Nanospace and Open-source Tools for CubeSat Preliminary Design: Review and Pedagogical Use-case

Thibault Gateau[1], Sophia Salas Cordero[2],Jérôme Puech[3],Rob Vingerhoeds[4]

Abstract
This paper aims to facilitate getting acquainted with CubeSat preliminary design by presenting a review of open-source tools commonly used during project first steps, and a concrete example. The light but realistic preliminary design framework is based on a real 3U CubeSat use-case, the CREME project, relying on Nanospace and a package of selected Open-Source tools. This example should allow students and non-related field experts to fully grasp the concepts needed to achieve the basics of a typical preliminary design.

Keywords
Preliminary design, CubeSats, Softwares

[1]Corresponding author: ISAE-SUPAERO, Université de Toulouse, France, thibault.gateau@isae-supaero.fr
[2] ISAE-SUPAERO, Université de Toulouse, France
[3] ISAE-SUPAERO, Université de Toulouse, France
[4] ISAE-SUPAERO, Université de Toulouse, France

1. Introduction

The preliminary design phase is one of the ini- tial steps in a CubeSat project (Phase 0/A in ESA nomenclature). Preliminary design is an iterative and a multidisciplinary process. It involves many disci- plines such as orbital mechanics, telecommunication, electronics, mechanical engineering, control theory, and thermal engineering. Therefore interdisciplinary exchange of information is essential to attain a pre- liminary design. System engineers normally lead the process for the different disciplines to work together, but without the aid of tools to enable this, it becomes a very time consuming task. While many ways of how to proceed with spacecraft preliminary designs have been proposed, such as Concurrent Design Engineering [1], few really propose an Open-Source set of tools and a method to attain a preliminary design. This has been identified as a gap in students' training during the Nanostar project. Nanostar has led to the development of a consistent set of tools for achieving a CubeSat preliminary design. Nanospace started to be developed in the context of the Nanostar project. Nanospace is an open-source software dedicated to facilitate the preliminary design of CubeSats for re- mote teamwork.In an academic context, open-source approaches have many reasons to be applied in CubeSat projects [1].

A full package of Open-Source software is proposed as an operational Nanospace Constellation, along with the iterative process methodology to manage them in order to achieve a simple but realistic prelim- inary design of a 3U CubeSat. Note that the process is iterative and modular, meaning that users can sim- ply deploy their own tools (maybe replacing the ones proposed) in the process to get a deeper analysis of expertise.

Generally, a team proposing a preliminary Cube- Sat design for a specific space mission should obtain:

- ☐ a mass budget
- ☐ a power budget
- ☐ a link budget
- ☐ a data budget
- ☐ a dissipation budget
- ☐ a radiation budget
- ☐ sometimes a propellant budget
- ☐ a mechanical architecture
- ☐ a thermal architecture
- ☐ an ADCS sizing
- ☐ an activity profile
- ☐ a launcher restriction choice
- ☐ a check for compliance with space laws
- ☐ a check for payloads constraints

In the following sections, first, a brieft overview of existing open-source software that allows part of a CubeSat preliminary design and related work on concurrent design engineering tools is presented. Afterwards, a use-case scenario is explained, based on a lighten preliminary design of CREME project 3U CubeSat.

2. State of the art

2.1. Existing software

Preliminary design software is a very dynamic ecosystem. This section does not aim to be exhaustive ally used during preliminary design phase. This ar- ticle does not aim to cover proprietary solutions. However, most the software mentioned here is con- sidered as "academic-friendly" for sake of pragmatism. Many open-source software, methodologies and recommendations can be easily found online. This includes the Libre Space Foundation initiative, full open-source CubeSat projects such as the UPSAT initiative, FloripaSat-I [2], and educational projects [3]. In addition the "open-source Satellite" initiative provides a first list of teams and software of the open- Below we describe different software that can be used for CubeSat preliminary design. We have clas- sified them based on their purpose: mission analysis, link budget, data budget, mechanical architecture, energy and power system (EPS), attitude determi- nation and control System (ADCS), end-of-life disposal, dissipation budget, and radiation budget. A list of the different data inputs and outputs usually taken into account during the process is also included. Inputs and outputs are illustrative and can vary de- pending each project. They may be obtained through design iterations, and may be dependent to internal processes. For example, a list of visibility windows of the ground station can be provided by the mission analysis to the module computing the data budget. Another possible option would be to provide Keple- rian parameters and ground stations location to the module computing the data budget.

