A Case Study in Formal Specification and Runtime Verification of a CubeSat Communications System

Zachary Allen Luppen* and Dae Young Lee† and Kristin Yvonne Rozier‡
Iowa State University, {zaluppen,kyrozier,daylee}@iastate.edu

CubeSats are garnering a lot of attention from several research communities due to their cheap price and easy accessibility to space. Although a CubeSat offers the potential to replace large and complex satellites for various space missions, developing a high-integrity platform that can fly in space requires robust software performance. The communications system of a CubeSat is critical to mission success. However, failed communications to the ground stations are commonplace for CubeSat projects. Thus, robustifying against these potentially mission-ending failures poses one of the leading challenges for CubeSat missions both now and in the future. Runtime verification (RV) specializes in real-time error detection in these types of temporal, reactive systems, and the R2U2 verification engine proved to fit in the CubeSat's resource constraints. We design specifications to detect and trigger appropriate mitigations for communications system faults. We discuss communications system specification debugging, validation, and variable coverage. Experimental evaluation on simulated orbital and telemetry datasets demonstrates that running R2U2 within a CubeSat communications system would detect several unique faults and trigger mitigations for these such as atypical position measurements or a voltage supply failure from occurring. We provide a roadmap from large CubeSat datasets to specification patterns for automatically capturing common monitoring properties. We outline our plans for integrating RV into other CubeSat systems as well as verifying an entire CubeSat telemetry dataset.

I. Introduction

Over the past decade, the use of CubeSats and other nanosatellites by academic, governmental, commercial, and amateur entities has grown drastically [1–3]. The relatively low cost and fast development times of these projects make a CubeSat an ideal platform for experimenting in space [1]. As of June 2020, over 1,300 CubeSats have been launched into space [4]. CubeSats may offer a scaled-down, simplified platform for developers, but they still experience many of the same and more problems that large satellites do [5]. Design flaws, poor workmanship, software bugs, low reliability of hardware (often consisting of commercial off-the-shelf (COTS) components), lack of adequate testing, and communications failures are among a few of the many problems CubeSat projects experience [5]. The low cost and short development times for these missions are appealing to space researchers; their use is only expected to rise in the coming years [1, 5], so it is important that we find ways to mitigate their faults, within CubeSat resource constraints.

The communications system of a CubeSat is vital to mission success. A CubeSat could be performing invaluable science, but a failure to communicate the results back to the ground renders the mission pointless. Communications system failures are currently one of the most common problems experienced by CubeSat developers [5, 6]. Enabling a CubeSat to monitor for these faults in real-time on-board would enable the CubeSat to either automatically trigger mitigation actions, or better utilize the satellite's limited bandwith to send specific fault information, rather than a large collection of unanalized data, to the ground. Runtime verification (RV) specializes in identifying temporal fault signatures characteristic of communication systems faults. If we can specify a temporal fault signature using the data available on-board, then an RV engine can monitor for that fault. The CubeSat's strict power and computational constraints present a major

*Ph.D. Student, Aerospace Engineering, Iowa State University, Ames, Iowa, 50011, Student Member.
†Assistant Professor, Aerospace Engineering, Iowa State University, Ames, Iowa, 50011, Member.
‡Assistant Professor, Aerospace Engineering, Iowa State University, Ames, Iowa, 50011, Associate Fellow.

impediment to embedding RV on-board. Any solution would need to operate with real-time guarantees, without dominating the system's resources, and provide the ability to reconfigure the monitors during system operation, without resynthesis, so that we can efficiently adapt to the dynamic operaitng environment after the CubeSat has launched. The Realizable, Responsive, Unobtrusive Unit (R2U2) tool is the only existing RV engine that satisfies all of our CubeSat system's constraints; it was specifically designed to provide flight-certifiable, resource-award RV that can be easily reconfigured post-deployment.

Recently, others have recognized the need to advance capabilities for CubeSat verification, which is now a pressing research topic. Analysis of these faults at design time via probabilstic model checking [7], and a model-driven trace-checking approach to elucidate on the most frequent requirement types in cyber-physical systems [8] provide promising platforms for producing more robust CubeSat designs. Recently, [9] formally verified a CubeSat's attitude, determination, and control system (ADCS) at design time and used this to cross-check the runtime behaviors of an unverified version. Now we must face the challenge of bridging the gap to on-board runtime verification, so that we can detect and mitigate any unexpected faults, e.g., due to the dynamic environment, that these previously-explored verification efforts could not anticipate at design time.

We examine a CubeSat communication system model and overview its design and implementation. The model consists of two separate systems, one consisting of orbital data (similar to information acquired by a ground station system) and the other comprised of telemetry information for a communications system module (simplified telemetry data with information such as voltages and temperatures). Using methods described in [10] and expanded upon in [11], we develop, revise and validate formal specifications for these two sub-systems and detail patterns observed during the process. We encode these specifications as runtime observers using R2U2 and validate them against datasets generated by the model to ensure the absense of fault-positive fault alerts. To further validate their accuracy, we inject realistic faults – such as unreasonable elevation and voltage measurements – into the datasets at known times and check that R2U2 immediately detects these faults without false negatives, which NASA identifies as one of the most prominent issues in developing intelligent, autonomous systems [12]. This case study provides a roadmap for developing a more robust CubeSat communications system through formal specification and details the process for integrating runtime verification (RV) into such a system to provide autonomous verification of its operation during operation.

Our contributions are as follows:

1) We develop a reference set of formal specifications in mission-time linear temporal logic (MLTL) describing a modeled CubeSat communications system
2) We detail our validation strategy over these specifications using experimental evaluation with the R2U2 tool
3) We highlight and discuss specification patterns that emerge while developing and revising the RV specifications
4) We discuss lessons learned from validating these specifications that may inform future CubeSat runtime verification efforts.

This paper is organized as follows. Section III provides an overview of our communications system model, dividing. Section IV contains our formal specifications, relating to information sourced from a CubeSat telemetry dataset as well as a ground station. For future developers, we elucidate on the organization of the specifications, the need for temporal operators, and note specific patterns we encountered during their creation. Section V describes our test scenarios, output graphs from R2U2 for a handful of our specifications. In section VI, we describe variable coverage of the specifications, additional tools used by developers with communications systems, and discuss the limitations of the communications system model. Section VII concludes with lessons learned during the specification development and testing process and discusses future steps for RV integration into CubeSat systems and telemetry analysis.

II. Preliminaries

A. CubeSats

Small satellites are divided into classes based on total mass, such as nanosatellites, picosatellites, and femtosatellites. These low-cost, small spacecraft platforms have cheapened access to space experimentation [13].

The term "CubeSat" used throughout this case study refers to all of these satellites. A typical CubeSat consists of a flight computer, communications system, ADCS, electrical power supply (EPS) and an experimental payload [1]. A CubeSat is a complex system of systems; the flight computer controls and manages each of the other subsystems throughout a mission.

B. MLTL

The R2U2 tool reasons over specifications written in mission-time linear temporal logic (MLTL) [14, 15]. Temporal logic is used to formally describe requirements for a system.

Definition 1. *(MLTL Syntax) The syntax of an MLTL formula ϕ using a set of atomic propositions $\mathcal{AP}$ is recursively defined as the following:*

$$\phi ::= true \mid false \mid p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \Box_I \phi \mid \Diamond_I \phi \mid \phi_1 \mathcal{U}_I \phi_2 \mid \phi_1 \mathcal{R}_I \phi_2 \tag{1}$$

where p $\in \mathcal{AP}$ is an atom and all ϕ are atomic propositions. Each symbol stands for the following: $\neg$ = not, $\wedge$ = logical and, $\vee$ = logical or, $\Box_I$ = globally, $\Diamond_I$ = future, $\mathcal{U}_I$ = until, $\mathcal{R}_I$ = release. I is an interval [lb,ub], where lb $\leq$, lb, ub $\in \mathbb{N}$.

This formalism may be unfamiliar to some CubeSat developers, so we provide multiple examples encoding specifications into MLTL in Section IV. Extracting specifications for cyber-physical systems has proven to be a massive hurdle in practice [10]. Regular English statements pose the difficulty of handling unstated assumptions or implications. These factors pose a risk of misinterpretation when implemented into a real-time system [10]. Encoding these requirements removes this potential layer of abstraction, clearly defining the writer's intent.

For example, a requirement that "the voltage of subsystem X should never exceed 5V during the mission" can be encoded into an MLTL specification in the following manner. First, we define the measured system X voltage as `sys_voltage` and the 5V maximum to be a bound called `v_bound`. We write the relation that the system voltage should not exceed this bound with an atomic proposition, shown below:

$$\phi_1 = \texttt{sys\_voltage} \leq \texttt{v\_bound} \tag{2}$$

Analyzing a system trace with this proposition returns a Boolean; "True" this is met or "False" this is not met. This requirement is valid throughout the entire mission, per the English statement. Using the above atomic proposition, we encode the full requirement into a specification:

$$\Box_{[0,M]}(\phi_1) \tag{3}$$

With MLTL, we can more effectively express this requirement while accounting for off-nominal data points caused by sensor noise or incorrect bits. To take this into account, we provide an additional temporal operator to the specification:

$$\Box_{[0,M]}(\Box_{[0,2]}\phi_1) \tag{4}$$

This specification allows for up to three off-nominal data points before a fault is declared. Including this additional operator removes the chance of false positives or false negatives.

