Day-in-the-Life Hardware-in-the-Loop Satellite Simulator for Mission Concept Verification

Luke Bedrosian
Department of Aerospace Engineering
Texas A&M University
College Station, TX, USA
lukebedrosian@tamu.edu

***Abstract*— This paper presents the development of a satellite simulation tool that simulates a "day in the life" in the operations of a satellite to help establish the feasibility of a mission concept during Pre-Phase A conceptual mission studies. The paper provides a thorough description of the simulation tool, which propagates a rich satellite state vector which, in addition to position and velocity (or orbital parameters), includes on/off state of significant components, power in/out, battery state of charge, eclipse, and data storage. The simulator includes an interface that executes the flight software at discrete events in the simulation (e.g., attitude changes) on realistic satellite hardware, which is not conventionally done until much later i.e. Phase B-C. This can help us identify issues with a mission conops earlier. In this case, the hardware used is the EyasSat GEN 5 Nanosatellite Simulator, and is controlled using COSMOS by Ball Aerospace. The constraints imposed by this hardware are only representative of very small satellites. The paper discusses simulation performance and its implications for facilitating feasibility checks in early mission concept studies.**

Keywords—Simulation, Satellite, Mission Planning, Design

I. INTRODUCTION

A simulation is an essential tool in the design and verification of space systems, and in the Pre-Phase A stage of the systems design process can serve as a feasibility check for proposed mission concepts [1]. During these conceptual studies, numerous tools may be deployed to evaluate the expedience and stakeholder satisfaction of proposed mission architectures. These tools include but are not limited to parametric sizing, rules-based expert systems, and Concept of Operations (ConOps) simulators [2, 3, 4]. While current ConOps simulations ascertain the efficacy of the "day in the life" operations of a satellite, current tools do not incorporate the hardware aspect of such designs beyond basic software modeling. Satellite simulators that include the hardware-in-the-loop are typically reserved for later phases in the design process for design testing and verification i.e., Phase D [5]. *The application of a hardware-in-the-loop satellite simulator that simulates day-in-the-life operations provides an additional level of fidelity in early mission feasibility checks.* Such a tool allows testing if existing software and hardware can be used for the mission or if new developments must be done. It also helps verify that certain mission requirements can be met for new distributed mission concepts that rely more on on board processing, autonomy, and communications between satellites.

The Pre-Phase A design stage is crucial in determining the chances of success of any given mission concept. During this phase, a wide array of design ideas and alternatives are proposed and weighed according to a variety of factors including stakeholder needs, cost, risk, etc [1]. Poor decision-making in this stage may lead to cost overruns, project delays, low stakeholder satisfaction, and even termination of the project. In fact, it has been shown that early design decision making is the most frequently occurring risk area on flight software cost risk [6].

The implementation of various software tools provide designers with different means of assessing proposed architectures and their merits. For example, parametric scaling relationships provide systems engineers with estimates of the main physical and economic aspects of the mission, including the system budgets (e.g., mass, power) and lifefycle cost and schedule [2, 7]. Rule-based systems such as VASSAR allow engineers to consider qualitative requirements and subjective preferences in addition to the quantitative and objective information captured by parametric models [3]. Cognitive assistants have also been proposed to help the user find good mission designs by integrating these models mentioned above, plus the ability to query various databases in natural language [8]. All these tools provide valuable information to designers that can help to make more educated decisions and reduce the probability of pursuing flawed or suboptimal designs.

Numerous simulation tools have been implemented as well that provide data for calculating various figures of merit and performance metrics. Such proposed tools include constellation instrument simulations [9], simulators that perform "what-if" scenarios [10], space station simulators [11], and a 4-module approach that tracks position/veclocity, equipment states, power levels, and memory states for daily operations [4]. These are only a few of the numerous software-only simulations that are currently used to verify the feasibility of a given mission design in the Pre-Phase A stage.