2.2. Mission Analysis

Mission analysis usually consists of orbit definition, surface coverage, visibility windows, eclipse calculation.

Inputs

- ▲ Orbital parameters
- ▲ List of ground stations
- ▲ Satellites interlinks
- ▲ 3D structural model
- ▲ Ephemeris*
- ▲ Attitude profile*

Outputs

- ▼ Mission time availability
- ▼ Visibility windows
- ▼ Received solar flux
- ▼ Eclipses
- ▼ Activity profile

* Existing CCSDS standart format are already widely in use.

Current open-source solutions:
A wide variety of software are available to complete a mission analysis. We can mention C++ based GMAT, Scilab based Celestlab, Java based Orekit, Python based Poliastro, itself partially based on Astropy [4], Julia based SatelliteToolbox. Each space agency usually provides and uses their own software (e.g. CNES with Patrius, PSIMU, Genius). Similarly, different academic institutions provide and use their own software (e.g DOCKS, or JSatorb [5]).

Non open-source options:
We can also mention the VTS tool from CNES, supported by Spacebel, with many features such as Celestia window visualization.

Activity Profile:
A spacecraft may have different operating modes. Usually, at least a safe mode and an nominal mode are defined. Spacecraft modes impact other subsystems by affecting their power consumption and thermal dissipation for example. During safe mode, all non-essential systems shutdown. An activity profile defines a typical mission profile, i.e. the successive modes through which the satellite will pass. Different activity profile might be considered including expected scenario, alternative orbit scenario and downgraded scenario. It is recommended to define at least one activity profile for space missions. Activity profiles can be handled with a basic CSV (Comma-Separated Values) file. For more advanced features, we can mention academics contributions such as the jupyter framework proposed by UC3M (Universidad Carlos III de Madrid).

2.1.2 Link Budget

Estimation of the margin for uplink and downlink budget between the spacecraft and ground stations (or another spacecraft) should be computed. These margins usually allow to size the useful data flow that can be exploited during the visibility windows of the ground stations.

Inputs

- ▲ Orbital parameters
- ▲ Mission requirement (e.g: Bit Error Rate)
- ▲ Material description**

Outputs

- ▼ Raw and useful data flow (uplink, downlink)
- ▼ Link budget margins (uplink, downlink)

** Theoretically, it could be Electronic Data Sheets (EDS). Usually the standard material description is entered into the system manually.

Current open-source solutions:
Basic spreadsheets are usually sufficient for pre-sizing the link budget. For more advanced features, a wide range of online resources are available. We can mention the Amsat-UK spread sheet and some Python libraries: linkpredict or Dosa.

2.1.3 Data Budget

The storage capacity of on-board data should be sized according to data produced, the extent of the different telecommunications data streams, and the time between visibility windows to ground stations. During the time the spacecraft is not in visibility with ground stations, on-board storage capacity should be sufficient to accommodate gathered data until the next visibility window, including for instance housekeeping data.

Inputs

- ▲ Visibility windows
- ▲ Activity profile
- ▲ List of ground stations
- ▲ Satellites inter-links

Outputs

- ▼ Available data flow (up-link/down-link)
- ▼ Required on-board storage

Basic spreadsheets are usually sufficient for pre-sizing the data budget.

