Simplifying Operational Scenario Simulation for CubeSat Mission Analysis Purposes

Pedro Ângelo Vaz de Carvalho*
pedroangelovaz@gmail.com
National Institute for Space Research
São Paulo, Brasil

André Aparecido de Souza ivo
andre.ivo@cemaden.gov.br
National Institute for Space Research
São Paulo, Brasil

Guilherme Venticinque
guilherme.venticinque@inpe.br
National Institute for Space Research
São Paulo, Brasil

Gustavo Vicari Duarte
vicariduarte@gmail.com
National Institute for Space Research
São Paulo, Brasil

Matheus Henrique de Abreu Miranda
matheus.ham@outlook.com
National Institute for Space Research
São Paulo, Brasil

Fatima Mattiello-Francisco
fatima.mattiello@inpe.br
National Institute for Space Research
São Paulo, Brasil

ABSTRACT

The low budget and tight schedule of many CubeSat educational projects make important systems engineering practices unfeasible due to the high cost of the computational infrastructure and commercial software tools, which require long setup and learning times. Simulation of space mission operational scenarios is one of the best practices recommended for system engineers to get familiarity with the expected behavior of the satellite in orbit. This work presents the results of the use of Atom SysVAP, an alternative for mission analysis studies in the context of concept of operation of CubeSat-based space projects. By building their simulation environment on tools like Atom, designers have the flexibility to add complexity to the model according to necessity and achieve reliable initial results in a costless and swift manner.

CCS CONCEPTS

• **Computing methodologies** → **Model verification and validation**; *Modeling methodologies*; *Simulation types and techniques*; *Simulation tools*.

KEYWORDS

CubeSat, Concept of operation, Simulation, Memory simulation, Power simulation, Mission analysis

ACM Reference Format:
Pedro Ângelo Vaz de Carvalho, André Aparecido de Souza ivo, Guilherme Venticinque, Gustavo Vicari Duarte, Matheus Henrique de Abreu Miranda, and Fatima Mattiello-Francisco. 2022. Simplifying Operational Scenario Simulation for CubeSat Mission Analysis Purposes. In *11th Latin-American Symposium on Dependable Computing - 3rd Workshop on validation and verification of Future cyber-physical systems, November 21–24, 2022, Fortaleza/CE, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3569902.3570189

*Corresponding author.

LADC-WAFERS '22, November 21–24, 2022, Fortaleza/CE, Brazil

ACM ISBN 978-1-4503-9737-7/22/11…$15.00
https://doi.org/10.1145/3569902.3570189

1 INTRODUCTION

In recent years, one can observe an exponential growth in the number of CubeSats. Initially introduced for educational purposes, the growth of CubeSat projects has been supported by the various initiatives of both universities and companies to provide solutions in products, services, and tools that attract more and more interest in the sector, as depicted in Figure 1. This quick advance was possible because of reduced development time and cost, allied with technological evolution such as miniaturization of components and subsystems, and interface standardization, with the potential to perform scientific and commercial missions [6]. But the availability of standardized solutions and rapid prototyping might mistakenly suggest that important stages of development, particularly the clear and concise definition of the mission and thus the elicitation of complete technical requirements, may be skipped. The lack of attention to the elicitation of appropriated technical requirements is one of the causes of CubeSat failure in orbit. Even with the components' reliability increasing over time due to technology maturation [8], the use of systems engineering (SE) practices remains a key and necessary factor for mission success [1].

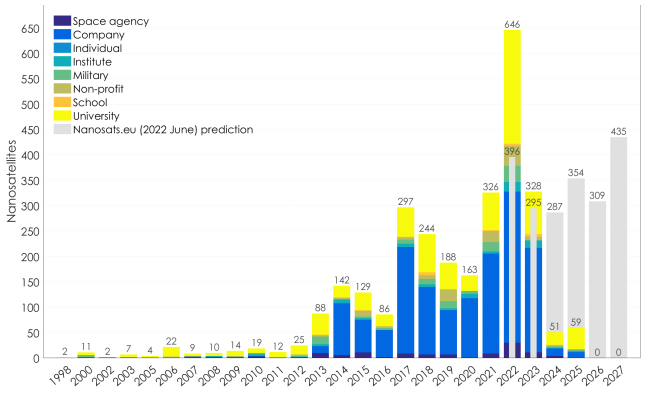

Figure 1: Nanosat Launches by organization. Source: Erik Kulu [5]