While using a purely mathematical model of the system allows for a good approximation of the performance of a design, some parts are historically difficult to model, e.g., actuators, decreasing the fidelity of the simulation [12]. Due to this decreased veracity, the introduction of hardware-in-the-loop in the simulation allows for higher-fidelity simulations and feasibility checks. The current literature on hardware-in-the-loop simulations for space systems focuses on the integration

and testing phase of the design process, with simulations relating to satellite attitude control, rendezvous and docking, and CubeSats [5, 13].

This paper seeks to bridge the gap by incorporating hardware-in-the-loop into Pre-Phase A simulations. Section II presents the developed simulator with both the software development and hardware used for this simulation. Section III goes through a case study of a "day in the life" of a satellite using our hardware-in-the-loop simulator. Finally, in Section IV we discuss the applicability of such a tool and future work to be done.

II. System Overview

A. Overall Architecture

The architecture of the hardware-in-the-loop simulator consists of four modules: the Agent module, the Orbit Propogator Module (poisiont, velocity, attitude, coverage data), the satellite hardware interface, and the satellite hardware. A diagram of the architecture is provided in Figure 1.

The simulator uses the concept of an Agent class, which is part of our lab's Distributed Multi-Agent Satellite System Simulation (DMAS) project developed by Alan Aguilar. This provides the user optionality to simulate satellite constellations with distributed architectures with hardware-in-the-loop. In this architecture, satellites, ground stations, and aircraft are all treated as Agents.

Each Agent has an Actor, a Scheduler, and a Platform Simulator. The Actor perceives information from its environment or from other agents and performs the actions given by the Scheduler. The Scheduler processes information perceived by the agent, maintains an internal knowledge base, and determines the next sequence of actions to be performed. The Platform Simulator tracks the current state of the agent and can turn off the Agent if actions exceed capabilities.

This approach provides the flexibility to run multiagent simulations with decentralized planning, where each agent is an independent actor, multiagent simulations with centralized planning, where there is a central agent that sends actions to the rest of the agents, or to run with a single agent i.e., no constellation. A benefit of including an onboard Agent is that one can configure the onboard Agent program to change its behavior based on what it perceives, as is the case with many modern satellite flight softwares. Additionally, it provides a framework for reinforcement learning with the flight software and onboard agent [14].

In addition to this Agent Class, the simulator draws upon the Orbit Propogator module for position, attitude and coverage data. The agent class also receives information from the hardware interface which can receive and send information from other agents, e.g. satellites, in the simulation. The Agent sends commands and updates to the hardware interface that may include executable flight software or commands to message other agents.

B. Software Development

The simulator is primarily written in Python and uses the SimPy package, a process-based discrete-event simulation framework. This package enables the user to simulate in real-time, "as fast as possible," or by manually stepping through discrete events [14]. The simulator provides flexibility to the user, with the option to run the satellite simulation in real-time, or to speed up the simulation until a discrete event with commands to be executed on the satellite hardware.

The simulator propagates the state of various parameters throughout the simulation including position, velocity, attitude, access/coverage, power systems, and communications/data handling. The information coming from Orbit Propogator module is a cartesian state vector of position and velocity, along with a Boolean value of whether the satellite is in an eclipse state or not. The Orbit Propogator Module uses OrbitPy, a software framework which facilitates the design of novel observation systems [15]. The coordinate system selected is the J2000 celesital coordinate system, although the user can change this if desired. Additionally, coverage metrics can be calculated on a coverage grid given the swath and scan characteristics of the simulated payload. These orbit and coverage calculations are pre-computed before the start of the simulation with an inputted start time, orbit, and simulation duration. The precomputation of orbit propagation and coverage does indicate, however, that performing orbit changes in the flight software is not currently supported. This is something that can be implemented in future work. The power and communications simulations can be performed in two ways, either entirely mathematically modeled, or using the state of the hardware-in-the-loop.

The mathematically-modeled state of the power and communications systems track the battery charge state, power out, power in (solar arrays), available data storage, and the amount of data transmitted. They use known characteristics to model the expected behavior of the systems for each interpolated interval. The use of the mathematically-modeled system states is useful when testing out specific flight software on individual systems, e.g. the attitude, determination, and control subsystem (ADCS). Additionally, it may be difficult to simulate the proper solar input for the solar panels when performing a hardware-in-the-loop simulation, so mathematical modeling is best suited when proper environments are not available.