III. CubeSat Communications System Model

We created, tested, and evaluated a model of the Wertz CubeSat communications system [16] using MATLAB/Simulink as well as software-defined radio (SDR) hardware to provide real communication between transmitting and receiving antennas. We decompose the model into three sub-systems: orbital simulation (Section III.A), simulated communications (Section III.B), and radio transmission (Section III.C).

A. Orbital Simulation

We utilize the MathWorks Aerospace Blockset CubeSat Simulation Library in MATLAB/Simulink to develop a simulated CubeSat orbiting the Earth [17]. This library provides a robust set of functions that can be used to model, simulate, and analyze the orbit and attitude dynamics of CubeSats [18]. The CubeSat Vehicle Model block diagram provided by the library is able to perform this simulation once a specified set of Keplerian orbital parameters (eccentricity, semi-major axis, inclination, right ascension of the ascending node, etc.) are passed to it [19]. An orbital model of a CubeSat vehicle is then generated according to these parameters and placed into a spherical harmonic gravitation model representing Earth's gravity. When the Simulink model is run using an ordinary differential equations (ODE) solver, it outputs the position and velocity of the craft at the next time step, along with the quaternion between ECI to Body frames. This movement is animated using the Simulink 3D Animation block, as shown in Figure 1 and then the process is repeated with the new position and attitude values.

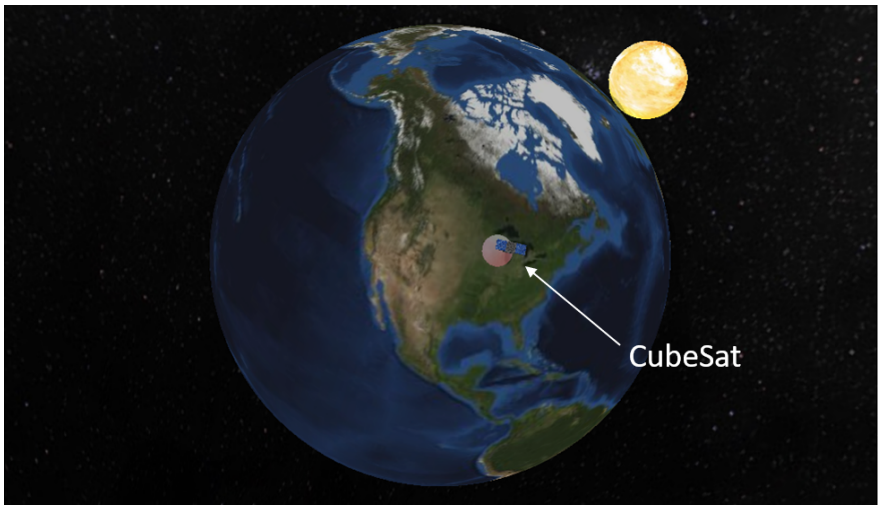

Fig. 1 A screen capture of the CubeSat Orbit Animation window from the Aerospace Blockset CubeSat Simulation Library. This image shows the CubeSat within line-of-sight of the proscribed virtual ground station located in Ames, Iowa.

The values we use for the orbital simulation are shown in Table 3. The simulation time is chosen to be one week in order to provide a non-trivial time interval with many orbital passes, while also being reasonable for computation time. All values shown in this table are selected based upon the expected orbital parameters of ISU's first CubeSat mission, CySat [20]. This upcoming mission will be deployed via the nanoracks system on-board the International Space Station (ISS), so the orbital parameters are very close to those of the ISS. This portion of the model serves to provide us with orbital information of a user-defined CubeSat mission rather than building and launching a physical craft. We could use any Earth-orbiting CubeSat mission for this case study, provided that orbit brings the craft within line-of-sight of the virtual ground station at some point in time during the simulation.

The positions and velocities of the CubeSat calculated in the orbital simulation allow for imitating its interaction with a virtual ground station. We choose the location of the ground station to be at a latitude of 42.0236°N and a longitude of -93.6528°W as described in Table 3 in the Appendix. These parameters correspond to expected values for the CySat mission, which will use a real ground station at these coordinates to download data from the satellite [20].

We also note that the simulation assumes sea-level altitude for the ground station's position, and does not include topographical differences that might be present if the ground station location were to be changed.

We next use output from the orbital simulation blocks along with the ground station position to determine if line-of-sight (LOS) between the CubeSat and ground station is achieved at each time step. Using the satellite position vector, and ground station position, we calculate the azimuth and elevation of the CubeSat with respect to the ground station. The simulation can then determine whether or not LOS is achieved between the CubeSat and the ground station by monitoring only the calculated elevation angle. Figure

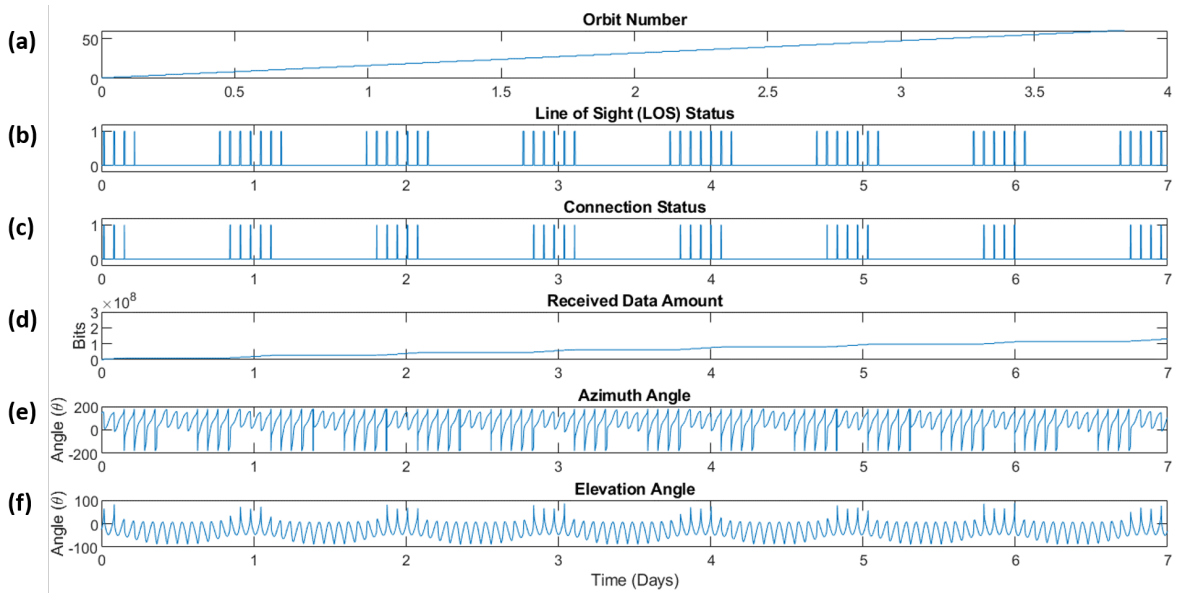

Fig. 2 Datasets recorded during the orbital simulation. (a) The orbit number of the CubeSat throughout the simulation. (b) Binary representation of when line-of-sight (LOS) is achieved between the CubeSat and the ground station. (c) Binary representation of when the signal-to-noise ratio (SNR) is great enough during LOS that communication between the CubeSat and ground station is possible. (d) Estimated number of bits received by the ground station with a 9600 baud rate. (e) Azimuth angle of the CubeSat with respect to the ground station. (f) Elevation angle of the CubeSat with respect to the ground station.

2 shows every dataset generated by the orbital simulation, including the number of orbits the CubeSat experiences, a binary representation of the LOS calculation, a binary representation of CubeSat and ground station connection, simulated received bits with the specified baud rate, and the azimuth and elevation angles of the CubeSat.

The system variables referring to all of the datasets in Figure 2 are shown in Table 1. We refer to these variables throughout the rest of the paper, when describing the formal specifications written for each as well as in analyzing the RV output.

Table 1 Variables representing the dataset outputted by the orbital simulation portion of the communications system model.

Signal	Description
`Time`	Number of seconds into simulation.
`LOS`	Line of sight indication.
`Conn`	Connection status indication.
`Received`	Theoretical data downlinked with 9600 baud.
`OrbitNum`	Number of orbits since simulation start.
`Az`	Azimuth angle of spacecraft wrt ground station.
`El`	Elevation angle of spacecraft wrt ground station.

B. Simulated Communications

When designing a real CubeSat communications system, developers typically perform link budgeting early to determine several properties in their communications system [21]. These calculations aid in understanding

the gain and power requirements for the transmitting and receiving antennas of the system and are a primary factor in determining what kind of equipment is necessary to build a communication link. Link budgeting is not an exact science, since the main equations assume an ideal system [22], but it is commonly used by CubeSat developers. Because we are simulating a communications system for this case study, we model the antenna directly first and then calculate the link during the simulation.