Systems engineering methodologies usually adopted in space missions are burdensome and costly, adopting commercial software and high workload processes that would make a CubeSat

project prohibitively expensive. Therefore, efforts on tailored SE practices have been observed in recent literature, focusing on inexpensive modeling and simulation while maintaining desired reliability. Examples of this development can be seen in [7] and [3], where CubeSats are inserted in model-based systems engineering environments.

Simulation of operational scenarios is a critical factor in the decision-making on mission analysis of a project [3]. Evidence of failure in operational scenario simulation early in the project anticipates the designer's understanding regarding resource constraints and the expected behavior of the communicating subsystems. This understanding can prevent engineers from realizing the insufficiency of common resources only later in the project, like power generation or data transmission, when design rework is more expensive and often unfeasible. Furthermore, an overestimation of these parameters can also be avoided, which would result in increasing the cost and weight of the system.

Therefore, simulators are largely used in the concept analysis of space missions for the context of verification and validation of mission requirements and the mission concept of operation. Through their use, designers can certify that the proposed solution satisfies user requirements, where expected operation scenarios would not lead to undesirable results after project realization. In the early stages, simulators are commonly used to assess power and data balance on the satellite, while the following phases might require further assessment, such as orbit, thermal and structural simulations, as well as refined results from power and data.

Most of the simulators available on the market have high license costs, slow learning curves, and a long time required for simulation setup. Therefore, CubeSat projects, especially educational ones, have the challenge of finding cheap and reliable alternatives for the mission concept analysis.

This paper presents the use of a state machine simulation software, called Atom SysVAP, to implement a costless, fast, and reliable simulation environment for CubeSats. The results are compared with the professional simulation tool ForPlan, INPE's proprietary simulator, highlighting differences and improvement possibilities for each tool. The results will give further insights into the simplification of operational scenario simulation for CubeSat-based mission, improving accessibility into an important verification and validation tool, following the partnership with ADVANCE[1]

The remainder of this paper is organized as follows: Section 2 (context) presents the tools Atom SysVAP and ForPlan with their purposes and uses, as well as a brief description of the NanoSatC-Br2 space segment simulated in both software. Section 3 (methodology) shows part of the simulation code in Atom and presents the simulation setup and operational scenarios used on both tools. Finally, section 4 presents the simulation results and the discussion comparing both tools and their uses and potential.

2 CONTEXT

For the development of the simulations, a basic knowledge of the used tools Atom SysVAP and ForPlan is necessary, as well as the modeled mission NanoSatC-Br2

[1]https://www.advance-rise.eu

2.1 Atom SysVAP

Atom SysVAP[2] is an open source software, protected by GNU General Public License v3.0. It is a simulation and development tool that allows for the representation of Mealy and Moore state machines. The software was built on *Java*, and the state machine models within the project can be coded with the language *Lua* [**?**].

The tool works as a general-purpose state machine simulator with an integrated coding environment (IDE). Despite its general purpose characteristic, the implementation of the satellite concept of operation simulations can be performed with a fast learning curve, especially with the examples created by the author and made available with the tool distribution. The use of the tool for satellite flight plan simulation was explored in [**?**].

