediately satisfy SelectNeededJet	INSTANT REACTION	[7]
24-A	If three jets satisfying the given thresholds cannot be found, the algorithm considers pairs, or, as a last resort, single jets	Whenever !ThreeJets SRC shall immediately satisfy Considerpairs	INSTANT REACTION	[7]
24-B	If three jets satisfying the given thresholds cannot be found, the algorithm considers pairs, or, as a last resort, single jets	Whenever !TwoJets SRC shall immediately satisfy Considersingle	INSTANT REACTION	[7]
25	During the final phase of Shuttle flight, the orbiter must enter a "heading alignment cylinder"	In FinalPhase Orbiter shall eventually satisfy HeadingAlignmentsCylinder	MODAL DELAYED REACTION	[7]
26	if three Shuttle main engines fail sequentially or simultaneously begin calculating/commanding safe abort manoeuvres.	Upon ThreeEngineFailure ThreeE_O shall immediately satisfy CalculatePlan & SafeManoeuvres	TRIGGERED INSTANT REACTION	[7]
27	If it is required to know the state of the spacecraft, even during the section of the orbit without a communication link with ground segment, store telemetry data.	If StateRequired & !CommunitcationLink CubeSat shall immediately satisfy StoreData	TRIGGERED INSTANT REACTION	[27]

28-A	in charge of providing the ground segment with telemetry data about the state and health of the spacecraft, therefore this service shall be able to automatically collect telemetry data.	`Cubesat shall always satisfy CollectData`	MAINTAIN SAFE SPACE	[27]
28-B	in charge of providing the ground segment with telemetry data about the state and health of the spacecraft, therefore this service shall be able to automatically store telemetry data.	`Cubesat shall always satisfy StoreData`	MAINTAIN SAFE SPACE	[27]
28-C	in charge of providing the ground segment with telemetry data about the state and health of the spacecraft, therefore this service shall be able to automatically transmit telemetry data.	`Whenever groundsegmentconnection & !telemtrydata Cubesat shall until telemtrydata satisfy Transmit`	TRANSMIT	[27]
29	At least one side shall be the pilot flying side.	`FGS shall always satisfy PilotFlying <= 1`	SEMI-AUTONOMOUS	[6]
30	At most one side shall be the pilot flying side.	`FGS shall always satisfy PilotFlying >= 1`	SEMI-AUTONOMOUS	[6]
31	Pressing the Transfer Switch shall always change the pilot flying side.	`Upon TransferSwitch FGS shall immediately satisfy SwitchSides`	TRIGGERED INSTANT REACTION	[6]
32	The system shall start with the Primary Side as the pilot flying side.	`Upon Startup FGS shall at the next timepoint satisfy PrimarySide`	TRIGGERED INSTANT REACTION	[6]
33	The system shall not change the pilot flying side unless the Transfer Switch is pressed.	`FGS shall until switch satisfy !SwitchSides + FGS shall eventually satisfy switch`	WAIT	[6]
34	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).	`Whenever Limits & !Standby & supported & !apfail FSM shall immediately satisfy Pullup`	INSTANT REACTION	[19]

35	While flying, remain separated from an intruder aircraft by at least 250 ft horizontally or 50 ft vertically	In FlightMode AirCraft shall always satisfy (horizontalIntruderDistance >250 | verticalIntruderDistance >50)	MODAL MAINTAIN SAFE SPACE	[1]
36	The probability that the aircraft leaves the taxiway, i.e., |cte| >8 meters, shall be extremely low	Aircraft shall with probability <= 0.001 eventually satisfy absReal(cte) >8	PROBABILISTIC MAINTAIN SAFE SPACE	[1]
37	The probability that the aircraft turns more than a prescribed degree ($|he| \leq 35°$) shall be extremely low	Aircraft shall with probability <= 0.002 eventually satisfy absReal(he) <= 35	PROBABILISTIC MAINTAIN SAFE SPACE	[1]
38	We also require that the rear propeller be always used, except in HC mode	If not in HCMode LPC shall always satisfy RearPropeller	MODAL MAINTAIN SAFE SPACE	[26]
39	If the vehicle is slowing down from the wing-borne mode (WB), the transition to semi-wing-borne (SWB) kicks in at an indicated airspeed of 90 knots (kias <= 90.0)	In Wbmode whenever airspeed <= 90 LPC shall eventually satisfy SWBMode	MODAL DELAYED REACTION	[19]
40	whereas if the vehicle is speeding up from a SWB mode, the transition to WB mode occurs at kias >100.0 knots	In SWBmode whenever airspeed >100 LPC shall eventually satisfy WBMode	MODAL DELAYED REACTION	[19]
41	The vehicle remains in the thrust-borne mode (TB) as long as kgs <= 20.0 knots and Hover Control (HC) mode is selected.	In HCmode whenever TBMode & Kgs <= 20 LPC shall always satisfy TBmode	MAINTAIN MODE IN HIERARCHY	[26]
42-A	during takeoff and landing, the aircraft motion is controlled by the lifting rotors only	In TakeoffMode LPC shall always satisfy LiftingRotors & !FlightSurfaces	MODAL MAINTAIN SAFE SPACE	[26]
42-B	during takeoff and landing, the aircraft motion is controlled by the lifting rotors only	In LandingMode LPC shall always satisfy LiftingRotors & !FlightSurfaces	MODAL MAINTAIN SAFE SPACE	[26]