For the transmitting antenna, we model a microstrip patch antenna using the Mathworks Antenna Toolbox [23]. This type of antenna is popular with CubeSat developers due to their compact size and efficiency [24–26]. The ground station antenna parameters are derived from an MSquared 436CP42UG antenna, though in our model this only influences the antenna receiver gain [27]. As with the orbital simulation, we choose the parameters for the transmitting antenna used in the simulation based on those of a real antenna on the upcoming CySat mission [20]. All parameters for the transmitting antenna, shown in 4, approximately model this antenna except for the transmitting frequency. The transmission frequency is chosen to be 3.4 GHz in the S-band, which is currently a popular band for developers interested in downlinking relatively large amounts of data [21, 28–31]. We then produce the antenna's gain pattern in matrix-form outside of the simulation, which is used in determining the link.

During the simulation, the antenna gain pattern is loaded in during the beginning setup. While the orbital simulation of the model progresses, the signal-to-noise ratio (SNR) is calculated at each time step that LOS is achieved. This ratio is calculated using the link equation, which relates all necessary parameters in a communication system including but not limited to: data rate, antenna size, propagation path length, and transmitter power [32, 33]. The general form of the link equation is shown in Equation 5. A required SNR for the connection is predefined as 3 dB, and communication is considered possible when the calculate SNR is equal to or greater than this threshold. Communication achievement is plotted as a Boolean in Figure 2.c. In addition to communication status, the theoretical amount of data that can be transmitted is also tracked, and is shown in Figure 2.d.

$$\frac{E_b}{N_o} = P + G_t + L_{pt} + L_l + L_{pr} + L_s + L_a + G_r + 228.6 - 10 \cdot \log(T_s) - 10 \cdot \log(R) \tag{5}$$

where E_b/N_o is the ratio of received energy per bit to noise-density, P is the power of the transmitter, G_t is the TX antenna gain, L_l is the transmitter-to-antenna line loss, L_{pr} is the receiving antenna pointing loss, L_s is the space loss, L_a is the propagation and polarization loss, G_r is the RX antenna gain, T_s is the system noise temperature, and R is the data rate of the link [32]. The value of L_s changes greatly as the CubeSat orbits, and is therefore an important parameters in this calculation. L_{pt} and L_{pr} also play an important role, since accurate pointing is necessary for directional antennas. The term L_a is a function of factors consisting of atmospheric and precipitation-induced losses, and is assumed to be negligible in the model.

C. Radio Transmission

The radio transmission portion of the model can be considered a component of the communications simulation. It uses a Universal Software Radio Peripheral (USRP) B210 software-defined radio (SDR) from Ettus Research to perform quaternary phase-shift keying transmission and reception of sample telemetry data, as if a CubeSat were transmitting to the ground station. This portion of the model is built using the MathWorks Communications Toolbox [34]. Telemetry is broadcasted from the transmitting antenna to the receiving antenna when the communication status, shown in Figure 2.c is active. The transmission takes place specifically at each time step where the communication status goes from inactive to active.


```
{{"_index":"beacon_1","_type":"_doc","_id":"aa23xx-32123d","_score":1,"_source":
{"LI 3V3 Current":0.22943999999999998,"LI 3V3 Voltage":3.316,"LI VBATT Current":-
0.19289599999999998,"LI VBATT Voltage":8.2698,"Lithium #RX":0,"Lithium #TX":173484,
"Lithium MSP430 Temp":0,"Lithium Op Count":7911,"Lithium PA Temp"
:16.764199712583377,"Lithium RSSI":106}}}
```


Fig. 3 A sample of the simplified telemetry messages that are broadcasted in the communications model. This information is contained within a file. Each parameter listed in the telemetry is a typical piece of information that could be collected in a real CubeSat mission.

Each transmission consists of a sample telemetry dataset contained within a file. The telemetry message is comprised of a handful of parameters relating to the operation of a CubeSat's on-board communications system that would be of interest during a real mission. A sample telemetry message that is broadcasted in this model is shown in Figure 3. This sample message is a simplified version of a typical CubeSat telemetry message, which may consist of hundreds of parameters covering all CubeSat systems. Using a larger message is possible with the configuration we have developed, but for the sake of simplicity, we maintain a small set of values in the message.

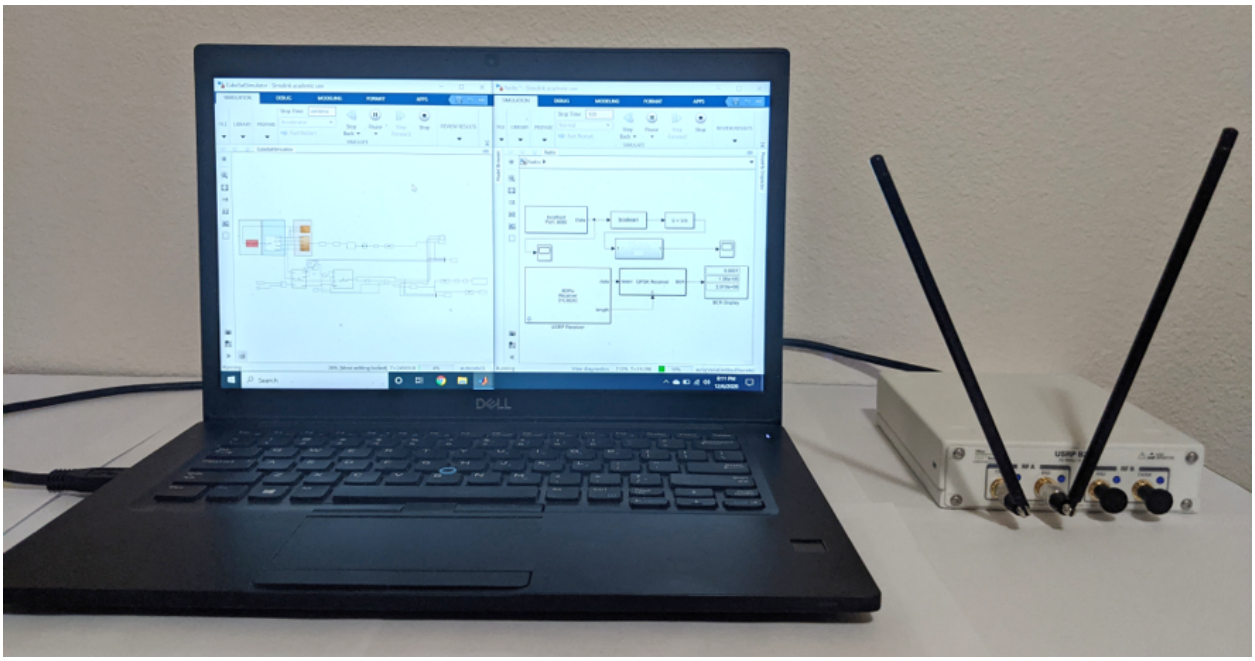

Fig. 4 The test setup used in this case study. A Dell Latitude 7490 is used for running the communication simulation. A Universal Software Radio Peripheral (USRP) B210 Software Defined Radio (SDR) performs the communication between ANT500 telescopic transmitting and receiving antennas.

When the communications simulation indicates transmission is active, this telemetry message is read into the model and encoded into bits, preceded by a unipolar Barker code [35, 36]. The message then goes through quadrature phase shift keying (QPSK) modulation with a $\pi/4$ phase offset. The signal is sent through a raised cosine transmit filter, which upsamples and filters the input signal to prepare it for transmission. This signal is then sent to the Universal Software Radio Peripheral (USRP) B210 SDR and transmitted at 3.4 GHz; the same frequency used in the model. The receiver, which is constantly listening for a signal throughout the simulation period, acquires the transmitted waveform and processes it through a similar demodulation process and a data decoder. This decoder outputs the telemetry message, which is then organized into a comma-separated variable (CSV) file for later use. Each transmission that is received is appended to this file, generating a collection of telemetry messages received throughout the simulation. The system variables referring to all of the datasets in Figure 3 are shown in Table 2. We refer to these variables throughout the rest of the paper, when describing the formal specifications written for each as well as in analyzing the RV output.

IV. Communications System Runtime Specifications

We elicitate specifications for communicating orbital data (Section IV.A and telemetry data (Section IV.B) in turn. The specifications for the model are written using temporal logic, to clearly define their meaning and prevent misinterpretation [10]. Mission-time Linear Temporal Logic (MLTL) is ideal for describing such

Table 2 Variables that make up the sample telemetry that is broadcasted by the radio communications portion of the model.

Signal	Description
`Beacon`	Identifier
`Type`	Identifier
`ID`	Identifier
`Score`	Identifier
`Radio_3V3_C`	3V3 line current to radio.
`Radio_3V3_V`	3V3 line voltage to radio.
`Radio_VBATT_C`	Current to radio battery.
`Radio_VBATT_V`	Voltage to radio battery.
`Radio_RX`	Number of radio receptions.
`Radio_TX`	Number of radio transmissions.
`Radio_Temp`	Temperature of radio.
`Radio_PA_Temp`	Temperature of radio power amplifier.
`Op_Count`	Number of operations perfomed by radio.
`RSSI`	Received signal strength indicator.

a model because it can specify over integer-bounded time intervals that we can map to our system update rates at implementation time, thus allowing for more generic, re-useable specifications [14, 15, 37]. MLTL language has been used in many industrial projects and since 2018 has been an official logic of the Runtime Verification Benchmark Competition [11, 15, 38–45].