Using the state of the hardware in the simulation produces the greatest fidelity of simulation this tool provides. The user can track the real time characteristics of the power in/out, available data storage, and the states of both the thermal and ADCS subsystems. An application of the hardware-in-the-loop simulator is expanded upon in Section III.

C. Hardware Components

For industrial applications, a majority of the hardware components will likely be readily available at the time of mission conception, especially for small satellites and CubeSats [16]. The application to CubeSats is evermore important with the increasing proportion of CubeSats being launched and the advent of CubeSat Comercial-Off-The-Shelf (COTS) components [17]. On account of this, likely, the majority of the hardware components being considered for the mission architecture in industrial use cases are readily available, and hence may be used as the hardware-in-the-loop for simulation.

Texas A&M Engineering Undergraduate Summer Research Grant

Since industrial use cases require the flexibility to incorporate their own hardware and flight software, which may be executed through different software, the simulator developed has the open flexibility of connecting to different APIs to interface with the satellite hardware. It is commonplace for companies to maintain their own catalog of components, either developed by them directly or used in past missions, so this API allows for easy integration of industrial components on hand. As further discussed in Section III, we applied this to the EyasSat GEN 5 Nanosatellite simulator, using an API to control it with COSMOS by Ball Aerospace.

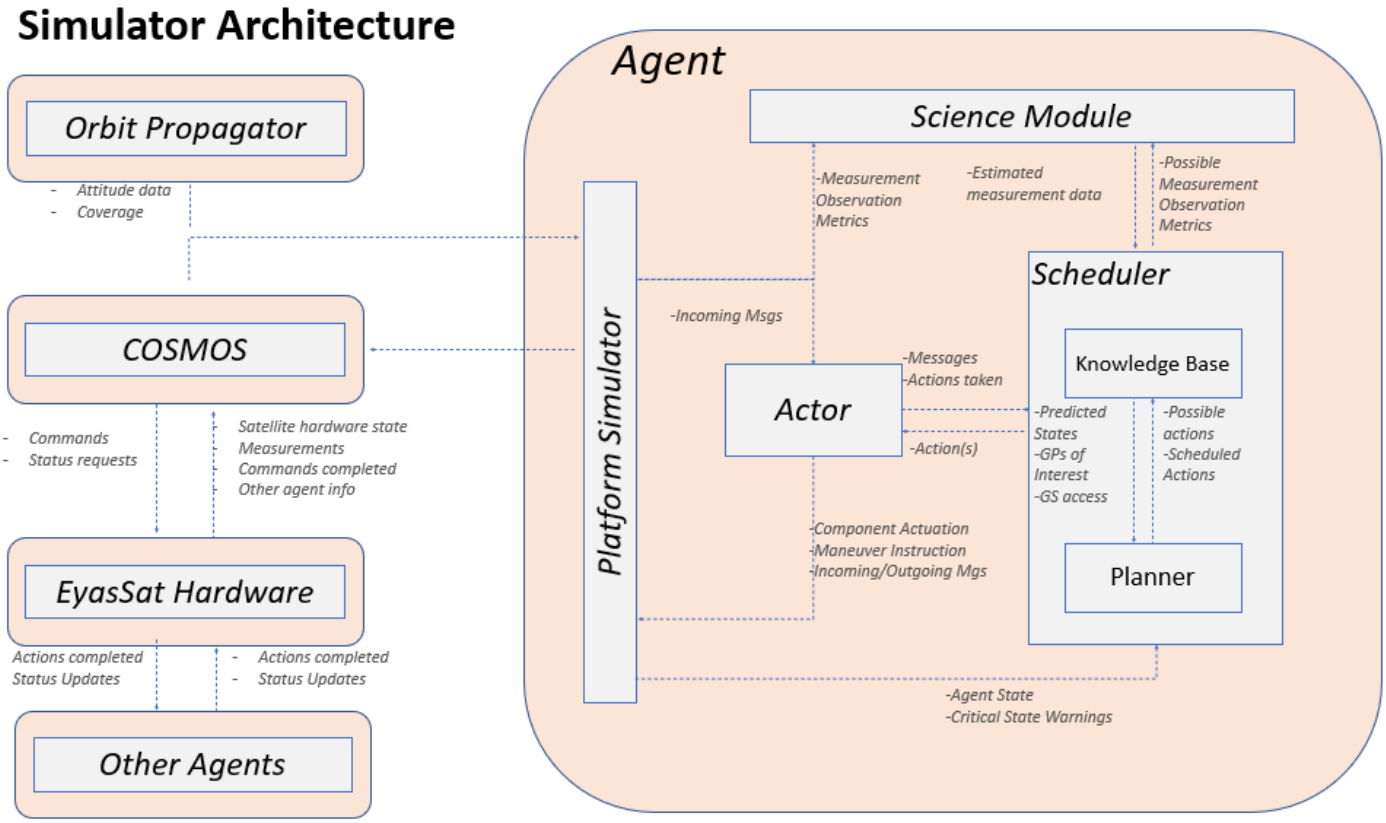

Figure 1 System Architecture of the hardware-in-the-loop simulator.

III. Case Study

To test the efficacy of the simulator developed, we applied it to the aforementioned EyasSat GEN 5 Nanosatellite [18], a prime example of readily available COTS hardware.

A. Hardware Specifications

A summary of the systems and components of the satellite is shown in Table 1. The EyasSat is controlled using the COSMOS Open Source Command-and-Control System built by Ball Aerospace [19]. This includes a GUI for user controls and data viewing, as well as a built-in JSON API. Communications between the simulator and the satellite hardware are conducted using the ballcosmos Python package, the official Python API for COSMOS. In COSMOS, the executable flight software commands are written in Ruby, and are then converted into command packets that are readable for the satellite components.

