Elucidation and Analysis of Specification Patterns in Aerospace System Telemetry *

Zachary Luppen[1 [0000−0003−0704−843X]], Michael Jacks[2 [0000−0002−1680−4535]], Nathan Baughman[2 [0000−0003−1217−2267]], Muhamed Stilic[2 [0000−0002−0184−9925]], Ryan Nasers[2 [0000−0003−2114−062X]], Benjamin Hertz[3 [0000−0003−1627−9715]], James Cutler[4 [0000−0002−6984−6851]], Dae-Young Lee[2 [0000−0002−6083−1805]], and Kristin Yvonne Rozier[2 [0000−0002−6718−2828]]

[1] Space Exploration Technologies Corp., Hawthorne, CA 90250
[2] Iowa State University, Ames IA 50010
[3] Collins Aerospace, Cedar Rapids IA 52498
[4] University of Michigan, Ann Arbor MI 48109

Abstract. Experimental aerospace projects often require flight vehicle platforms for testing, such as high-altitude balloons, sounding rockets, unmanned aerial systems (UAS), and CubeSats. The system telemetry transmitted by these vehicles is crucial to understanding overall performance. A growing desire to implement greater levels of system autonomy and AI-enhanced control into these systems merits introducing rigorous safety analysis from formal methods techniques, such as Runtime Verification (RV). RV depends heavily upon the accuracy and robustness of the specifications it reasons over, and the task of developing a comprehensive set of system specifications often poses a significant challenge. To aid specification development for new systems, we provide an analysis on the process of implementing RV into four real aerospace systems with increasing complexity. We design and validate fourteen formal specifications for a real high-altitude balloon mission and draw on three past formal specification efforts on a sounding rocket, UAS Traffic Management (UTM) system, and CubeSat to compare specification patterns and overlapping system needs. We identify four common temporal logic subformulas for specifications within and between these systems, providing metrics on development resources, frequency, and perceived automation difficulty. We generalize our results and discuss considerations for automatically generating formal specifications in aerospace projects.

Keywords: Runtime Verification · Temporal Logic · System Health Monitoring · Formal Specification · R2U2 · High-Altitude Balloon · Sounding Rocket · UAS · CubeSat · Satellite.

* This project/material is based upon work supported by the Iowa Space Grant Consortium under NASA Award No. 80NSSC20M0107. Work partially supported by NSF CAREER Award CNS-1552934, NASA ECF NNX16AR57G, and NSF PFI:BIC grant CNS-1257011. Thanks to Kaili Henry and Yang He for their work on specification development and Matthew Nelson for providing resources from HABET. Reproducibility artifacts are available at `http://temporallogic.org/research/AerospaceSystems-NFM22`.

1 Introduction

Academic, industrial, commercial, and amateur entities profoundly use small aerospace systems to perform small-scale, experimental research [24,30,45,46]. The intended experiments and mission goals for these projects vary greatly, from testing experimental hardware to evaluating cosmic radiation levels at progressive altitudes in Earth's atmosphere [34]. The process of building, designing, and flying these systems is a non-trivial task despite their benefits of low cost and fast turnaround times. In practice, many developers meet unforeseen challenges and setbacks that can occur at practically any stage of a given mission [3, 26, 47]. Conceivable problems are documented and well-known, but developers generally have the means only to develop a baseline working system due to limited resources, engendering a need for greater system autonomy.

Enabling small aerospace systems to monitor system faults automatically and in real time provides the ability to trigger mitigation actions and optimize performance. Runtime verification (RV) specializes in identifying fault signatures and provides a deeper understanding of a given system's behavior [35, 36]. Recent studies have explored the integration of RV into autonomous aerospace systems, like sounding rockets [14], unmanned aerial systems (UAS) [6, 13, 29], and CubeSats [2, 12, 19, 33]. However, there have been few efforts to understand the similarities and differences, along with scaled complexity, in applying formal specification and RV to these systems [40]. It is crucial to understand how mission needs compare within and across each system to elicit formal specifications automatically for real-time verification of future designs.