When writing our specifications, we consider the runtime specification patterns and specification coverage strategies introduced in [10] and utilized in [11]. These studies highlighted the necessity of classifying specifications according to how they bound the system behavior, to ensure coverage of common fault sets. For example, a sensor bound specification may say that the azimuth angle of a CubeSat cannot exceed (-180°, 180°). Typical CubeSat systems share many of these specification patterns with the Unmanned Aircraft Systems (UAS) verified in [11]. Due to limitations with our model, we do not present specifications in the control-sequence and inter-sensor relationship categories, since we validate the orbital simulation and telemetry collections separately.

A. Orbit Specifications

The orbital simulation dataset defines updates at a time resolution of one second between measurements due to the ordinary differential equation (ODE) solver used to produce this dataset. Because so many constants and unchanging values define this part of the model, we actually need relatively few verification specifications. These specifications stem from a basic understanding of how communications systems operate as well as an understanding of spherical geometry. We discuss two of these specifications in-depth.

Communication between the spacecraft and the ground station can only occur when the spacecraft is above the horizon. If the CubeSat and ground station are not within line-of-sight of each other, then communication cannot occur. Such a situation would be physically impossible, but these bounds are not automatically assumed by an automated system. We therefore define this requirement in MLTL, with an atomic proposition indicating the elevation of the CubeSat must be greater than 0. An additional atomic proposition indicates when successful communication takes place, when the value of CONN is 1. The full specification, reasoning over the length of the entire CubeSat's mission, is shown in Specification IV.A.1. This specification ensures that communication only takes place when physically possible, and any measurement indicating otherwise can be flagged by the RV engine as an issue with measuring the position of the craft.

The azimuth angle of the spacecraft must always be bounded between -180° and 180°. This information

is easy to understand, considering the conventional spherical coordinate system. It is physically impossible in this coordinate system for a spacecraft to be located at an azimuth angle outside of this range. Any measurement outside of this range would certainly have to be either a bad data point, or the indication that some fault has occurred in the system. It is easy as humans to assume this requirement, but we must explicitly specify it for our autonomous CubeSat. With MLTL, we can robustly express this requirement while accounting for likelihood of off-nominal data points, e.g., from sensor noise. We declare atomic propositions for each of the bounds. Finally, we also specify a Δ value, indicating that the change in the azimuth angle of the spacecraft cannot change by more than 10° from the previous time step. If the spacecraft went from an azimuth angle of, say, -90° to 90° in one second, that would certainly be concerning. A third atomic proposition encodes this requirement. The full specification, reasoning over the length of the entire CubeSat's mission, appears in Specification IV.A.2. We write a very similar specification for the elevation angle, but with different bounds owing to the spherical coordinate system.

In addition to these specifications, we also describe the bounds of the elevation angle in Specification IV.A.3, and a monitor for the number of orbits the CubeSat experiences, in Specification IV.A.4. (We additionally wrote a specification to monitor for the amount of theoretically received data, shown in Figure 2.d that is almost identical to Specification IV.A.4.

1. ***Necessity of Line-of-Sight Communication***

"Communication between the spacecraft and ground station can only occur when the spacecraft is above the horizon, i.e., the elevation angle of the spacecraft must be above 0. (Line of sight conditions are met.)"

$$Atomic\ Propositions \begin{cases} \varphi_1 & (El_{SAT} > 0) \\ \varphi_2 & CONN == 1) \end{cases}$$

$$G_{[0,M]}\{(G_{[0,3]}(\varphi_2 \rightarrow \varphi_1))\} \tag{6}$$

This specification stems from common sense, as there must be line of sight between the CubeSat and the ground station.

2. ***Azimuth Angle Bounds***

"The azimuth angle of the CubeSat, Az_{SAT}, will be bounded by 0 and 360 at all instances. The current value of Az_{SAT} will not vary more than 1° from the previous time step."

$$Atomic\ Propositions \begin{cases} \varphi_1 & (Az_{SAT} \geq -180) \\ \varphi_2 & (Az_{SAT} \leq 180) \\ \psi & (abs(Az_{SAT} - Az_{SAT(i-1)}) < 1.0) \end{cases}$$

$$G_{[0,M]}\{(G_{[0,3]}(\varphi_1 \wedge \varphi_2 \wedge \psi))\} \tag{7}$$

3. ***Elevation Angle Bounds***

"The elevation angle of the CubeSat, El_{SAT}, will be bounded by -90 and 90 at all instances."

$$Atomic\ Propositions \begin{cases} \varphi_1 & (El_{SAT} \geq -90) \\ \varphi_2 & (El_{SAT} \leq 90) \\ \psi & (abs(El_{SAT} - El_{SAT(i-1)}) < 1.0) \end{cases}$$

$$G_{[0,M]}\{(\varphi_1 \wedge \varphi_2 \wedge \psi)\} \tag{8}$$

4. ***Orbit Number Check***

"The orbit number of the spacecraft provided by the model will never be less than 0. The orbit number of the current time step will be equal to or greater than the orbit number from the previous time step."

$$Atomic\ Propositions \begin{cases} \varphi_1 & (OrbNum \geq 0) \\ \psi_1 & (OrbNum \geq OrbNum_{i-1}) \end{cases}$$

$$G_{[0,M]}\{(\varphi_1 \wedge \psi_1)\} \tag{9}$$

This information is crucial to knowing how many passes the CubeSat has made, and also acts as a check to ensure the information pulled from the model is sensible.

B. Telemetry Specifications

CubeSat developers tend to use component specification sheets when testing their systems, and use the bounds described in those sheets for validating their work. Because the data sheet component specifications often end up acting as requirements when testing the CubeSat, we use them to write our specifications of the telemetry variables.

When considering voltage lines powering CubeSat systems, developers generally accept a tolerance of ±10%. Unless switched into another state, voltage lines commonly remain active during the mission. Therefore, we write the following specification. The radio 3.3V line must not vary more than 10% from the desired 3.3V value. The upper and lower bounds are each encoded into atomic propositions. We also consider the case where the measured voltage is within these bounds but may experience oscillation. For this reason, we define a third atomic proposition indicating that the voltage measurement must not vary more than .5V from the previous time step. The full specification, reasoning over the length of the entire CubeSat's mission, appears in Specification IV.B.7. Such a specification is useful for any voltage that is measured and added to the telemetry message, and a similar form exists for measured currents and temperatures as well.

The sample telemetry also lists the number of times that the CubeSat's on-board radio has transmitted. This is an interesting piece of information since it alerts ground station operators as to how many times the radio is attempting to communicate, and if whether or not those communications are received. We write the following specification: the number of times the CubeSat transmits to the ground station will be greater than the previous time the transmission was received. This is another situation where common sense is helpful. We know that the transmit count should be higher between one transmission and the next, but we don't necessarily know if that transmission was the next one to be sent. Hence, keeping track of the radio transmit counts sheds light on how often communication occurs. This specification is encoded into a single atomic proposition without a temporal operator, since each successive measurement of the transmit count should be higher. The full specification, reasoning over the length of the entire CubeSat's mission, is shown in Specification 11. Such a specification is useful for understanding the time evolution of CubeSat telemetry, and diagnosing how effective the communications system is.

We have also written specifications for different temperatures of the radio systems, shown in Specifications 5 and 6. Current variation for the 3.3V source and Battery Voltage are shown in Specifications 8 and 10. Similar to Specification 11, we also look at the number of operations the CubeSat's radio performs, detailed in Specification 12.

*5. **Radio Temperature Variation***

"The temperature of the radio, Radio_PA_Temp, will not vary more 1° from the previous time step."

$$\textit{Atomic Propositions} \begin{cases} \varphi & (abs(Radio\_Temp - Radio\_Temp_{i-1} \leq 1.0)) \end{cases}$$

$$G_{[0,M]}\{(\varphi)\} \tag{10}$$

*6. **Radio Power Amplifier Temperature Variation***

"The temperature of the radio's power amplifier, Radio_PA_Temp, will not vary more 1° from the previous time step."

$$\textit{Atomic Propositions} \begin{cases} \varphi & (abs(Radio\_PA\_Temp - Radio\_PA\_Temp_{i-1} \leq 1.0)) \end{cases}$$

$$G_{[0,M]}\{(\varphi)\} \tag{11}$$

7. ***Radio 3.3V Voltage Variation***

"The Radio 3.3V line, 3.3VRadioVolt, must not vary more than 10% from the desired 3.3V value. It must also not vary more than .5V from the previous time step."

$$Atomic\ Propositions \begin{cases} \varphi_1 & (3.3VRadioVolt \geq 2.97V) \\ \varphi_2 & (3.3VRadioVolt \leq 3.63V) \\ \psi & (abs(3.3VRadioVolt - 3.3VRadioVolt_{i-1}) < .5) \end{cases}$$

$$G_{[0,M]}\{G_{[0,3]}(\varphi_1 \wedge \varphi_2 \wedge \psi))\} \tag{12}$$

8. ***Radio 3.3V Current Variation***

"The Radio 3.3V line must not have a current, 3.3VRadioCurr, varying more than 5% from the desired current value, 3.3VRadioExpCurr. It must also not vary more than .05A from the value of the previous time step."