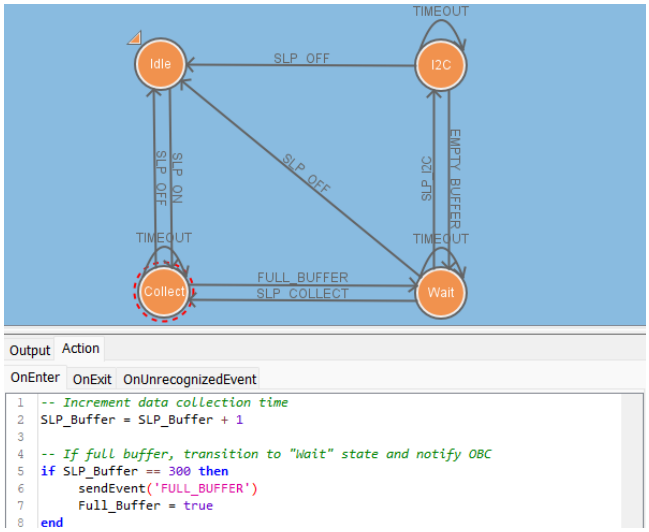

Figure 2: Example of Atom's state machine model for an SLP. Source: Authors

Figure 2 illustrates a state machine representation of a Langmuir Probe (described in section 2.3), with the following states: Idle, collecting measurements and saving data to buffer, waiting for onboard computer (OBC) commands to either start collecting data again or to transfer data to OBC, and the I2C data transfer mode. In each of the states, it is possible to insert code that will run on entering, exiting or unrecognized event in the state. The code from all parts of a project shares variables, making it possible for triggering events in any machine or state to share data among them as global variables. The state changes happen through a "sendEvent" command programmatically or via user manual input.

Furthermore, it is also possible to loop a state's code with a self-referencing transition, represented in "TIMEOUT". this functions as a "while loop" repeatedly running the "OnEnter" and "OnExit" codes, and the exit command is inside the code when a particular condition is true, transitioning to another state. It is possible to insert a timeout in the transition. In this example, the code loops according to processor capacity, with the time increment already present in the variable "SLP_Buffer".

[2]https://github.com/andreivo/Atom

Another advantage of the state machine simulator is that it is possible to simulate not only the components of the satellite but also the environmental factors affecting its behavior in orbit. Examples are the solar energy generated, the visibility of the ground station, and the passage through the region of interest (ROI), all automatically synchronized by the orbit propagation, which is also possible to include in the code.

A detailed explanation of all the tool functionalities and possibilities is presented in [**?**].

2.2 ForPlan

ForPlan Satellite Simulator is a software designed at INPE for the simulation of satellite concepts, especially used at CPRIME for pre-phase A studies [2]. The software is registered at brazilian's National Institute of Industrial Property (INPI) under the process number *BR 51 2020 000534-9*.

The simulator, based on *Julia*'s library SatelliteToolbox[3] from the same author, presents an environment for the analysis of mission concepts, with a focus on power and data balance, evaluating instruments operation parameters, while generating power in sunlight and unloading data in ground stations visibility regions [3].

The simulation setup requires basic knowledge in *Julia* programming language since it does not have a graphical interface for this purpose. The configuration script includes several parameters, like an equipment list with power and data, battery pack definition, ground stations, orbit definitions, and functions that govern instrument operation.

2.3 NanoSatC-Br2 Mission

The NanoSatC-Br2 mission is part of the NanoSatC-Br program at INPE, a program integrated mainly by students and professors from UFSM and INPE, with educational, scientific, and technological goals. The space segment is accomplished with a 2U CubeSat carrying a sweeping Langmuir Probe and a magnetometer to study the South Atlantic Magnetic Anomaly, a communication store forward experiment, not modeled in the simulation discussed herein, and two radiation tolerant circuits and sensor.

The Langmuir Probe is a sensor capable of measuring electron temperature and density, and the plasma density in the ionosphere. It is used to study this atmospheric layer, which is greatly affected by the magnetic anomaly.

The magnetometer provides direct measurements of the magnetic field on the anomaly and provides information about the satellite's attitude.

The fault-tolerant attitude determination system (SDATF) is a scientific and technological payload, which works with data from magnetometers, sun sensors, and orbital parameters. It works with three micro-controllers for redundancy purposes and since it is a newly developed system, it is meant to be validated in flight conditions.

The SMDH payload consists of radiation-tolerant field programmable gate arrays and application-specific integrated circuits also being validated onboard NanoSatC-Br2.

AMSAT-BR is a payload capable of storing and replicating messages from amateur radios. It was not included in the modeling, since it is not one of the main payloads and it is not present in the model used as a starting point for the simulations.

[3]https://github.com/JuliaSpace/SatelliteToolbox.jl

3 METHODOLOGY

The ultimate goal of this work is to compare the simulation results done on a simple state machine developed in the Atom SysVAP with a similar simulation of operational scenarios configured in the professional environment ForPlan.