We examine four real aerospace systems designed, developed, and flown/launched independently: a high-altitude balloon, a sounding rocket, a UAS Traffic Management (UTM) system, and a CubeSat. We contribute (1) formal high-altitude balloon specifications and successful RV on the real dataset using the R2U2 RV engine [41], (2) a comparative analysis and identification of patterns in aerospace system specifications, and (3) a map for auto-generating formal specifications. The remainder of this paper is organized as follows. In Section 2, we discuss the syntax and formal semantics of Mission-time Linear Temporal Logic (MLTL), the common specification language of these studies. Section 3 briefly outlines each of the four aerospace vehicles and their mission profiles and discusses their respective telemetry data collections. Section 4 describes development and validation of formal specifications for a high-altitude balloon system and scaling to larger systems. We provide metrics and comparisons of formal specifications and patterns identified in all four aerospace systems in Section 5. In Section 6, we discuss lessons learned and conclude with plans for developing more automated techniques to generate formal specifications.

2 MLTL Syntax and Semantics

We utilize mission-time linear temporal logic (MLTL) for all specifications developed for the aerospace systems [17, 38]. MLTL employs closed interval time bounds over a set of bounded natural numbers on all temporal operators, rather than literal time increments.

Definition 1. *(MLTL Syntax) The syntax of a given MLTL formula ϕ comprised of atomic propositions $\mathcal{AP}$ is recursively defined as such:*

$$\phi ::= true \mid false \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \Box_I \phi \mid \Diamond_I \phi \mid \phi_1 \mathcal{U}_I \phi_2 \mid \phi_1 \mathcal{R}_I \phi_2$$

where p $\in \mathcal{AP}$ is a Boolean and all ϕ are atomic propositions. The symbols utilized in this syntax stand for the following: $\neg$ is not, $\wedge$ is logical and, $\vee$ is logical or, $\Box_I$ is globally, $\Diamond_I$ is eventually, $\mathcal{U}_I$ is until, $\mathcal{R}_I$ is release. I is an interval [lb,ub] from a lower to an upper bound, where lb $\leq$, ub, and lb, ub $\in \mathbb{N}$ [17, 38].

As MLTL is derived from linear temporal logic (LTL), a majority of the semantics are identical: `false` $\equiv$ `true`, $\phi_1 \vee \phi_2 \equiv \neg(\neg\phi_1 \wedge \neg\phi_2)$, $\neg(\phi_1 \mathcal{U}_\mathcal{I} \phi_2) \equiv (\neg\phi_1 \mathcal{R}_\mathcal{I} \neg\phi_2)$ and $\neg\Diamond_I \phi \equiv \Box_I \neg\phi$. Note that unlike LTL, MLTL does not possess the next $\mathcal{X}$ operator because it is logically equivalent to $\Box_{1,1}\phi$ [17].

Definition 2. *(MLTL Semantics) A MLTL formula ϕ, over a set of propositions $\mathcal{AP}$, by a computation/trace π starting from position i (denoted as π, i $\models \phi$) has satisfaction recursively defined as follows:*

- π, i $\models$ true
- π, i $\models$ p iff p $\in \pi$[i]
- π, i $\models \neg\phi$ iff π, i $\not\models \phi$
- π, i $\models \phi_1 \wedge \phi_2$ iff π, i $\models \phi_2$
- π, i $\models phi_1 \, \mathcal{U}_{lb,ub} \phi_2$ iff $|\pi| \geq$ i + lb and, there exists j $\in$ [i + lb, i + ub] such that π, j $\models \phi_2$ and for every k ¡ j, k $\in$ [i + lb, i + ub], π, k $\models \phi_1$.

3 Aerospace Systems

We analyze and compare formal specifications for four separate aerospace systems: a high-altitude balloon, a sounding rocket, a UTM, and a CubeSat. This section briefly describes each system and the exigence for applying formal methodologies. We note that, while each system here is progressively more complex than the last, this is not indicative of aerospace systems as a whole. Additional information and visuals of the telemetry datasets are available at `http://temporallogic.org/research/AerospaceSystems-NFM22`.

3.1 High-Altitude Balloon

The Make 2 Innovate (M:2:I) Laboratory, located at Iowa State University (ISU), developed a high altitude balloon as part of its High Altitude Balloon Experiments in Technology (HABET) series [20]. The goal of this program is to design, build, fly, and recover small payloads developed entirely by undergraduate students at ISU [20, 21]. This launch tested the functionality and accuracy of a NEO-M9N GPS module developed by SparkFun for use on future HABET projects. While the GPS module proved capable, a handful of extraneous measurements were made that provided ground station operators with inaccurate readings while the balloon remained almost stationary on the ground. While not utilized during this launch, some balloon launch teams include onboard mechanisms to pop, vent, or detach the balloon at a defined altitude (often either for experimentation or to prevent further drift in upper atmospheric winds) measured by the GPS [4, 10, 23, 25, 32]. If such a mechanism had been used on this project, extraneous measurements may have led to prematurely popping the balloon and thereby ending the mission. Studies of this system do not exist in current literature.