$$Atomic\ Propositions \begin{cases} \varphi_1 & (3.3VRadioCurr \geq .95 * 3.3VRadioExpCurr) \\ \varphi_2 & (3.3VRadioCurr \leq 1.05 * 3.3VRadioExpCurr) \\ \psi & (abs(3.3VRadioCurr - 3.3VRadioCurr_{i-1}) \leq .05) \end{cases}$$

$$G_{[0,M]}\{(\varphi_1 \wedge \varphi_2 \wedge \psi)\} \tag{13}$$

9. ***Radio VBatt Voltage Variation***

"The Radio VBatt voltage, RadioVBattVolt, must not vary more than 10% from the desired 8.3V value. It must also not vary more than .5V from the previous time step."

$$Atomic\ Propositions \begin{cases} \varphi_1 & (RadioVBattVolt \geq 7.47V) \\ \varphi_2 & (RadioVBattVolt \leq 9.13V) \\ \psi & (abs(RadioVBattVolt - RadioVBattVolt_{i-1}) < .5) \end{cases}$$

$$G_{[0,M]}\{(\varphi_1 \wedge \varphi_2 \wedge \psi)\} \tag{14}$$

10. ***Radio VBatt Current Variation***

"The Radio VBatt voltage line must not have a current, RadioVBattCurr, varying more than 5% from the desired current value, RadioVBattCurrExp. It must also not vary more than .05A from the value of the previous time step."

$$Atomic\ Propositions \begin{cases} \varphi_1 & (RadioVBattCurr \geq .95 * RadioVBattCurrExp) \\ \varphi_2 & (RadioVBattCurr \leq 1.05 * RadioVBattCurrExp) \\ \psi & (abs(RadioVBattCurr - RadioVBattCurr_{i-1}) \leq .05) \end{cases}$$

$$G_{[0,M]}\{(\varphi_1 \wedge \varphi_2 \wedge \psi)\} \tag{15}$$

11. ***Transmit Count Check***

"The number of times the CubeSat transmits to the ground station, Radio_TX, will be greater than the previous time the transmission was received."

$$Atomic\ Propositions \begin{cases} \varphi & (Radio\_TX > Radio\_TX_{i-1}) \end{cases}$$

$$G_{[0,M]}\{(\varphi)\} \tag{16}$$

The number won't necessarily be greater by exactly 1, since the ground station could conceivably not receive a transmission.

12. ***Operation Count Check***

"The number of times the CubeSat performs operations, Radio_Op_Count, will be greater than the previous time the transmission was received."

$$Atomic\,Propositions \begin{cases} \varphi & (Radio\_Op\_Count > Radio\_Op\_Count_{i-1}) \end{cases}$$

$$G_{[0,M]}\{(\varphi)\} \tag{17}$$

The number won't necessarily be greater by exactly 1, since the ground station could conceivably not receive a transmission.

V. Results

The simulated communications system we have developed for this case study monitors and outputs orbital data and simulated CubeSat telemetry during a weeklong simulation period. In this section, we compare this data offline, apart from the simulation, against the designed MLTL specifications using the R2U2 tool. In particular, we examine the time necessary to locate faults within these datasets, the amount of memory required by the tool while processing, and other additional statistics detailing the resources required to verify system performance. The R2U2 tool is not integrated into the model due to being run offline. Post-simulation, the data is run through a pre-processing layer that maps multiple inputs into a single Boolean atomic input for R2U2, which is described in further detail in [11]. This methodology provides lower precision than embedding the tool but greatly simplifies the verdicts that it returns.

Throughout the duration of this case study, R2U2 was hosted on an Ubuntu 20.04 LTS operating system on an Intel Core i7-6700K CPU with a 4.00 GHz clock and 32GB of RAM. Separate instances of R2U2 were run for the two datasets, independent of one another. To better understand how the specifications are encoded into observation trees for R2U2, see [11, 41, 42].

A. Orbital Dataset Verification

The orbital dataset is collected throughout the simulation. The data file consists of the **Time** (s), the system's **Connection** status in Boolean representation, the CubeSat's estimated **Received** data with a 9600 baud rate measured in bits, the system's **LOS** status in Boolean representation, the CubeSat's **OrbitNumber** measured as an integer, and the **Azimuth** and **Elevation** of the CubeSat measured in degrees. In total, there are 604,801 measurements of each of these parameters during the simulation.

After the simulation is run, we put the orbital data through the pre-processing layer to output a single Boolean atomic input. Using this methodology, we individually validate each of the specifications we have developed for the model. For the orbital simulation, R2U2 monitors and reports if operating ranges, sensor bounds, and rates of change are satisfied for all of the orbital measurements.

Figure 5 shows the results for Specification 1 discussed in Section IV. The top panels show the elevation angle and CONN status of the CubeSat throughout the simulation, and the bottom panels show the RV output when that data is run through R2U2 with Specification 1. Figure 5.a shows a nominal simulation wherein no faults occur. Figure 5.b shows a simulation with an injected fault in the elevation data, ∼3 days into the simulation. R2U2 correctly identifies this fault with our specification.

Figure 6 shows the results of Specification 2 discussed in Section IV. The top panels show the azimuth angle of the CubeSat throughout the simulation and the bottom panels show the RV output when the data is run through R2U2 with Specification 2. Figure 6.a also shows a nominal situation with no faults. Figure 6.b shows a simulation with an injected fault in the azimuth data, ∼4.5 days into the simulation. R2U2 correctly identifies the fault here as well.

B. Telemetry Dataset Verification

The telemetry data is collected each time the Connection parameter in the orbital simulation goes active. Using the parameters given in Tables 3,4, and 5 with the simulation results in 46 instances of a virtual connection between the CubeSat and the ground station. The telemetry data consists of a sample set of variables relating to a CubeSat's radio system that would be measured during a real CubeSat mission.

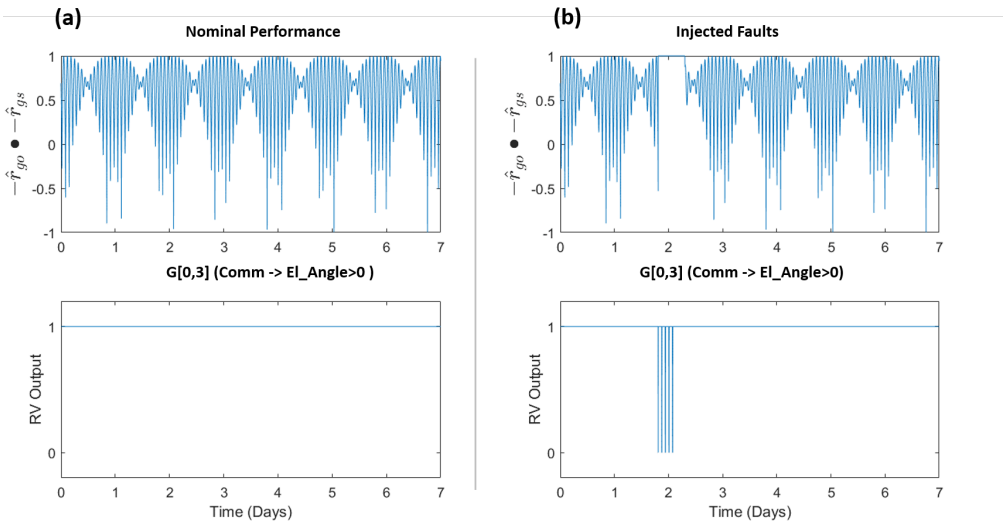

Fig. 5 Results of running Specification 1, titled "Necessity of Line-of-Sight Communication", in R2U2 with multiple datasets. Top plots show a calculated elevation angle of the CubeSat with respect to the ground station. Bottom plots show RV output when the specification is run against the dataset in R2U2. (a) Nominal performance with no change. (b) Injected faults, one during the time of an orbital pass. Temporal operators account for multiple off-nominal measurements before a fault is declared.