2.1.4 Mechanical architecture

A 3D structural model (often a simplified version) is necessarv. In this model it is defined the mechanical structure of each sub-component, and the composition of the different sub-components (or mass of the different sub-components depending of the software).

Inputs	Outputs
▲ Components listing and specification	▼ 3D structural model

Current open-source solutions:
Some generic software are designed to build the mechanical structure: OpenSCAD or Blender. See following models for OpenSCAD and Blender.

The Libre Cubes Space foundation repositories and the OreSat [6] project also provide their source code and CAD available (here for the OreSat project).

Alternative non open-source options:

IDM-CIC is not open source software. However academics usually have access to a free license. IDM-CIC is designed to build a preliminary mechanical architecture, and compute mass and power budgets. It allows to take into consideration the different spacecraft modes when estimating the power consumption. IDM-CIC incorporates Sketchup which is not open-source either.

2.1.5 Energy and Power System

Concerning the Energy and Power System (EPS), preliminary design includes checking the ability of the platform to provide enough power for the mission. In CubeSats, the source of the power usually comes from energy collected with solar panels. Batteries allow to store energy and to provide it during eclipses, phases of peak demand (e.g. telecommunication bursts), or when solar panels have not yet been deployed.

Inputs	Outputs
▲ Activity profile	▼ Required batteries
▲ Eclipses	▼ Required solar cells
▲ Attitude profile*	▼ Power consumption profile
▲ Material description**	

Basic spreadsheets are usually sufficient for the presizing of the EPS. For a more advanced features, the reader can refer to Libre Cube Space foundation repositories or academic projects such as nanopower.

2.1.6 ADCS

Concerning the Attitude Determination and Control System (ADCS), during the preliminary design, a basic sizing study is usually enough. According to sensors accuracy and actuators capacities, pointing and stabilization requirements must be checked. In addition, the preliminary ADCS simulations may usually provide an AEM file (Attitude Ephemeris Message) related to the platform.

Inputs	Outputs
▲ Material Description**	▼ Check actuators and sensors mission compliance
▲ Activity Profile	▼ Attitude Profile*
▲ Ephemeris*	

Basic spreadsheets are usually sufficient for the presizing of the ADCS. For a more advanced features, the reader can refer to the UPSAT repositories or academics contributions.

2.1.7 End-of-life disposal

Legal or regulatory aspects should not be neglected during preliminary design. ESA highlights that “it is recommended that satellites and orbital stages be commanded to reenter Earth’s atmosphere within 25 years of mission completion, if their deployment orbit altitude is below 2000 km (in the LEO region).” This 25 year period is also supported by the Inter-Agency Space Debris Coordination Committee as stated in their IADC Space Debris Mitigation Guidelines [7]. In order to respect this recommendation, orbit should be propagated with sufficient accuracy over a long period of time.

Inputs	Outputs
▲ Orbital parameters	▼ Check compliance with Space Law
▲ 3D structural model	
▲ Launch date	

Current open-source solutions:

Most mission analysis tools described previously (see 2.1.1) can be adjusted to propagate orbits with sufficient accuracy on a long term period. For example, GMAT already include some propagators such as RungeKutta89 or PrinceDormand78 that can be adjusted to propagate orbits on a long time period by setting large temporal step size.

Alternative non open-source options:

The Java freeware Stela (Semi-analytic Tool for End of Life Analysis) [8] allows to propagate LEO (and also GTO or GEO) orbits on a long term period. This allows to estimate time of re-entry.

2.1.8 Dissipation budget

Building a simplified thermal architecture allows to estimate temperature reached on each relevant node of the spacecraft. A thermal architecture allows to check if each equipment is operating within its correct temperature range.