3.2 Sounding Rocket

The Cyclone Rocketry team at ISU developed a sounding rocket called *Nova Somnium*. Nova Somnium flew at the 2019 Spaceport America Cup near Las Cruces, New Mexico [8, 14]. Originally designed to reach an apogee altitude of 10,000 ft AGL, it carried a telemetry system for data transmission back to a dedicated ground station, and an aerobraking control system (ACS). During the launch, the ACS actuated prematurely and resulted in a critical failure of the system. Development of formal specifications and RV analysis of this system is described in greater detail in [14].

3.3 UAS Traffic Management System (UTM)

The UAS considered in the UTM project is an AeroVironment (formerly produced by Pulse Aerospace) VAPOR 55 and is owned and operated by the University of Iowa's Operator Performance Laboratory (OPL) in Iowa City, IA [1, 6]. UAS are quickly integrating into the National Air Space (NAS) over the United States, and the Federal Aviation Administration (FAA) expects their use to "expand rapidly" in the coming years. Given that this increased air traffic will likely produce congestion and safety concerns, there is a growing need to integrate UAS with intelligent, automated systems for UAS Traffic Management. Previous studies mapped out RV implementations for three separate aspects of the UTM framework: onboard the VAPOR 55, each ground control system, and within the UTM cloud-based framework [6, 13].

3.4 CubeSat

The CubeSat, called the GEO-CAPE ROIC In-Flight Performance Experiment (GRIFEX), was developed by the Michigan eXploration Lab (MXL) [16]. Launched in December 2015 from Vandenberg Air Force Base in California, GRIFEX is a 3U CubeSat carrying a NASA Jet Propulsion Laboratory (JPL)-developed all-digital in-pixel high frame rate read-out integrated circuit (ROIC) being tested for use on future spacecraft [5, 16, 27, 31, 37]. Although CubeSats generally have mission lifetimes between 6 months to two years, GRIFEX has been in operation for over five years. Such an abnormally long lifetime provides developers with unique data regarding performance degradation due to solar and cosmic radiation. The majority of GRIFEX operations involve manual processes and was not subject to rigorous formal methodologies during its development phase [16]. Recent efforts characterized the CubeSat's performance degradation to provide further insight on methods for updating formal specifications for a dynamically changing system [18].

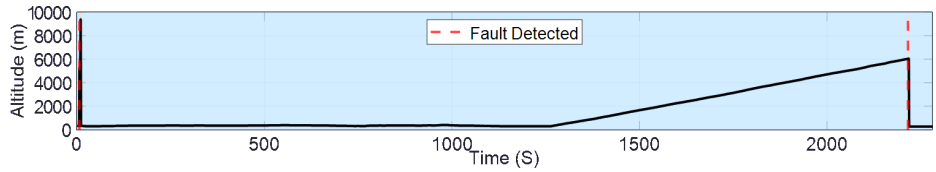
4 Methodology

The four aerospace systems described above are reactive to their environments and possess well-defined operational timelines; this merits a specification logic like Linear Temporal Logic (LTL) to provide finite-bounded reasoning for each system [22]. Mission-time Linear Temporal Logic (MLTL) encodes the system requirements generically with integer-bounded time steps that provide ease of mapping to real mission data,

providing optional integer bounds on temporal operators [17, 38]. MLTL has been used in many industry-based research projects [2, 6, 7, 11, 13, 14, 15, 18, 28, 38, 39, 42, 43, 44].

Past studies utilized a similar methodology to develop runtime specifications for aerospace systems [6, 13, 14, 18]. We employ these techniques in developing specifications for the high-altitude balloon, constructing requirements in English from known mission parameters and tracking system coverage to capture as many system constraints as possible. We organize our specifications into the categories defined in [40]: operating ranges (RNG), rates of change (RAT), relationships (REL), control sequences (CTRL), and consistency checks (CHE). Our specification validation and debugging uses previously described methods [14].