The simulation in ForPlan was configured based on previous work [3], although some parameter modifications in the input file and adjustments in the software source code to accept region of interest detection functions were necessary.

In Atom, the simulation was built on top of a power balance model from the software author, with the addition of the orbit and data balance simulation, region of interest, and ground station detection, as well as other details relative to the satellite scenario.

3.1 Simulation code

The simulation code in Atom was developed in state machines representing each relevant onboard component, the transition between sunlight and eclipse, and the internal clock looping in a timeout transition and running the main section of the code. In accordance with the simplicity of the model, the state machines for the instruments were developed with only two states, on and off, with the mean power consumption and data generation. The receiver, OBC, and electric power system are always on, while the transmitter will decrease internal memory during ground station visibility.

The extracts of the code below outline the simulation approach:

```
-- operation times
SMDH_on, SMDH_off = T_orbital * 10, T_orbital * 6
SDATF_on, SDATF_off = T_orbital * 1, T_orbital * 15
-- Simulation scenario
SMDH_always, SDATF_always = false, false
SLP_always, Mag_always = false, false
```

The operation times define the timed schedule for SMDH and SDATF. The flags in the simulation scenario section determine whether the instruments will remain continuously active or follow the predetermined operation schedule and functions.

```
---- Orbit Propagation ----
-- Update Latitude
Sat_lat = Sat_lat + Sat_speed
if Sat_lat > 90 then
    Sat_lat = 180 - Sat_lat
    Sat_speed = - Sat_speed
    Sat_long = Sat_long - 180
end
if Sat_lat < -90 then
    Sat_lat = -180 -Sat_lat
    Sat_speed = - Sat_speed
    Sat_long = Sat_long - 180
end
-- Update Longitude
Sat_long = Sat_long - earth_speed
if Sat_long < -180 then
```

```
    Sat_long = Sat_long + 360
end
```

The simplified orbit represents a circular 90° inclination orbit, which allows for uncoupling of the longitudinal movement, caused by the earth's rotation, and the latitudinal motion by the satellite speed. This simplification reduces the time for coding and execution while maintaining a similar path to ForPlan results at 97.9° inclination. The extracts of the code below outline the simulation approach.

```
-- Turn on mag and slp on ROI
if Br2_Mag_state~=nil then
    if (not (Mag_always and SLP_always)) then
        -- sunlight is illuminating the system
        if Br2_Mag_state:getName() == 'Off' then
            if onROI() then
                if not Mag_always then
                    sendEvent('MAG_ON')
                end
                if not SLP_always then
                    sendEvent('SLP_ON')
                end
            end
        else
            if not onROI() then
                if not Mag_always then
                    sendEvent('MAG_OFF')
                end
                if not SLP_always then
                    sendEvent('SLP_OFF')
                end
            end
        end
    end
end
```

The sequence above presents how the instrument SLP and magnetometers are turned on and off based on the squared region of interest. A function is also in place to detect ground station visibility to activate the transmitter.

Finally, the internal memory and battery level of the satellite are incremented or decremented based on equipment operation, ground station visibility, and sun exposure. The simulation emits a warning message and stops execution if the battery reaches the minimum depth of discharge or if the memory is full.

The simulation code is available with the tool's distribution under the name "Br2_sim_orbit.vap". Alternatively, the Nanosat-br2 sub-folder[4] contains the standalone file for download.

3.2 Simulation Scenarios

The simulations implemented in Atom were carried out for three operation scenarios, seeking a solution where battery levels and internal memory remained stable after long periods of operation. This approach also shows the importance of such simulations, in which an operation scenario with the battery discharging below the minimum depth of discharge or the internal memory saturating, violates operational requirements and would lead to mission failure. Hence, serving as feedback to the development team if modifications in the operation schedule or even on the instruments and solar panels are necessary.

Since the main goal of this study is to evaluate the use of simple models for the simulation, and the model built in Atom can be as complete as the programmer is willing to code, two orbital models were used for comparison purposes. The first model, easier to code, uses a simplified and unrealistic polar orbit, where the longitudinal and latitudinal components are uncoupled and the modeling is very simple. The second model was the same sun-synchronous orbit mathematical model used in ForPlan simulations, with an inclination of 97.9°.