43	On the other hand, during the higher speeds of the en-route phase, the wings provide lift, the rear propeller provides thrust, and the lifting rotors are inactive (wing-borne mode, WB)	In EnRoute LPC shall always satisfy !LiftingRotors & ThrustRearPropeller & WingsLift	MODAL MAINTAIN SAFE SPACE	[26]
44	In a SLM survey, crew takes measurements at locations described in procedures, attempting to take the measurement as close to the described point as possible	Astrobee shall eventually satisfy SoundLocation & SLMSurvey	VISIT WITH REACTION	[5]
45	This type of data could be supplemented with denser, though shorter duration, measurements from a mobile platform. The Radiation Environment Monitor (REM) hardware developed at the University of Houston and NASA Johnson Space Center is an example of the sort of small, light-weight sensor that Astrobee could carry to create higher resolution maps of the ISS environment.	Astrobee shall eventually satisfy RadiationLocation & RadiationSurvey	VISIT WITH REACTION	[5]
46	The SPHERES satellites, however, triangulate their position using infrared/ultrasonic beacons, preventing them from navigating outside the two-meter cube defined by the fixed beacon locations.	Whenever moving SPHERES shall immediately satisfy x<2 & y<2 & z<2	STAY-IN-PERIMETER	[17]
47	Like SPHERES, Int-Ball cannot operate without a direct line-of-sight to its markers.	whenever Operating IntBall shall immediately satisfy LOS1 & LOS2	STAY-IN-PERIMETER	[17]
48	The PerchCam is identical to the HazCam and it turns on to detect ISS handrails when Astrobee perches autonomously	Whenever Perched Astrobee shall immediately satisfy PerchCam	INSTANT REACTION	[5]
49	the top-facing SpeedCam sensor package provides an independent over-speed cutoff function, estimating velocity using its own optical flow, infrared ranging, and IMU sensors	Whenever Moving Astrobee shall always satisfy cutoff >currentspeed	MAINTAIN SAFE SPACE	[5]

50	After a sortie, Astrobee transfers large files through a hard-wired Ethernet connection with its dock	whenever ISSConnection & Ethernet & !LargeFile Astrobee shall until LargeFile satisfy Transfer	TRANSMIT	[5]
51	Once Astrobee grasps a handrail, it powers down its propulsion system.	Upon Perched Astrobee shall at the next timepoint satisfy !PropulsionSystem	TRIGGERED PROMPT REACTION	[5]
52	Initially, Astrobee will use these components primarily to help crew understand its state and intentions (for example, by providing turn signals)	whenever Turning Astrobee shall at the next timepoint satisfy Indicate	PROMPT REACTION	[5]
53	When docking, Astrobee autonomously approaches its berth using visual servoing relative to fiducials mounted to the dock	In DockingMode Astrobee shall eventually satisfy approachberth	MODAL REACTION	[5]
54	When mating is complete, permanent magnets on the berth attract striker plates on the robot, providing a passive retention force	Upon MatingComplete DS shall eventually satisfy StrikeMagnets	TRIGGERED DELAYED REACTION	[5]
55	To enable undocking, linear actuators within the berths pull the magnets away from the striker plates, allowing the propulsion system to easily overcome the reduced magnetic force	In UndockingMode DS shall at the next timepoint satisfy LinearActuators	MODAL PROMPT REACTION	[5]
56	If multiple Astrobees are active, the Control Station displays the positions of all of the Astrobees so that the operators are aware of the other activities and can avoid collisions.	Whenever numberofAtrobees > 1 CS shall immediately satisfy DisplayALL	INSTANT REACTION	[5]
57	Operators use the Plan Editor tab in the Control Station to construct and validate sequences of commands for Astrobee (".fplans"), that include waypoints and actions to perform at the waypoints.	whenever CommandReceived Astrobee shall eventually satisfy PerformCommand	DELAYED REACTION	[5]
58-A	Astrobee can lose signal, when signal is lost Astrobee enters LOS-Mode	whenever LostSignal Astrobee shall immediately satisfy LOSMode	INSTANT REACTION	[5]