(a) Balloon Altitude



(b) (| `Altitude[i] - Altitude[i-1]` | < `Altitude_Delta_Max`)

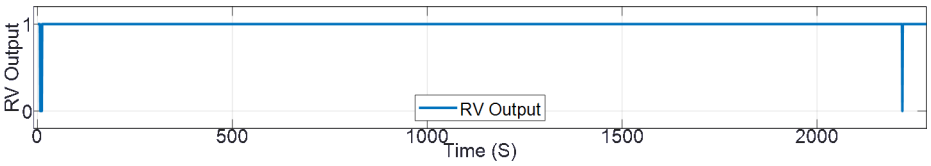
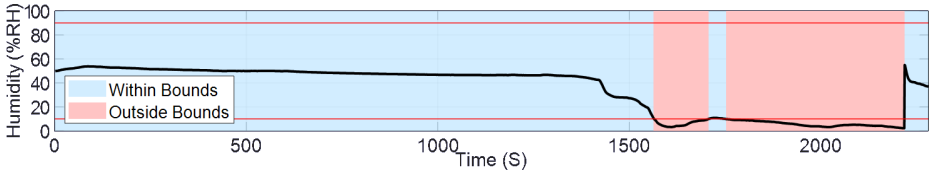


Fig. 1: R2U2 RV engine [41] monitoring for specification RAT3 of the HABET high-altitude balloon. (a) The high altitude balloon's measured altitude throughout the flight. Half of the duration shown here is when the balloon was secured to the ground for final checkouts. (b) RV output from the R2U2 tool, correctly identifying two faults when the change in altitude between time steps exceeds more than 20 m. The GPS status for the first fault was reported as high-integrity but is clearly an incorrect measurement. The second fault occurred during balloon burst and descent.

To demonstrate the validity of this approach, we examine two specifications and the resulting RV output, produced using R2U2 [41], from the HABET telemetry data obtained during the balloon's mission. Figure 1 shows multiple instances of off-nominal altitude delta measurements. The altitude delta spike seen at the beginning of the time series data occurred while the balloon was affixed to the ground and risked the balloon's systems popping itself, thinking apogee had been reached. The out-of-bounds measurements of the balloon's humidity sensor appear in Figure 2.

With runtime specification development efforts performed for each of the four aerospace systems, comparing these separate processes is now possible. We primarily want to understand how each system's complexity affects the efforts needed to develop formal specification sets and highlight four critical metrics: (1) specification number and (2) type, (3) estimated development time, and a (4) measure of the perceived level of difficulty in generating a specification. This latter metric, which we refer to as the Au-

(a) Balloon Relative Humidity Measurement



(b) ((`Relative Humidity >= RH_Minimum`) & (`Relative Humidity <= RH_Maximum`))

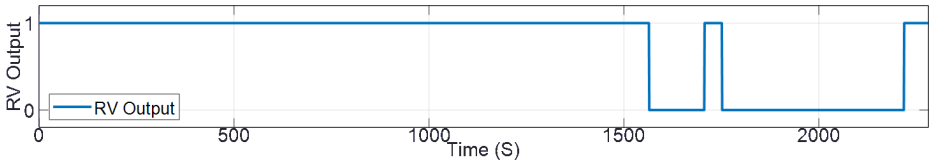


Fig. 2: R2U2 monitoring for specification RAT3 of the HABET high-altitude balloon. (a) The high altitude balloon's measured relative humidity throughout the flight. (b) RV output from the R2U2 tool, correctly identifying two prolonged durations when the humidity data is not considered accurate per the sensor's datasheet bounds.

tomation Level (AL) provides a general subjective understanding of how easily a specification is elicited and consists of three separate rankings. These ranks are a simplified version of the automation levels defined in past literature [9]. Specifications ranked **AL 1** require a brief examination of a datasheet to generate and are comparatively easy to extract and formalize. Additionally, most specifications with this ranking are dependent only on component data sheets (rather than mission-specific parameters) and are therefore easily applicable to other systems. One example of this rank would be a specification that dictates a temperature sensor must operate within its operating minimum and maximum; this information is readily available in the component's datasheet. Specifications with rank **AL 2** require a small degree of user/developer input. These specifications require a minor parameter or range adjustment but are still reasonably easy to generate. An example of this rank would be a specification describing the minimum and/or maximum altitudes that an aerospace system should experience; these requirements vary by system. Lastly, specifications with rank **AL 3** require a significant level of user interaction. These specifications are the most difficult to produce. Generally, specifications describing control sequences or mission phase durations fall under this category. Table 1 displays the key metrics of the formal specifications developed for each of the four aerospace systems.