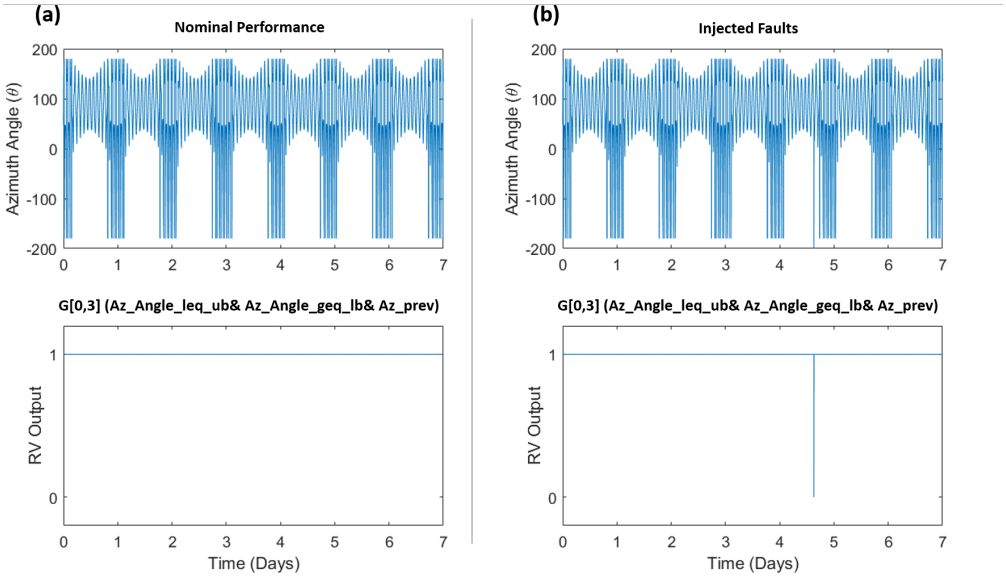

Fig. 6 Results of running Specification 2, titled "Azimuth Angle Bounds", in R2U2 with multiple datasets. (a) Nominal performance with no change. (b) Azimuth data with an injected fault. Temporal operators account for multiple off-nominal measurements before a fault is declared.

Similar to the orbital dataset, we put the data through a pre-processing layer to output a single Boolean atomic input and individually validate each of the specifications we have developed.

Figure 7 shows the 3.3V voltage measurement throughout the simulation period. Figure 7.a represents a nominal simulation wherein no faults occur. Figure 7.b represents a simulation where a few bad data points

are collected, but due to the temporal operators of Specification 7, they do not cause a fault. Figure 7.c represents a simulation in which a failure takes place and the 3.3V line slowly lowers to 0V. As indicated by the RV output in Figure 7.c, a few off-nominal data points are allowed by the temporal operators before declaring a fault has occurred.

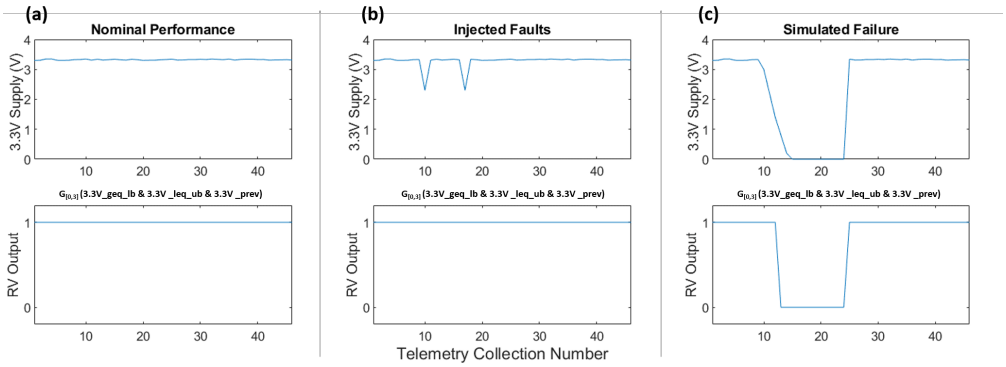

Fig. 7 Results of running Specification 7, titled "Radio 3.3V Voltage Variation", in R2U2 with multiple datasets. (a) Nominal performance with no change. (b) Multiple injected faults, considered false positives, are accounted for with temporal operators and do not indicate a fault. (c) Simulated failure of voltage supply, fault declared after multiple poor measurements indicates issue.

The 3.3V specification is worth discussing more extensively. Because a CubeSat is a single system composed of multiple subsystems, for a given CubeSat project there could be up to several dozen voltages the developers would want to monitor during the mission phase. Formally specifying these voltages is consistent with the 3.3V voltage monitor described in Specification 7 aside from the bounds assigned to the atomic propositions. This can be seen with Specification 9, which monitors the 8.3V supplied to the radio's battery. Aside from adjusting the lower and upper bounds, this specification is identical to that of the 3.3V voltage monitor.

Figure 8 shows the radio battery voltage measurement throughout the simulation period. Just like with the 3.3V monitor, R2U2 reports if the upper and lower bounds of this voltage parameter are met at each time step. Aside from separate bounds, the specification used to describe this variable is exactly the same as for the 3.3V variable.

VI. Discussion

All but two of the variables comprising the orbital dataset are formally specified, the `Time` and the `Received` data. The received data acts as a simple measure of bits, but in reality would be much more complex. Because the simulation time is user-defined in our model and is guaranteed to be correct for each time step, there's no utility in defining its operation. With a real-world CubeSat project however, it would be very useful to look at the timestamp at which a certain CubeSat position was recorded, in order to ensure that the desired orbit persists.

In reality, many CubeSat developers extract orbital information about their CubeSats from the gpredict satellite tracking and orbit prediction software [46]. This tool provides an extensive list of parameters for a given satellite or CubeSat that can be used to track its position, determine its velocity, altitude, expected Doppler shift, and more. Collecting all of this information in one location makes the tool popular with developers and amateurs alike. The gpredict tool's calculations for a CubeSat are verified through their own system. If, however, developers wanted to create a separate model of their CubeSat system, then gpredict could be used to verify that the outputs are correct.

There are fourteen separate variables contained within the sample telemetry. Of these, our specifications cover nine. Information such as "_index", "_type" and "_id" are handy identifiers for the telemetry message but do not serve to specify a variable in any way. The purpose of these variables is to identify the telemetry message, and would instead be used to check if the signal is from a specific CubeSat. Formally specifying

these variables is possible, but it would likely be just as well to check them with a simple string comparison. The variable "_score" may either rate the telemetry or indicate what kind of telemetry the specific message is, so we have not explored it here. Finally, we did examine the received signal strength indicator (RSSI) variable "Lithium RSSI", which is a measure of beacon signal strength from the radio receiver. This variable also essentially acts as a rating for the signal, so we have not explored it here.

A. Lessons Learned

Without real hardware to test, verification of a CubeSat communications system model presents several difficulties. Many of the results in the simulation are guaranteed to be correct, such as the time the CubeSat is in a given position. We are limited in this respect because timestamps of spacecraft position are examined in practice to ensure they are sensible. Additionally, the telemetry messages that were transmitted and received were not influenced by the system from which they were derived. We have discussed how utilizing COTS components, usually cheaper than the parts used on large-scale spacecraft missions, comes with a trade-off of lower accuracy. Because we developed a model instead of working with actual hardware, we could not account for these problems. To comprehensively explore the verification and validation of a CubeSat communications system, we would develop a real mission like that shown in Figure 9 and integrate RV into the ground station portion to validate orbital data and information sourced from the CubeSat.

CubeSats are systems comprised of multiple subsystems, and the chance of an incorrect measurement or bad transfer of data through the systems is a real possibility. In order to ensure that CubeSat systems are operating correctly, it is essential to check that the data measured in the telemetry meets requirements. We have performed introductory exploration of this for a sample set of telemetry values, but did not examine an entire telemetry dataset. As pointed out in Section IV, the specification for a line voltage (3.3V) is all but identical to the specification for another system voltage (VBATT). While system states need to be taken into account,we could use this method for formally specifying every system voltage contained within a CubeSat telemetry dataset. We plan to explore this further in future work.

While we have discussed implementing RV into a CubeSat communications system to verify orbital information and sample telemetry, a case could be made to verify all of the telemetry measurements prior to communicating them to a ground station. More specifically, RV could be embedded into the CubeSat flight computer so that all of the measurements contained within the telemetry are validated by the CubeSat itself.

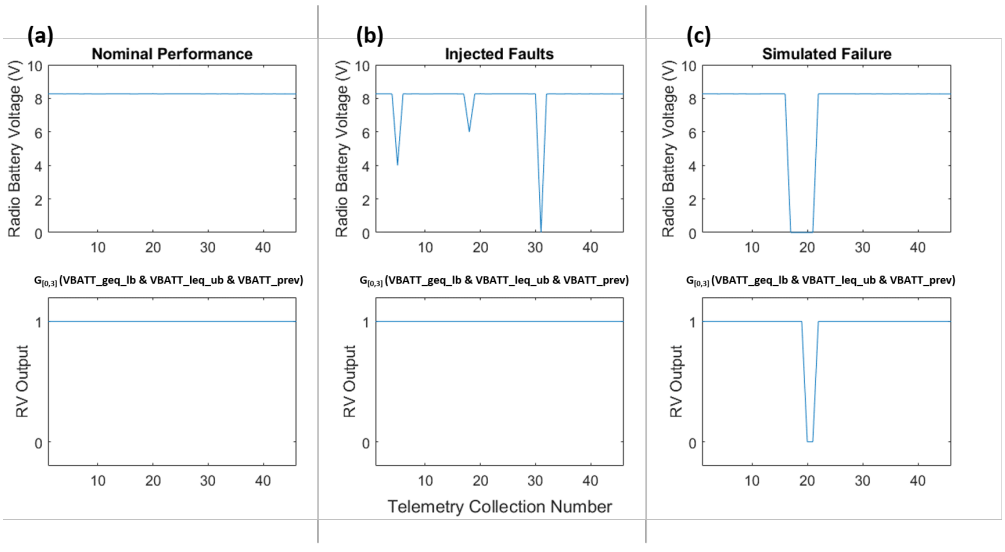

Fig. 8 Results of running Specification 9, titled "Radio VBatt Voltage Variation", in R2U2 with multiple datasets. (a) Nominal performance with no change. (b) Multiple injected faults, considered false positives, are accounted for with temporal operators and do not indicate a fault. (c) Simulated failure of voltage supply, fault declared after multiple poor measurements indicates issue.