58-B	During loss-of-signal (LOS) with the ground, Astrobee continues to hold its position while recording and storing video on its internal file system.	In LOSMode Astrobee shall always satisfy Hold & WorkInternally	MODAL MAINTAIN SAFE SPACE	[5]
58-C	Once ground signal has been reacquired	Whenever !ISSConnection & !Groundsignal Astrobee shall until ISSConnection & Groundsignal satisfy reconnect	RECONNECT	[5]
58-D	Once ground signal has been reacquired, Astrobee resumes downlinking the live video stream to the Control Station.	Whenever ISSConnection & Groundsignal & !Stream Astrobee shall until Stream satisfy downlink	TRANSMIT	[5]
59-A	Astrobee is programmed to stop when it detects an obstacle	Upon ObstacleDetected Astrobee shall immediately satisfy Stop	TRIGGERED INSTANT REACTION	[5]
59-B	we are considering using Astrobee's lights and/or speaker to signal when it enters a hatchway	Whenever EntersHatchway Astrobee shall immediately satisfy EntranceAlarm	INSTANT REACTION	[5]
60-A	White "Vid" LEDs indicate that cameras are in use	In VideoRecordingMode Astrobee shall always satisfy VidLED	MODAL MAINTAIN SAFE SPACE	[5]
60-B	A blue "Aud" light tells the crew that the microphone is on	In AudioRecording Astrobee shall always satisfy BlueAudLED	MODAL MAINTAIN SAFE SPACE	[5]
60-C	"Live" LEDs indicate that cameras are streaming	In StreamingMode Astrobee shall always satisfy LiveLED	MODAL MAINTAIN SAFE SPACE	[5]
61-A	The Control Station warns operators when they create plans that translate through a KOZ.	Whenever KOZPlan ControlStation shall at the next timepoint satisfy Warn	PROMPT REACTION	[5]
61-B	The Control Station prevents operators from sending plans that translate through a KOZ to Astrobee until the violating segments are modified	ControlStation shall until !KOZPlan satisfy !SendPlan + ControlStation shall eventually satisfy !KOZPlan	WAIT	[5]
62	As a final safeguard, Astrobee itself has an internal list of KOZs that it checks before moving	Whenever moving Astrobee shall immediately satisfy !KOZ1 & !KOZ2	KEEP-OUT-ZONE	[5]

63-A	The robot can periodically update multi-sensor 3D maps of the vehicle. air quality tracking can all help flight controllers understand system status	Whenever TimeForAir Astrobee shall eventually satisfy AirSurvey	DELAYED REACTION	[5]
63-B	The robot can periodically update multi-sensor 3D maps of the vehicle. RFID quality tracking can all help flight controllers understand system status	Whenever TimeForRFID Astrobee shall eventually satisfy RFIDSurvey	DELAYED REACTION	[5]
63-C	The robot can periodically update multi-sensor 3D maps of the vehicle. Visual Imaging tracking can all help flight controllers understand system status	Whenever TimeForVisualImaging Astrobee shall eventually satisfy VisualImagingSurvey	DELAYED REACTION	[5]
63-D	The robot can periodically update multi-sensor 3D maps of the vehicle. Thermal imaging tracking can all help flight controllers understand system status	Whenever TimeForThermalImaging Astrobee shall eventually satisfy ThermalImagingSurvey	DELAYED REACTION	[5]
64	Automated change detection and trending. Once a baseline sensor map is available, changes at the next update can indicate developing problems at an early stage	Whenever SurveyDone Astrobee shall eventually satisfy CompareMaps	DELAYED REACTION	[5]
65	Localizing problems. For example, if a leak produces a whistling sound, acoustic or ultrasonic sensors onboard the robot can be used to pinpoint its location.	Whenever AnomalyDetected Astrobee shall eventually satisfy PinpointProblem	DELAYED REACTION	[5]
66	When flight controllers have a question about something, they can use the robot to get an updated view, filling a role currently played by crew on ISS	Whenever SpotCheck Astrobee shall eventually satisfy UpdateMap	DELAYED REACTION	[5]
67	The first is the observing and planning phase for acquiring motion information of the target satellite and planning when and where the robot will grasp the target satellite	Upon FirstPhase ServicingSatellite shall eventually satisfy AcquireMotionInformation & Planning	PHASES	[12]