5 Results

We identify four separate generalized temporal logic patterns from the analysis of the aerospace system specifications. The atomic propositions in each pattern can take on varying levels of complexity, ranging from a simple equality check to propositional logic formulas containing multiple sub propositions (e.g., $a0 : !a1 \rightarrow (a2 \,||\, a3)$, where

Table 1: Specification development summary for the high-altitude balloon, sounding rocket, UTM, and CubeSat examined in this study. Development time estimates account for time spent debugging and validating. The Automation Level (AL) metric provides a measure 1-3 of the difficulty in eliciting specifications for each pattern.

Aerospace System	MLTL Spec Category	Count	Estimated Development Time	AL 1	AL 2	AL 3
Balloon	**All specifications**	**14**	**13 person-hours**	**6**	**2**	**8**
	RNG Specifications	7	4 person-hours	6	0	1
	RAT Specifications	6	8 person-hours	0	6	0
	REL Specifications	1	0 person-hours	0	0	0
	CTRL Specifications	0	0 person-hours	0	0	0
	CHE Specifications	0	1 person-hours	0	0	1
Sounding Rocket	**All specifications**	**19**	**50 person-hours**	**4**	**7**	**8**
	RNG Specifications	6	14 person-hours	2	3	1
	RAT Specifications	6	15 person-hours	2	4	0
	REL Specifications	0	0 person-hours	0	0	0
	CTRL Specifications	7	21 person-hours	0	0	7
	CHE Specifications	0	0 person-hours	0	0	0
UTM	**All specifications**	**124**	**69 person-hours**	**30**	**87**	**7**
	RNG Specifications	80	12 person-hours	27	53	0
	RAT Specifications	18	18 person-hours	3	15	0
	REL Specifications	18	18 person-hours	2	9	7
	CTRL Specifications	8	21 person-hours	2	6	0
	CHE Specifications	0	0 person-hours	0	0	0
Cube Satellite	**All specifications**	**265**	**77 person-hours**	**180**	**25**	**60**
	RNG Specifications	149	39 person-hours	68	25	56
	RAT Specifications	112	32 person-hours	112	0	0
	REL Specifications	4	6 person-hours	0	0	4
	CTRL Specifications	0	0 person-hours	0	0	0
	CHE Specifications	0	0 person-hours	0	0	0

$a1, a2, a3$ are defined from system variable comparisons). Table 2 details metrics on each of these patterns and their occurrences within the aerospace system specifications.

The first specification pattern, written as $G[0, M](a0)$, appears the most frequently in all four subsystems. The M bound specifies that the specification should hold true for every time step of the entire mission. Specifications with this pattern are well suited to enforcing operating ranges, bounding rates of change, specifying relationships between variables, and checking for logical inconsistencies. Most $G[0, M](a0)$ specifications are AL 1.

The second pattern, $G[0, M]F[0, N](a0)$, states that $a0$ must be true at least once every N time steps, where $N < M$. This specification pattern is identical to $G[0, M](a0)$ when $N = 0$, but when $N > 0$ allows $a0$ to be periodically violated without violating the specification. This provides the flexibility to handle anticipated stochastic or cyclic behavior. These specifications usually fall into AL 2 or 3 due to the value of N typically needing human definition.

The third pattern, $G[0, M](a0 \rightarrow F[0, N]a1)$, represents a temporal relationship between a condition ($a0$) and a behavior ($a1$). Specifications with this pattern primarily

encode control sequences. The sounding rocket ACS and UTM system provide air and spacecraft control, so these specification sets benefit most from this pattern. Most of these specifications are AL 3 due to the need for higher knowledge of mission event sequencing and coordination with other systems.

The final pattern we identify, $G[0, M](a0 \rightarrow a0\ U[0, N]a1)$, places a temporal constraint $a1$ on $a0$. Whenever the $a0$ condition is met, $a0$ must continue to hold until $a1$ occurs, and $a1$ must occur within N time steps of $a0$ first being met. Like the previous pattern, this pattern primarily encodes control sequences and typically falls into AL 3.

Table 2: Metrics on the generalized specification forms described in Section 5. Under "Aerospace System" we tabulate the number of specifications of each pattern for each aerospace system case study.