Inputs	Outputs
▲ Solar Flux Received	▼ Nodes temperature
▲ Activity profile	▼ Check compliance with hardware specification
▲ Attitude profile files	
▲ 3D structural model	

Current open-source solutions:
open-source software dedicated to thermal analysis are uncommon. Proprietary software remain heavily used to simulate the thermal behavior, even in academic CubeSats. Nevertheless, it is possible to proceed to a basic analytical study [9]. In addition, Simusat is an academic open-source software which allows basic thermal node simulations. It is also interesting to point out that thermal models can be found, for example SwissCube [10], along the feedback of thermal flight data.

2.1.9 Radiation Budget

Considering radiation during the design stage may increase mission success. Some initiatives are already proposing open-source radiation tested platforms such as the PyCubed project [11]. A short description of the radiation environment is provided in [12].

Inputs	Outputs
▲ Solar Flux Received	▼ Received radiation flux per equipment
▲ Attitude profile files	▼ Check compliance with hardware specification
▲ 3D structural model	

Current open-source solutions:
Open-source software dedicated to radiation analysis are uncommon. Proprietary software remains heavily used, even on academic CubeSats. Nevertheless, an order of magnitude of the radiation expected to be received on the equipment can be roughly estimated

with conventional models [13].

Alternative non open-source options:

Space environment and radiation effects can be modeled using OMERE. Online tools such as ESA's SPENVIS [14] or NASA's Oltaris [15] are also available.

2.2 Methodology

Concurrent Design Engineering (CDE) applied to space mission preliminary design [16] is a methodology applied to facilitate the process of subsystem design parameters converging into preliminary models, and architectures. A review and comparison table between these tools has been proposed by Knoll and Golkar [17], also accessible online. Concurrent engineering does not replace standard project management tools. Such functionalities need to be covered by other tools. In the following, we assume that we have deployed:

- Nanospace [18] framework which supports a CDE methodology;
- standard project versioning (e.g a Git deposit);
- common team chat service (e.g RocketChat);
- video conference service (e.g Jitsi);
- project scheduler (e.g ProjectLibre);
- dedicated framework for task management, bugs and issues tracking, often coupled with previous tools (e.g GitLab).

3 Use-case scenario and tools

One of the prospects for the CREME mission is to offer space industry companies a low-cost, low-profile and small mass radiation monitor that is very versatile; so that it can be easily integrated on commercial satellites.

For CREME's preliminary design a python script was developed and used. Python's scripts used for this preliminary design are available here[1] under AGPL v3 licence. For pedagogical purposes, simple mission analysis scripts are provided, as well as an example of an input file is provided in .yml format (orbital parameters, power consumption of the platform, etc.) and an example of outputs, including:

- intermediary results such as remaining power graph and data budget graph
- a full report (in Markdown format) as a light but realistic preliminary design synthesis.

Orbitography calculus - Orbit propagation, eclipses determination and contact with ground station events - are managed with a automatically generated GMAT script. Link budget analysis is done with Dosa [2] (also in Python). Python scripting is used for the miscellaneous computations. The script is self sufficient for a first step mission analysis preliminary design. Nanospace [18] benefits from a python API that can be directly used in the script to share data between experts.

In this preliminary design example, neither ADCS, nor radiation or thermal considerations are included. These elements are of course fundamental during a real CubeSat mission preliminary design and each of these elements can be easily added since the approach is modular.

Traditionally, to ensure mission success, the worst case scenario is considered. This allows to consider margins, even if refinement of models may be required when the problem ends up being too constrained.

[1] `https://gitlab.isae-supaero.fr/creme-project/creme-scripts`

[2] `https://sourceforge.isae.fr/projects/dosa_link_budget_analysis`

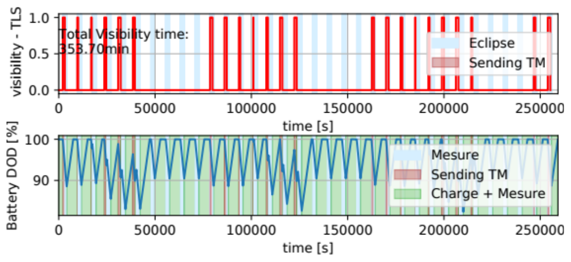

Figure 1. Simulation example: battery remaining power (3 day at 2000km with a SSO midday/midnight orbit - 4 solar arrays of 6 W).