Finally, the results from the simulation of the final operation scenario, with stable battery and memory levels, were compared for the two orbit models in Atom, and the results from ForPlan. The differences are discussed together with the potential for each tool and model used. The flexibility of the simulations for new functionalities, the learning curve for simulation building and setup, and costs were considered in the comparisons.

The mean data generation and power consumption for each instrument were obtained from the project documents and are used in the simulation as presented in Table 1. The power generation during sun exposure was used as constant and equal to $3.7w$ with a conservative calculation from the solar panel's data. The downlink rate when in the ground station range is $4800bit/s$ and the available internal memory of $100Mb$ is purposefully smaller than in the real case to improve the visualization of its usage.

Payload	Data generation [bit/s]	Power use [mW]
Magnetometer	96	16
SLP	800	800
SDATF	512	264
SMDH	5	1092
OBC	5	383
Transmitter	-	1078
Receiver	-	193

Table 1: Instruments power and data parameters

4 RESULTS AND DISCUSSION

This section presents the results of the simulations performed in the Atom and ForPlan tools, as well as a discussion of the results and a comparison of the simulators.

As a first step, consecutive operation scenarios were simulated using the refined orbit version of the Atom model, in a search for maintaining stable levels of battery and internal memory. The first scenario consists of all instruments powered on during the whole operation of the satellite. In the second scenario, only the SDATF worked in continuous operation, while the other instruments follows the scheduling from Table 2. Finally, all instruments were modeled to follow the operation schedule.

It is possible to see from Figures 3 and 4 that the first scenario is unsustainable, in which the battery would discharge completely

[4]https://github.com/andreivo/Atom/blob/master/binaries/Examples/Nanosat-br2/Br2_sim_orbit.vap

Payload	Operation	Off Time	On Time
Magnetometer	ROI		
SLP	ROI		
SDATF	Pre-established	15 orbits	1 orbit
SMDH	Pre-established	6 orbits	10 orbits

Table 2: Payloads Operation Times

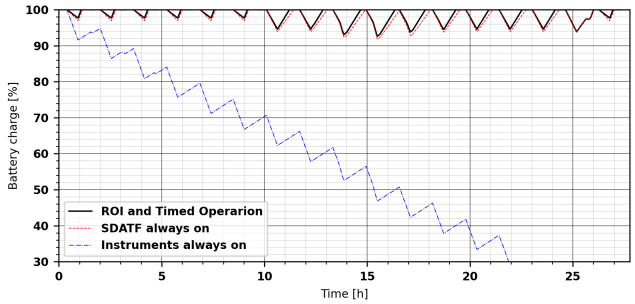

Figure 3: Battery Usage by Scenario.

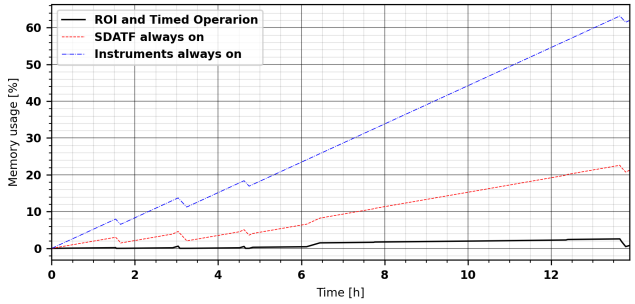

Figure 4: Memory Usage by Scenario.

and the memory would overflow. The second scenario attempted to keep the attitude determination experiment always on but was also not possible due to the data generation from this payload. Finally, It was possible to keep both power and data in balance with the SLP and magnetometers activation only over the SAMA region of interest and a timed operation for the other two payloads.

The effects of the payloads and ground station visibility on the memory balance of the spacecraft can be seen in Figure 5. It is possible to visualize the necessity of SDATF intermittent operation, due to its large data generation when powered on.

Finally, the results from both Atom simulations and ForPlan are compared side by side, for a better assessment of the differences. The orbit on each simulation can be seen in Figure 6. While the simplified unrealistic orbit, with straight lines, greatly differs from the other coincident orbits for large latitudes, it can be seen that the rate at which the satellite passes over each region is similar for all simulations, which is ultimately what causes instrument operation and ground station overpasses.