		Aerospace System			
Specification Pattern	**Typical AL**	**Balloon**	**Sounding Rocket**	**UTM**	**CubeSat**
$G[0, M](a0)$	1	8	8	77	265
$G[0, M]F[0, N](a0)$	2	6	4	35	0
$G[0, M](a0 \rightarrow F[0, N]a1)$	3	0	5	6	0
$G[0, M](a0 \rightarrow a0U[0, N]a1)$	3	0	2	0	0

6 Conclusion

The four common temporal logic subformulas we identify in the aerospace systems are quickly and easily realizable as formal specifications. However, such aerospace systems generally do not see the level of formal reasoning and validation that larger projects do, owing to time and resource constraints. To fully bridge this gap, we suggest a tool that will, given a list of parts/components on a designed mission, reference a database and return a set of temporal logic specifications in the forms of the identified patterns; parts databases already exist for common components and sensors. Development of this tool would be non-trivial but would significantly aid formal specification development for system developers who are not familiar with formal methods.

References

1. AeroVironment, I.: Vapor uas: Helicopter drone with drop delivery (2021), https://www.avinc.com/uas/vapor
2. Aurandt, A., Jones, P., Rozier, K.Y.: Runtime Verification Triggers Real-time, Autonomous Fault Recovery on the CySat-I. In: Proceedings of the 14th NASA Formal Methods Symposium (NFM 2022). Springer, Caltech, California, USA (May 2022)
3. Balloonnews, Balloonnews: 10 ways that a high altitude balloon flight can go wrong (Aug 2014), https://balloonnews.wordpress.com/2014/04/10/10-ways-that-a-high-altitude-balloon-flight-can-go-wrong/
4. Basta, T., Miller, S., Clark, R.T.: Weather Balloon Altitude Control System. Montana State University (2014-2015)
5. Bekker, D.L., Bryk, M.B., Delucca, J.M., Franklin, B.R., Hancock, B., Klesh, A.T., Meehan, C., Meshkaty, N., Nichols, J.E., Pingree, P.J., Rider, D.M., Werne, T.A., Wu, J.: Grifex payload data system architecture for on-orbit focal plane array evaluation. In: Proceedings of the American Geophysical Union, Fall Meeting 2012 (2012)
6. Cauwels, M., Hammer, A., Hertz, B., Jones, P., Rozier, K.Y.: Integrating Runtime Verification into an Automated UAS Traffic Management System, pp. 340–357. Springer, Cham (09 2020). https://doi.org/10.1007/978-3-030-59155-7_26
7. Dabney, J.B., Badger, J.M., Rajagopal, P.: Adding a verification view for an autonomous real-time system architecture. In: AIAA Scitech 2021 Forum. p. 0566 (January 2021). https://doi.org/https://doi.org/10.2514/6.2021-0566
8. ESRA Board of Directors: 2019 spaceport america cup (2019), http://www.soundingrocket.org/2019-sa-cup.html
9. Fisher, M., Mascardi, V., Rozier, K.Y., Schlingloff, B.H., Winikoff, M., Yorke-Smith, N.: Towards a framework for certification of reliable autonomous systems. Autonomous Agents and Multi-Agent Systems **35** (04 2021). https://doi.org/10.1007/s10458-020-09487-2
10. Garg, K.: Autonomous Navigation System for High Altitude Balloons. Ph.D. thesis, Luleå Technical University, Graphic Production 2019 (2019)
11. Geist, J., Rozier, K.Y., Schumann, J.: Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems. In: Proceedings of the 14th International Conference on Runtime Verification (RV14). vol. 8734, pp. 215–230. Springer-Verlag (September 2014)
12. Gross, K.H., Clark, M., Hoffman, J.A., Fifarek, A., Rattan, K., Swenson, E., Whalen, M., Wagner, L.: Formally Verified Run Time Assurance Architecture of a 6U CubeSat Attitude Control System, pp. 1–15. AIAA Infotech (2020). https://doi.org/10.2514/6.2016-0222, https://arc.aiaa.org/doi/abs/10.2514/6.2016-0222
13. Hammer, A., Cauwels, M., Hertz, B., Jones, P., Rozier, K.Y.: Integrating runtime verification into an automated uas traffic management system. Innovations in Systems and Software Engineering: A NASA Journal (July 2021). https://doi.org/10.1007/s11334-021-00407-5
14. Hertz, B., Luppen, Z., Rozier, K.Y.: Integrating runtime verification into a sounding rocket control system. In: Dutle, A., Moscato, M.M., Titolo, L., Muñoz, C.A., Perez, I. (eds.) NASA Formal Methods. pp. 151–159. Springer International Publishing, Cham (2021)
15. Kempa, B., Zhang, P., Jones, P.H., Zambreno, J., Rozier, K.Y.: Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2. In: Proceedings of the 18th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS). Lecture Notes in Computer Science (LNCS), vol. TBD, p. TBD. Springer, Vienna, Austria (September 2020). https://doi.org/TBD, http://research.temporallogic.org/papers/KZJZR20.pdf