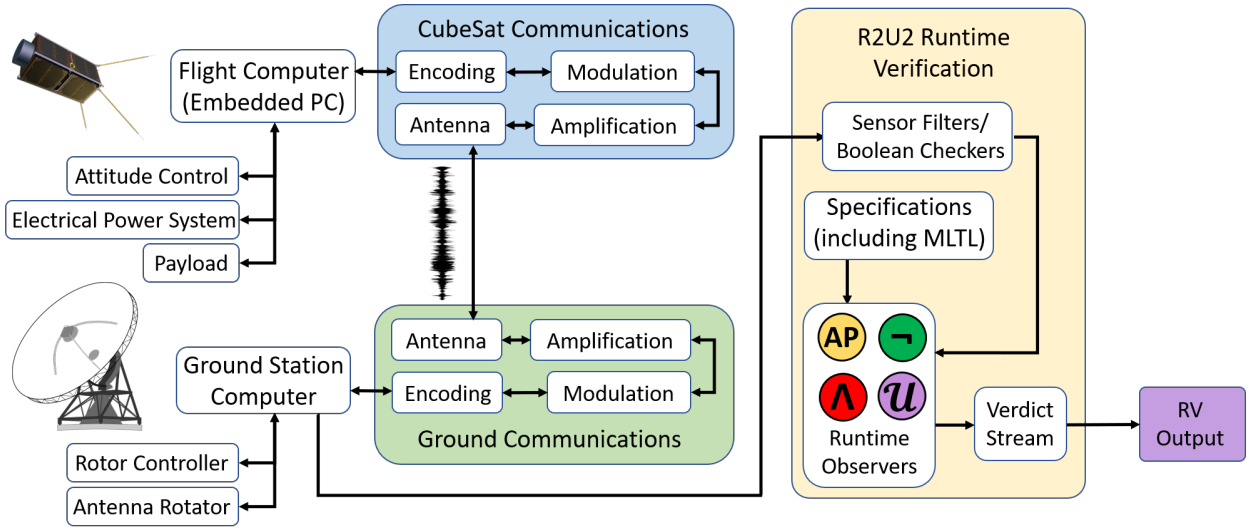

Fig. 9 A representation of how runtime verification fits into a real CubeSat mission's communications system. The CubeSat measures system values and saves them for transmission. The telemetry information is then encoded, modulated, amplified and downlinked to the ground station. After a similar decoding process, the telemetry variables are checked against their specifications using the R2U2 tool. For an actual mission, this would consist of an online process wherein R2U2 is run each time the ground station receives a new set of telemetry. The communications system portion references [47] and the runtime verification section is derived from [48].

The ability to have the CubeSat validate its own systems would remove the need to send detailed telemetry messages, replaced instead with simple verdicts of each subsystem. Implementing this functionality into a CubeSat would mean fewer resources are used to communicate system health to the ground station, and would instead be available to transmit payload data. We will explore this idea as well in future work.

VII. Conclusion

The popularity of CubeSat projects is only expected to grow in the coming years, due mainly to their low cost and fast turnaround times. However, without addressing the high rate of failure of these projects, the common problems faced by developers will persist. Communications system failures have been described as one of the most common issues experienced in CubeSat projects, and such failures often lead to premature mission end. To aid in improved reliability of CubeSat communications systems, we have integrated the R2U2 runtime verification tool into two separate layers of the system: one within a ground station layer to monitor the CubeSat's orbit and connection, and one to monitor the CubeSat's telemetry. We provide a small reference set of specifications for these two separate aspects of the communications system, demonstrating their correctness by validating the model's system parameters offline. While we were limited by the model developed for this case study, we discuss how our work could be expanded for a real CubeSat mission.

Appendix

A. Orbital Simulation Data Values

B. Antenna Design Parameters

Table 3 Parameters of the simulated CubeSat orbit and location of ground station.

Parameter	**Value**
Radius of Earth (km)	6378.137
Avg. Radius of Orbit (km)	7078.137
Eccentricity	0
Inclination (deg)	51.6
RA of Ascending Node (deg)	0
Argument of Perigee (deg)	0
True Anomaly (deg)	0
GS Latitude (deg)	42.0236
GS Longitude (deg)	-93.6528

Table 4 Design Parameters of the Microstrip Patch Antenna used in developing the communications system model.

Parameter	**Value**
Length (m)	0.0397
Width (m)	0.0397
Strip Line Width (m)	0.0025
Notch Length (m)	0.0059
Notch Width (m)	0.0044
Patch Center [m,m,m]	[0,0,0]
Feed Location [m,m,m]	[-0.0441,0,0.0022]
Height(m)	0.0022
Ground Plane Length (m)	0.0882
Ground Plane Width (m)	0.0882
Patch Center Offset [m,m]	[0,0]
Feed Offset [m,m]	[-0.0441,0]
Substrate	1x1 dielectric [default values]
Substrate Thickness (m)	0.00220435630882353
Tilt	0
Tilt Axis	[1,0,0]

C. Downlink Constant Values

Table 5 Constant values used in the link budgeting step of the model.

Parameter	Value
frequency (Hz)	3.4 · 10^9
Aperture Efficiency	0.0
TX Power (W)	10.0
TX Line Loss (dB)	0.0
Sky Temperature (K)	290
Polarization Loss (dB)	3.0
Pointing Error Loss (dB)	3.0
LNA Gain (dB)	16.0
LNA Noise Figure (dB)	1.5
RX Line Loss (dB)	3.0
Radio Noise Figure (dB)	10.0
RX Antenna Gain (dB)	18.9
R (baud)	9600
Eb_N0_required (dB)	10

Acknowledgments

The authors would like to thank Dr. James W. Cutler of the University of Michigan for providing data, as well as his experience and knowledge of CubeSat communications systems.

References

[1] NASA CubeSat Launch Initiative, *CubeSat 101*, 1st ed., California Polytechnic State University, San Luis Obispo (Cal Poly) CubeSat Systems Engineer Lab, 2017.

[2] Poghosyan, A., and Golkar, A., "CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions," *Progress in Aerospace Sciences*, Vol. 88, 2016. https://doi.org/10.1016/j.paerosci.2016.11.002.

[3] Villela, T., Costa, C. A., Brandão, A. M., Bueno, F. T., and Leonardi, R., "Towards the Thousandth CubeSat: A Statistical Overview," *International Journal of Aerospace Engineering*, Vol. 2019, 2019, p. 5063145. https://doi.org/10.1155/2019/5063145, URL https://doi.org/10.1155/2019/5063145.

[4] Kulu, E., "Nanosatellite & CubeSat Database," , Apr 2020. URL https://www.nanosats.eu/database.

[5] Mike Tolmasoff, Renelito Delos Santos, and Catherine Venturini, "Improving Mission Success of CubeSats," , May 2007.

[6] Langer, M., and Bouwmeester, J., "Reliability of CubeSats – Statistical Data, Developers' Beliefs and the Way Forward," , 08 2016.

[7] Peng, Z., Lu, Y., Miller, A., Johnson, C., and Zhao, T., "A Probabilistic Model Checking Approach to Analysing Reliability, Availability, and Maintainability of a Single Satellite System," *2013 European Modelling Symposium*, 2013, pp. 611–616. https://doi.org/10.1109/EMS.2013.102.

[8] Boufaied, C., Menghi, C., Bianculli, D., Briand, L., and Parache, I. Y., "Trace-Checking Signal-based Temporal Properties: A Model-Driven Approach," *ASE 2020*, 2020.

[9] Gross, K. H., Clark, M., Hoffman, J. A., Fifarek, A., Rattan, K., Swenson, E., Whalen, M., and Wagner, L., *Formally Verified Run Time Assurance Architecture of a 6U CubeSat Attitude Control System*, chapter and pages. https://doi.org/10.2514/6.2016-0222, URL https://arc.aiaa.org/doi/abs/10.2514/6.2016-0222.

[10] Rozier, K. Y., "Specification: The Biggest Bottleneck in Formal Methods and Autonomy," *Proceedings of 8th Working Conference on Verified Software: Theories, Tools, and Experiments (VSTTE 2016)*, LNCS, Vol. 9971, Springer-Verlag, Toronto, ON, Canada, 2016, pp. 1–19. https://doi.org/10.1007/978-3-319-48869-1_2.

[11] Cauwels, M., Hammer, A., Hertz, B., Jones, P., and Rozier, K., *Integrating Runtime Verification into an Automated UAS Traffic Management System*, chapter and pages, pp. 340–357. https://doi.org/10.1007/978-3-030-59155-7_26.

[12] NASA, "NASA Technology Roadmaps: TA 4: Robotics and Autonomous Systems," , Jul 2015. URL https://www.nasa.gov/sites/default/files/atoms/files/2015_nasa_technology_roadmaps_ta_4_robotics_and_autonomous_systems_final.pdf.