68	The second phase is to control the robot to move toward the planned grasping location to make the robot ready for the capturing of the target.	Upon AcquireMotionInformation & Planning ServicingSatellite shall at the next timepoint satisfy SecondPhase + Upon SecondPhase ServicingSatellite shall eventually satisfy MoveToPosition	PHASES	[12]
69	The third phase is the capture (physical interception) phase in which the manipulator physically captures the target satellite	Upon MoveToPosition ServicingSatellite shall at the next timepoint satisfy ThirdPhase + Upon ThirdPhase RobotManipulator shall eventually satisfy PhysicalCapture	PHASES	[12]
70	The fourth phase is the post-capture phase in which captured target satellite is stabilized along with the servicing system	Upon PhysicalCapture ServicingSatellite shall at the next timepoint satisfy FourthPhase + Upon FourthPhase ServicingSatellite shall eventually satisfy Stabilization	PHASES	[12]
71	The maximum rotation speed is restricted within 23,100 rpm to ensure the crew's safety	IntBall2 shall always satisfy RPM <= 23100	MAINTAIN SAFE SPACE	[17]
72	Once the variances of the derivatives of acceleration and angular velocity from the IMU exceed predefined upper thresholds, the status shifts to collision mode.	If (VelocityVariancesc >UpperVelocityThreshold) & (AccelerationVariances >UpperAccelerationThreshold) IntBall2 shall immediately satisfy CollisionMode	TRIGGERED INSTANT REACTION	[17]
73	If the variances immediately decrease below lower thresholds, the impact cause is presumed to be an impulsive external force, and the Int-Ball2 tries to maintain its current pose	In CollisionMode if (VelocityVariances <LowerVelocityThreshold) & (AccelerationVariances <LowerAccelerationThreshold) IntBall2 shall immediately satisfy MaintainCurrentPose	MODAL TRIGGERED INSTANT REACTION	[17]

74	Otherwise, it is assumed to be held by the astronaut's hands, and maneuver control is turned off	In CollisonMode if !(VelocityVariances <LowerVelocityThreshold) | !(AccelerationVariances <LowerAccelerationThreshold) IntBall2 shall immediately satisfy ManeuverControl=0 & AstronautControl	MODAL TRIGGERED INSTANT REACTION	[17]
75	After the astronaut releases the robot, the control for maintaining the pose at the released point is restarted if the variance falls below the lower threshold	If AstronautControl=0 & VarianceThreshold <VarianceThreshold IntBall2 shall immediately satisfy MaintainCurrentPose	TRIGGERED INSTANT REACTION	[17]
76	Additionally, when the navigation camera is blocked by crew interference or positioned too close to a wall so that the feature points for vSLAM cannot be detected, the navigation subsystem shifts to inertial navigation that uses the IMU without relying on the vSLAM output.	Whenever vSLAMUnavailable IntBall2 shall at the next timepoint satisfy NavigatewithIMU & NavigatevSLAM=0	PROMPT REACTION	[17]
77-A	However, if the vSLAM output remains unavailable for an extended period, the robot rotates in place until the feature points detected in the current view align with those in the stored map	whenever vSLAMUnavailable IntBall2 shall until FeaturePointDetected satisfy RotateProtocol + whenever vSLAMUnavailable IntBall2 shall eventually satisfy FeaturePointDetected	CONDITIONAL WAIT	[17]
77-B	if the vSLAM output remains unavailable for an extended period	Whenever vSLAMOutput=0 & TimePassed <= ExtendedPeriod IntBall2 shall at the next timepoint satisfy vSLAMUnavailable	PROMPT REACTION	[17]
78	Furthermore, when the Int-Ball2 automatically detects that the remaining battery power is low, it returns to the DS for recharging	Whenever IntBall2Power <= SafeBattery IntBall2 shall at the next timepoint satisfy RechargeMode	PROMPT REACTION	[17]