16. eXploration Lab, T.M.: Grifex (2021), https://exploration.engin.umich.edu/blog/?page_id=2684
17. Li, J., Vardi, M.Y., Rozier, K.Y.: Satisfiability checking for Mission-time LTL. In: Proceedings of 31st International Conference on Computer Aided Verification (CAV). LNCS, vol. 11562, pp. 3–22. Springer, New York, NY, USA (July 2019). https://doi.org/https://doi.org/10.1007/978-3-030-25543-5_1
18. Luppen, Z., Jacks, M., Baughman, N., Stilic, M., Nasers, R., Lee, D.Y., Rozier, K.Y., Cutler, J.: Runtime verification of the dynamic performance degradation of the grifex cubesat (under review). In: NASA Formal Methods. Springer International Publishing (2022)
19. Luppen, Z.A., Lee, D.Y., Rozier, K.Y.: A case study in formal specification and runtime verification of a CubeSat communications system. In: AIAA Scitech 2021 Forum. American Institute of Aeronautics and Astronautics (Jan 2021). https://doi.org/10.2514/6.2021-0997, https://doi.org/10.2514/6.2021-0997
20. M2I: Make to innovate (m:2:i) (2021), https://m2i.aere.iastate.edu/
21. M2I: Project goals (habet) (2021), https://m2i.aere.iastate.edu/habet/project-goals-and-scope-of-work/
22. Manna, Z., Pnueli, A.: Temporal Verification of Reactive Systems: Safety. Springer New York (2012), https://books.google.com/books?id=lfIGCAAAQBAJ
23. Marshall, R.: Cutdown mechanisms (Mar 2021), https://sites.google.com/site/ki4mcw/Home/cutdown-mechanisms
24. Merkert, R., Bushell, J.: Managing the drone revolution: A systematic literature review into the current use of airborne drones and future strategic directions for their effective control. Journal of Air Transport Management **89**, 101929 (Oct 2020). https://doi.org/10.1016/j.jairtraman.2020.101929, https://doi.org/10.1016/j.jairtraman.2020.101929
25. Meyer, J.J., Flaten, J.A., Candler, G.V.: Pdf (Apr 2021)
26. Mike Tolmasoff, Renelito Delos Santos, and Catherine Venturini: Improving mission success of cubesats. In: Proceedings of the U.S. Space Program Mission Assurance Improvement Workshop (May 2007)
27. Moldwin, M., Sharma, S., Deshmukh, A., Scott, C., Cutler, J.: Machine learning algorithms for spacecraft magnetic field interference cancellation: Enabling satellite magnetometry without a boom. Earth and Space Science Open Archive p. 1 (2019). https://doi.org/10.1002/essoar.10500304.1, https://www.essoar.org/doi/abs/10.1002/essoar.10500304.1
28. Moosbrugger, P., Rozier, K.Y., Schumann, J.: R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems. Formal Methods in System Design pp. 1–31 (April 2017). https://doi.org/10.1007/s10703-017-0275-x
29. Munoz, C., Carreno, V., Dowek, G.: Formal Analysis of the Operational Concept for the Small Aircraft Transportation System, pp. 306–325. Springer Berlin Heidelberg, Berlin, Heidelberg (2006). https://doi.org/10.1007/11916246_16, https://doi.org/10.1007/11916246_16
30. NASA CubeSat Launch Initiative: CubeSat 101. California Polytechnic State University, San Luis Obispo (Cal Poly) CubeSat Systems Engineer Lab, 1 edn. (2017)
31. Norton, C.D., Pasciuto, M.P., Pingree, P., Chien, S., Rider, D.: Spaceborne flight validation of nasa esto technologies. In: 2012 IEEE International Geoscience and Remote Sensing Symposium. pp. 5650–5653 (2012). https://doi.org/10.1109/IGARSS.2012.6352330
32. Papp, D.: Archery release becomes reusable balloon cutdown mechanism (Mar 2021), https://hackaday.com/2021/03/27/archery-release-becomes-reusable-balloon-cutdown-mechanism/