The EyasSat has four distinct boards that control the various subsystems of the satellite. These include the power distribution board, the ADCS board, and two data handling boards. The communications and thermal subsystems are controlled via the two data handling boards.

The electric power subsystem (EPS) is controlled by the power distribution board. This boards monitors and manages the various components of the EPS. The two main components within the power system are the battery, for energy storage, and the solar arrays, the energy source. The battery can be recharged either using the solar arrays, or by using an external 5.3 V charger. The solar array is body-mounted on one side of the satellite, with 6 solar cells. These can be run in two configurations, either 3 parallel 2 series, or 2 parallel 3 series.

The Attitude, Determination, and Control Subsystem (ADCS) board controls the various components of the ADCS. The EyasSat has three degrees of rotational freedom, with one reaction wheel for rotation about the z-axis, and two orthogonally placed magnetorquers for rotation about the x-axis and y-axis.

The Thermal subsystem is controlled by the data handling boards. It contains a thermal panel mounted opposite the solar array, which has two heaters beneath 2 plates, a copper rod, and heat pipe. It also contains temperature probes for sensing and maintaining proper temperature within the satellite.

The communications subsystem is controlled by the data handling board. It contains a transceiver and antenna that can

connect to the ground support equipment (GSE) antenna, which plugs in via USB to the GSE computer.

Lastly, the Command and Data Handling (CDH) subsystem is controlled by the two data handling boards, and contains the main CPU and data storage of the satellite.

Table 1 Summary of subsystem components included in the EyasSat GEN 5 Nanosatellite

Subsystem	Components
EPS	Power Distribution Board, Rechargeable battery, Solar Array
ADCS	ADCS Board, Reaction Wheel, Magnetorquers, Sun Sensors (Top, Bottom, Yaw)
Thermal	Thermal Array Panel with two heaters beneath two plates
COMM	Radio, Antenna
CDH	CPU, Data Handling Boards, Reference Thermosistor