The battery and memory balance for each simulation tool and model can be seen in Figures 7 and 8.

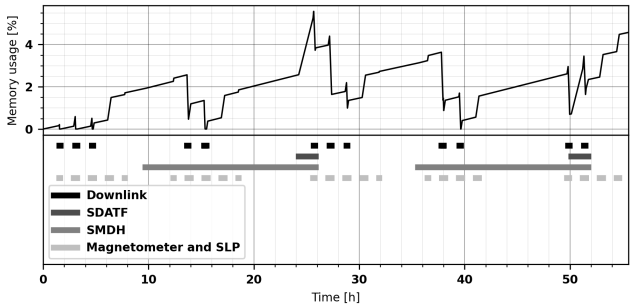

Figure 5: Payloads effect on internal memory.

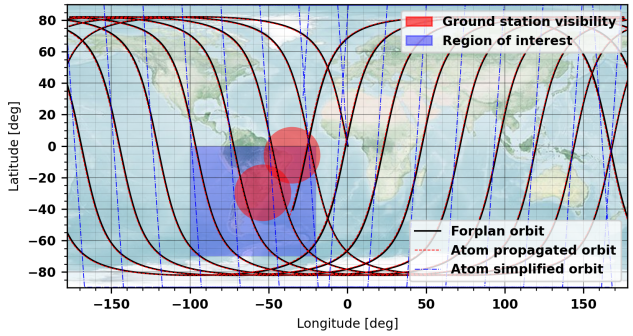

Figure 6: Orbit comparison for each simulation.

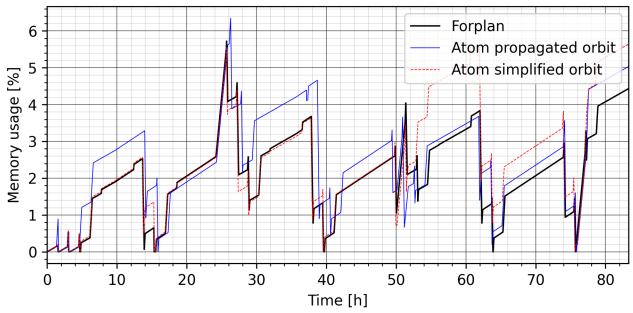

Figure 7: Memory balance for each simulation tool and model.

The results show little difference in the battery balance over time. The memory has significant differences in specific points, mostly due to small differences in orbit and instrument activation between the models. However, in both cases, the general trend for the internal memory and battery levels over larger periods is similar in all simulation models.

Pre-phase A of a project consists of early analysis of operation scenarios and concept analysis of the desired instruments and architecture working together. At this stage, even keeping the same general design, minor modifications up to the final phases will lead to simulation results different than the initial ones. Therefore, exact results from the memory and battery are not necessary nor

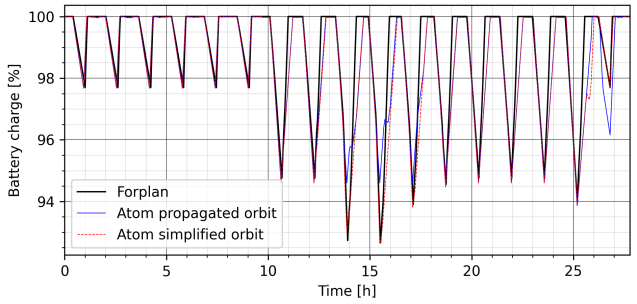

Figure 8: Battery balance for each simulation tool and model.

representative of the final design, and the important aspect of the simulation is that the battery will not be depleted or the memory saturated over time.

Thus, despite the differences between the simulations, all three results would be valid for the analysis of a concept of operation of a satellite. This shows that, while more complete and complex simulators are necessary for later steps of the project, even a simplified simulator, coded by the engineers of the project, can be used with confidence for initial simulations.

Some advantages can be observed in using such simplified simulators in this stage of the project. For one, many satellite simulators usually have a high license cost which alone could hinder a CubeSat project unfeasible, or even unavailable for general public use, as is the case of ForPlan at the current time. Furthermore, INPE's software currently only has Linux distributions, while Atom SysVAP can be used on all platforms.