33. Peng, Z., Lu, Y., Miller, A., Johnson, C., Zhao, T.: A probabilistic model checking approach to analysing reliability, availability, and maintainability of a single satellite system. In: 2013 European Modelling Symposium. pp. 611–616 (Nov 2013). https://doi.org/10.1109/EMS.2013.102
34. Phillips, T., Johnson, S., Koske-Phillips, A., White, M., Yarborough, A., Lamb, A., Herbst, A., Molina, F., Gilpin, J., Grah, O., Perez, G., Reid, C., Harvey, J., Schultz, J.: Space weather ballooning. Space Weather **14**(10), 697–703 (2016). https://doi.org/https://doi.org/10.1002/2016SW001410, https://agupubs.onlinelibrary.wiley.com/doi/abs/10.1002/2016SW001410
35. Pike, L., Goodloe, A., Morisset, R., Niller, S.: Copilot: A hard real-time runtime monitor. In: Proceedings of the 1st Intl. Conference on Runtime Verification. LNCS, Springer (November 2010), preprint available at https://leepike.github.io/pub_pages/rv2010.html
36. Pike, L., Goodloe, A., Morisset, R., Niller, S., Wegmann, N., Hathhorn, C., Mendelson, E., Laurent, J., Titolo, L., Jolayan, G.A., et al.: Copilot - realtime programming language and runtime verification framework (Mar 2022), https://copilot-language.github.io/
37. Pingree, P., Bekker, D., Bryk, M., DeLucca, J., Franklin, B., Hancock, B., Klesh, A., Meehan, C., Meshkaty, N., Nichols, J., Peay, C., Rider, D., Werne, T., Wu, Y.: Cove, marina, and the future of on-board processing (obp) platforms for cubesat science missions (12 2012)
38. Reinbacher, T., Rozier, K.Y., Schumann, J.: Temporal-logic based runtime observer pairs for system health management of real-time systems. In: Proceedings of the 20th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS). Lecture Notes in Computer Science (LNCS), vol. 8413, pp. 357–372. Springer-Verlag (April 2014)
39. Rozier, K.Y., Schumann, J., Ippolito, C.: Intelligent Hardware-Enabled Sensor and Software Safety and Health Management for Autonomous UAS. Technical Memorandum NASA/TM-2015-218817, NASA, NASA Ames Research Center, Moffett Field, CA 94035, USA (May 2015)
40. Rozier, K.Y.: Specification: The biggest bottleneck in formal methods and autonomy. In: Proceedings of 8th Working Conference on Verified Software: Theories, Tools, and Experiments (VSTTE 2016). LNCS, vol. 9971, pp. 1–19. Springer-Verlag, Toronto, ON, Canada (July 2016). https://doi.org/10.1007/978-3-319-48869-1_2
41. Rozier, K.Y., Schumann, J.: R2U2: Tool Overview. In: Proceedings of International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CUBES). vol. 3, pp. 138–156. Kalpa Publications, Seattle, WA, USA (September 2017). https://doi.org/TBD, https://easychair.org/publications/paper/Vncw
42. Schumann, J., Moosbrugger, P., Rozier, K.Y.: R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems. In: Proceedings of the 15th International Conference on Runtime Verification (RV15). Springer-Verlag, Vienna, Austria (September 2015)
43. Schumann, J., Moosbrugger, P., Rozier, K.Y.: Runtime Analysis with R2U2: A Tool Exhibition Report. In: Proceedings of the 16th International Conference on Runtime Verification (RV16). Springer-Verlag, Madrid, Spain (September 2016)
44. Schumann, J., Rozier, K.Y., Reinbacher, T., Mengshoel, O.J., Mbaya, T., Ippolito, C.: Towards real-time, on-board, hardware-supported sensor and software health management for unmanned aerial systems. International Journal of Prognostics and Health Management (IJPHM) **6**(1), 1–27 (June 2015)
45. Science, H.A.: Intro to weather balloons (2021), https://www.highaltitudescience.com/pages/intro-to-weather-balloons

46. Seibert, G.: The history of sounding rockets and their contribution to European space research. ESA History Study Reports (11 2006)
47. Wong, K.: NASA's DEUCE-carrying rocket fails to collect data due to technical glitch (Nov 2017), https://www.aerospace-technology.com/news/newsnasas-deuce-carrying-rocket-fails-to-collect-data-due-to-technical-glitch-5962942