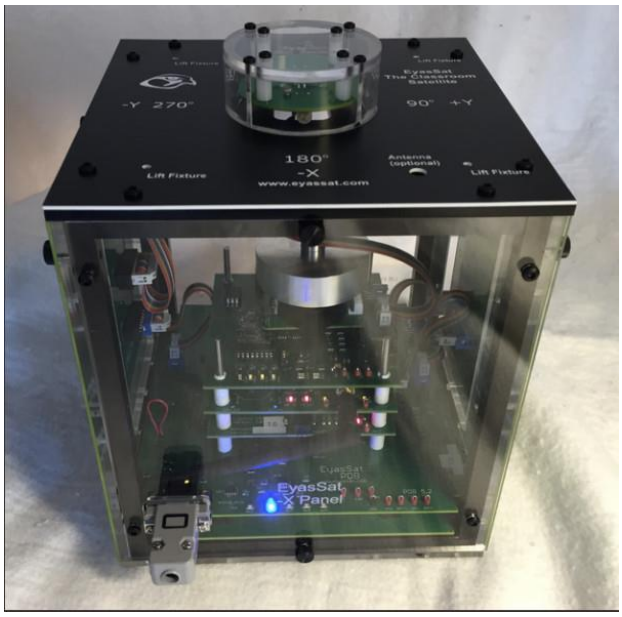

Figure 3 EyasSat GEN 5 Nanosatellite Simulator

B. Setup and Parameters

We modeled the simulation based off of existing CubeSat missions. With the exception of the payload, the EyasSat Nanosatellite closely resembles the bus for the BRITE constellations, so we opted to simulate the satellite with the same orbit as the BRITE-Toronto (ID: BTr) satellite. This operates in Low Earth Orbit at an altitude of 620-643 km, inclination of 97°, and a period of 97.1 minutes [20].

For the first runthrough of the simulation, the goal was to demonstrate the bus can sustain daily operations which include attitude pointing, power in and out, data handling, and communications. We essentially wanted to prove the satellite bus could handle the daily operations without a payload. In future work, a payload can be added to the EyasSat bus, or mathematically modeled in the simulation, however, due to time constraints this was not possible.

C. Simulation Performance

Due to a few unforeseen technical difficulties and workarounds paired with the time constraint of the deadline, we were unable to get a successful runthrough with the hardware-in-the-loop simulator. We expect that when run in a real-time simulation, the EyasSat GEN 5 should be able to reasonably execute the flight software and subsist during daily operations. The case study will be completed in future work.

IV. Discussion

This paper has presented a hardware-in-the-loop satellite simulator for Pre-Phase A mission planning. The simulator provides a feasibility check for proposed architectures with a higher fidelity than a purely mathematically-modeled simulator. The simulator has an intelligent agent "on-board" the satellite, which contains its own scheduler and actor. It communicates with the satellite hardware-in-the-loop through a hardware interface, which in our case study was the COSMOS interface. The simulator can be ran in real-time, accelerated time, or simply just run through executable discrete events. Due to time constraints, we were unable to complete our case study with data collection, but will do so in future work.

V. Future Work

Immediate future work demands a completed case study trial with the hardware-in-the-loop. As mentioned prior, due to the time constraints of the program, we were unable to complete the case study.

Beyond simply completing the case study, future work regarding hardware-in-the-loop simulations includes the addition of a payload to the satellite. In addition to simulating a payload, a mathematical model of the propulsion system may also be added to the hardware-in-the-loop simulation. This addition of the propulsion system would require that the simulator not pre-compute all attitude data, and that it computes the attitude with OrbitPy as the simulation goes along.

Acknowledgment

I thank Alan Aguilar of the Systems Engineering, Architecture, and Knowledge (SEAK) Lab at Texas A&M University for providing the structure of the DMAS Intelligent Agent architecture, and providing guidance on the matter.

References

[1] G. Shea, "3.3 Project Pre-Phase A: Concept Studies," *NASA*, 19-Apr-2019. [Online]. Available: https://www.nasa.gov/seh/3-3-project-pre-phase-a-concept-studies. [Accessed: 25-Jul-2022].

[2] P. N. Springmann and O. L. de Weck, "Parametric scaling model for nongeosynchronous communications satellites," *Journal of Spacecraft and Rockets*, vol. 41, no. 3, pp. 472–477, 2004.