Simple simulators, even one coded by the project engineers, can also reduce the work time for their implementation. Atom, for instance, is a tool with a fast learning curve, using an intuitive programming language as interface, while ForPlan uses a configuration file in *Julia* that can take a long time for the user to master. Such difficulty was even explored by Danilo et al. [3], who uses a model transformation to generate the configuration file starting from a Model-Based Systems Engineering approach with a graphical interface.

Furthermore, simple, open-source simulators, are more flexible, which allows the systems engineers to implement desired functions that even costly simulators might not have available. In this case, the simulations in ForPlan required source code modification to perform region of interest detection for payload scheduling and to output a file with the results of the simulation, which was only possible because of the previous experience of the systems engineers on the tool code. In Atom, the implementation of these functions was directly done during the construction of the model.

5 CONCLUSION

From the simulation results, it is possible to conclude that the use of simple simulation environments, in this case, implemented in Atom, has the potential to comply with the project's expectations, even surpassing complete simulators in some aspects: Low cost and learning time, high flexibility and simplicity of such simulations allow for an ideal tool in the concept phase, especially for educational and low-budget, space projects.

The flexible nature of the open source and self-implemented simulators allow for easy modification and addition of new features that would otherwise not be possible. The complexity and completeness of the model can also be increased according to desired accuracy up to a certain point while remaining cheaper and simpler than commercial software.

The choice of the simulator depends on several factors, like the precision required, the stage and complexity of the project, and the available budget. As a general rule, no tool will be essentially better than the other. It is up to the engineers to weigh the benefits and drawbacks, deciding on the most suitable option for each stage and need of the project. Starting simple and increasing complexity as necessary, especially on educational projects, is usually a promising path.

ACKNOWLEDGMENTS

We thank CAPES and INPE for the resources and infrastructure that made this work possible.

REFERENCES

[1] Sharan A Asundi and Norman G Fitz-Coy. 2013. CubeSat mission design based on a systems engineering approach. In *2013 IEEE Aerospace Conference*. IEEE, 1–9.
[2] Ronan AJ Chagas, Fabiano L de Sousa, Arcélio C Louro, and Willer G dos Santos. 2019. Modeling and design of a multidisciplinary simulator of the concept of operations for space mission pre-phase A studies. *Concurrent Engineering* 27, 1 (2019), 28–39. https://doi.org/10.1177/1063293X18804006 arXiv:https://doi.org/10.1177/1063293X18804006
[3] Danilo Pallamin de Almeida, Bence Graics, Ronan Arraes Jardim Chagas, Fabiano Luis de Sousa, and Fatima Mattiello-Francisco. 2021. Towards Simulation of CubeSat Operational Scenarios under a Cyber-Physical Systems View. In *2021 10th Latin-American Symposium on Dependable Computing (LADC)*. IEEE, 1–4.
[4] Jandreivo André Aparecido de Souza Ivo. [n. d.]. UMA FERRAMENTA PARA AVALIAÇÃO DE PLANOS DE VOO DE SATÉLITES USANDO MODELOS DE ESTADOS. ([n. d.]).
[5] Erik Kulu. 2022. Nanosatellite Launch Forecasts-Track Record and Latest Prediction. (2022).
[6] Armen Poghosyan and Alessandro Golkar. 2017. CubeSat evolution: Analyzing CubeSat capabilities for conducting science missions. *Progress in Aerospace Sciences* 88 (2017), 59–83.
[7] Sara C Spangelo, David Kaslow, Chris Delp, Bjorn Cole, Louise Anderson, Elyse Fosse, Brett Sam Gilbert, Leo Hartman, Theodore Kahn, and James Cutler. 2012. Applying model based systems engineering (MBSE) to a standard CubeSat. In *2012 IEEE aerospace conference*. IEEE, 1–20.
[8] Thyrso Villela, Cesar A Costa, Alessandra M Brandão, Fernando T Bueno, and Rodrigo Leonardi. 2019. Towards the thousandth CubeSat: A statistical overview. *International Journal of Aerospace Engineering* 2019 (2019).