[13] Toorian, A., Diaz, K., and Lee, S., "The CubeSat Approach to Space Access," *2008 IEEE Aerospace Conference*, 2008, pp. 1–14. https://doi.org/10.1109/AERO.2008.4526293.

[14] Li, J., Vardi, M. Y., and Rozier, K. Y., "Satisfiability Checking for Mission-Time LTL," *Proceedings of 31st International Conference on Computer Aided Verification (CAV)*, LNCS, Vol. 11562, Springer, New York, NY, USA, 2019, pp. 3–22. https://doi.org/https://doi.org/10.1007/978-3-030-25543-5_1.

[15] Reinbacher, T., Rozier, K. Y., and Schumann, J., "Temporal-Logic Based Runtime Observer Pairs for System Health Management of Real-Time Systems," *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, Springer-Verlag, 2014, pp. 357–372.

[16] Wertz, J. R., Everett, D. F., and Puschell, J. J., *Space Mission Analysis and Design*, Microcosm Press, 2015.

[17] MATLAB, *version 9.7 (R2019b)*, The MathWorks Inc., Natick, Massachusetts, 2010.

[18] MathWorks Aerospace Products Team, "Aerospace Blockset CubeSat Simulation Library," , Sep 2020. URL https://www.mathworks.com/matlabcentral/fileexchange/70030-aerospace-blockset-cubesat-simulation-library.

[19] MathWorks Aerospace Products Team, "Model and Simulate CubeSats," , 2020. URL https://www.mathworks.com/help/aeroblks/model-and-simulate-cubesats.html.

[20] Nelson, M., Lee, D. Y., Kilcoin, M., Gordon, L., and Brown, W., "Preparing CySat-1: A look at Iowa State University's first CubeSat," *Proceedings of the 34th Annual Small Satellite Conference*, 2020. URL https://digitalcommons.usu.edu/smallsat/2020/all2020/29/.

[21] Lee, D., Sharma, S., Park, H., and Cutler, J. W., *Design and Optimization of a Small Satellite Communication System*, chapter and pages. https://doi.org/10.2514/6.2018-1940, URL https://arc.aiaa.org/doi/abs/10.2514/6.2018-1940.

[22] ZYREN, J., "Tutorial on Basic Link BudgetAnalysis," *http://www.sss-mag.com/ebn0.html*, 1998. URL https://ci.nii.ac.jp/naid/20001249712/en/.

[23] Mathworks, "Antenna Toolbox," , 2020. URL https://www.mathworks.com/products/antenna.html.

[24] Tharun, K., Vivekanand, V., Ravi, T., and Sugadev, M., "Design and Fabrication of Micro Strip Antenna for Cubesat Applications," *IOP Conference Series: Materials Science and Engineering*, Vol. 590, 2019, p. 012052. https://doi.org/10.1088/1757-899x/590/1/012052, URL https://doi.org/10.1088%2F1757-899x%2F590%2F1%2F012052.

[25] Neveu, N., Garcia, M., Casana, J., Dettloff, R., Jackson, D. R., and Chen, J., "Transparent microstrip antennas for CubeSat applications," *IEEE International Conference on Wireless for Space and Extreme Environments*, 2013, pp. 1–4. https://doi.org/10.1109/WiSEE.2013.6737542.

[26] Liu, W.-C., and Tang, T.-Y., "High-gain patch antenna for cubesat-based automatic dependent surveillance-broadcast application," *Microwave and Optical Technology Letters*, Vol. 61, No. 1, 2019, pp. 187–190. https://doi.org/https://doi.org/10.1002/mop.31518, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/mop.31518.

[27] M2 Antenna Systems, I., "436CP42UG, 420-440 MHz," , 2020. URL https://www.m2inc.com/FG436CP42UG.

[28] Pittella, E., Pisa, S., Pontani, M., Nascetti, A., D'Atanasio, P., Zambotti, A., and Hadi, H., "Reconfigurable S-band patch antenna system for cubesat satellites," *IEEE Aerospace and Electronic Systems Magazine*, Vol. 31, No. 5, 2016, pp. 6–13. https://doi.org/10.1109/MAES.2016.150153.

[29] Nascetti, A., Pittella, E., Teofilatto, P., and Pisa, S., "High-Gain S-band Patch Antenna System for Earth-Observation CubeSat Satellites," *IEEE Antennas and Wireless Propagation Letters*, Vol. 14, 2015, pp. 434–437. https://doi.org/10.1109/LAWP.2014.2366791.

[30] Tubbal, F., , Raad, R., Chin, K. W., and Butters, B., "S - band Planar Antennas for a CubeSat," *International Journal on Electrical Engineering and Informatics*, Vol. 7, No. 4, 2015, pp. 559–568. https://doi.org/10.15676/ijeei.2015.7.4.2, URL https://doi.org/10.15676/ijeei.2015.7.4.2.

[31] Islam, M. T., Cho, M., Samsuzzaman, M., and Kibria, S., "Compact Antenna for Small Satellite Applications [Antenna Applications Corner]," *IEEE Antennas and Propagation Magazine*, Vol. 57, No. 2, 2015, pp. 30–36. https://doi.org/10.1109/MAP.2015.2420471.

[32] Wertz, J., and Larson, W., *Space Mission Analysis and Design*, Space Technology Library, Springer Netherlands, 1999. URL https://books.google.com/books?id=veyGEAKFbiYC.

[33] "The Link Equation," , 2014. https://doi.org/10.1049/SBEW516E_ch5, URL https://digital-library.theiet.org/content/books/10.1049/sbew516e_ch5.

[34] MathWorks, "Communications Toolbox," , 2020. URL https://www.mathworks.com/help/comm/index.html?s_tid=CRUX_lftnav.

[35] BARKER, R. H., "Group sysnchronizing of binary digital systems," *Communication Theory*, 1953, pp. 273–287. URL https://ci.nii.ac.jp/naid/10025451520/en/.

[36] Soba, J., Munir, A., and Suksmono, A. B., "Barker code radar simulation for target range detection using software defined radio," *2013 International Conference on Information Technology and Electrical Engineering (ICITEE)*, 2013, pp. 271–276. https://doi.org/10.1109/ICITEED.2013.6676251.

[37] Li, J., and Rozier, K. Y., "MLTL Benchmark Generation via Formula Progression," *Proceedings of the 18th International Conference on Runtime Verification (RV18)*, Springer-Verlag, Limassol, Cyprus, 2018.

[38] Geist, J., Rozier, K. Y., and Schumann, J., "Runtime Observer Pairs and Bayesian Network Reasoners On-board FPGAs: Flight-Certifiable System Health Management for Embedded Systems," *Proceedings of the 14th International Conference on Runtime Verification (RV14)*, Vol. 8734, Springer-Verlag, 2014, pp. 215–230.

[39] Schumann, J., Rozier, K. Y., Reinbacher, T., Mengshoel, O. J., Mbaya, T., and 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)*, Vol. 6, No. 1, 2015, pp. 1–27.

[40] Rozier, K. Y., Schumann, J., and 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.

[41] Schumann, J., Moosbrugger, P., and Rozier, K. Y., "R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems," *Proceedings of the 15th International Conference on Runtime Verification (RV15)*, Springer-Verlag, Vienna, Austria, 2015.

[42] Schumann, J., Moosbrugger, P., and Rozier, K. Y., "Runtime Analysis with R2U2: A Tool Exhibition Report," *Proceedings of the 16th International Conference on Runtime Verification (RV16)*, Springer-Verlag, Madrid, Spain, 2016.

[43] Moosbrugger, P., Rozier, K. Y., and Schumann, J., "R2U2: Monitoring and Diagnosis of Security Threats for Unmanned Aerial Systems," 2017, pp. 1–31. https://doi.org/10.1007/s10703-017-0275-x.

[44] Rozier, K. Y., "2018 Runtime Verification Benchmark Competition," https://www.rv-competition.org/2018-2/, 2018.

[45] Kempa, B., Zhang, P., Jones, P. H., Zambreno, J., and Rozier, K. Y., "Embedding Online Runtime Verification for Fault Disambiguation on Robonaut2," *Proceedings of the 18th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, Lecture Notes in Computer Science (LNCS), Vol. TBD, Springer, Vienna, Austria, 2020, p. TBD. https://doi.org/TBD, URL http://research.temporallogic.org/papers/KZJZR20.pdf.

[46] Csete, A., "About gpredict," , Nov 2018. URL http://gpredict.oz9aec.net/.

[47] Asundi, S. A., and Fitz-Coy, N. G., "Design of command, data and telemetry handling system for a distributed computing architecture CubeSat," *2013 IEEE Aerospace Conference*, 2013, pp. 1–14.

[48] Rozier, K. Y., and Schumann, J., "R2U2: Tool Overview," *Proceedings of International Workshop on Competitions, Usability, Benchmarks, Evaluation, and Standardisation for Runtime Verification Tools (RV-CUBES)*, Vol. 3, Kalpa Publications, Seattle, WA, USA, 2017, pp. 138–156. https://doi.org/TBD, URL https://easychair.org/publications/paper/Vncw.