[3] D. Selva and E. F. Crawley, “Vassar: Value Assessment of system architectures using rules,” *2013 IEEE Aerospace Conference*, 2013.

[4] Chagas, Ronan & Louro, Arcélio & Souza, Fabiano & Gomes dos Santos, Willer. (2016). Satellite Simulator for Verification of Mission Operational Concepts in Pre-Phase A Studies.

[5] R. Rodrigues, A. Murilo, R. V. Lopes, and L. C. Souza, “Hardware in the loop simulation for model predictive control applied to satellite attitude control,” *IEEE Access*, vol. 7, pp. 157401–157416, 2019.

[6] J. M. Hihn, K. Lum, and E. Monson, “Organizational structure impacts flight software cost risk,” *Journal of Cost Analysis and Parametrics*, vol. 2, no. 1, pp. 23–36, 2009.

[7] W. J. Larson and J. R. Wertz, *Space mission analysis and design*. Germany: Springer-science, 1992.

[8] H. Bang, A. Virós Martin, A. Prat, and D. Selva, “Daphne: An intelligent assistant for Architecting Earth Observing Satellite Systems,” *2018 AIAA Information Systems-AIAA Infotech @ Aerospace*, 2018.

[9] S. Nag, V. Ravindra, and J. L. Moigne, “Instrument modeling concepts for TRADESPACE analysis of satellite constellations,” *2018 IEEE SENSORS*, 2018.

[10] X. Cyril, “ROSESAT -- A Graphical Spacecraft Simulator for Rapid Prototyping ,” *SPACE OPS Symposium*, 1998.

[11] K. Wang, B. Zhang, and T. Xing, “Preliminary integrated analysis for modeling and optimizing space stations at Conceptual Level,” *Aerospace Science and Technology*, vol. 71, pp. 420–431, 2017.

[12] M. Bacic, “On hardware-in-the-loop simulation,” Proceedings of the 44th IEEE Conference on Decision and Control.

[13] S. Corpino and F. Stesina, “Verification of a CubeSat via hardware-in-the-loop simulation,” *IEEE Transactions on Aerospace and Electronic Systems*, vol. 50, no. 4, pp. 2807–2818, 2014.

[14] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. Cambridge, MA: The MIT Press, 2020.

[15] V. Castillo, “Parallel simulations of manufacturing processing using Simpy, a python-based discrete event Simulation Tool,” *Proceedings of the 2006 Winter Simulation Conference*, 2006.

[16] V. Ravindra, R. Ketzner, and S. Nag, “Earth observation simulator (EO-SIM): An open-source software for observation systems design,” *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*, 2021.

[17] L. C. Chang and A. Chandran, “Preface: Advances in small satellites for space science,” *Advances in Space Research*, vol. 66, no. 1, pp. 1–2, 2020.

[18] D. Selva and D. Krejci, “A survey and assessment of the capabilities of Cubesats for Earth Observation,” *Acta Astronautica*, vol. 74, pp. 50–68, 2012.

[19] O. Ritchey, J. Clark, J. White, T. White, D. Barnhart, and J. Sellers, “Eyassat: Transforming the way students experience space systems engineering,” *2004 Annual Conference Proceedings*.

[20] R. Melton and J. Thomas, “A Cloud-Based, Open-Source, Command-and-Control Software Paradigm for Space Situational Awareness,” *Advanced Maui Optical and Space Surveillance (AMOS) Technologies Conference*, 2017.

[21] Pablo, H., Whittaker, G. N., Popowicz, A., Mochnacki, S. M., Kuschnig, R., Grant, C. C., Moffat, A. F., Rucinski, S. M., Matthews, J. M., Schwarzenberg-Czerny, A., Handler, G., Weiss, W. W., Baade, D., Wade, G. A., Zocłońska, E., Ramiaramanantsoa, T., Unterberger, M., Zwintz, K., Pigulski, A., … Zee, R. E. (2016). The BRITE constellation nanosatellite mission: Testing, commissioning, and Operations. *Publications of the Astronomical Society of the Pacific*, *128*(970), 125001. https://doi.org/10.1088/1538-3873/128/970/125001