For the example, the following are the requirements derived from the Payload Principal Investigator needs and desires:

- Orbit: LEO Orbit
- Payload altitude: highest possible
- Telecommunication: Bit error Rate $< 10^{-5}$
- Payload required power: 1W

4 Conclusion

With this paper, we emphasize that different open-source tools, as well as methodologies, are available and can be used for achieving a complete and relevant CubeSat preliminary design. A pedagogical use-case, based on CREME CubeSat project, as well as corresponding open-source scripts are available on a gitlab repository, for a light illustration of a preliminary design.

Future works include graphical interface version of these scripts, full integration of relevant and different open-source tools with Nanospace framework (i.e a minimal thermal architecture of a CubeSat is required, but few open-source tools are really providing help to design it).

References

[1] A. Scholz and J.-N. Juang, “Toward open source cubesat design,” Acta astronautica, vol. 115, pp. 384–392, 2015.
[2] M. G. Mariano, F. E. Morsch, M. S. Vega, S. L. Oriel, S. L. Kessler, B. E. Augusto et al., “Qualification and validation test methodology of the open-source cubesat floripasat-i,” Journal of Systems Engineering and Electronics, vol. 31, no. 6, pp. 1230–1244, 2020.
[3] D. Geeroms, S. Bertho, M. De Roeve, R. Lempens, M. Ordies, and J. Prooth, “Ardusat, an arduino-based cubesat providing students with the opportunity to create their own satellite experiment and collect real-world space data,” in 22nd ESA Symposium on European Rocket and Balloon Programmes and Related Research, vol. 730. Citeseer, 2015, p. 643.
[4] A. M. Price-Whelan, B. Sip˝ocz, H. G¨unther, P. Lim, S. Crawford, S. Conseil, D. Shupe, M. Craig, N. Dencheva, A. Ginsburg et al., “The astropy project: Building an open-science project and status of the v2. 0 core package,” The Astronomical Journal, vol. 156, no. 3, p. 123, 2018.
[5] J. Hernanz-Gonzalez, T. Gateau, L. Senaneuch, T. Koudlansky, and P. Labedan, “Jsatorb: Isae-supaero’s open-source software tool for teaching classical orbital calculations,” ICATT, 2018.
[6] C. Spivey and E. Gizzi, “A modular, open source cubesat structure,” in AIAA Scitech 2021 Forum, 2021, p. 1256.
[7] Inter-Agency Space Debris Coordination Committee. (2007) IADC Space Debris Mitigation Guidelines. [Online]. Available: https://www.unoosa.org/documents/pdf/spacelaw/sd/IADC-2002-01-IADC-Space Debris-Guidelines-Revision1.pdf
[8] C. Le F`evre, H. Fraysse, V. Morand, A. Lamy, C. Cazaux, P. Mercier, C. Dental, F. Deleflie, and D. Handschuh, “Compliance of disposal orbits with the french space operations act: the good practices and the stela tool,” Acta Astronautica, vol. 94, no. 1, pp. 234–245, 2014.
[9] J. Claricoats and S. M. Dakka, “Design of power, propulsion, and thermal sub-systems for a 3u cubesat measuring earth’s radiation imbalance,” Aerospace, vol. 5, no. 2, p. 63, 2018.
[10] S. Rossi and A. Ivanov, “Thermal model for cubesat: a simple and easy model from the swisscube’s thermal flight data,” in Proceedings of the International Astronautical Congress, 2013, pp. 9919–9928.
[11] M. Holliday, A. Ramirez, C. Settle, T. Tatum, D. Senesky, and Z. Manchester, “Pycubed: An open-source, radiation-tested cubesat platform programmable entirely in python,” Proceedings of the Small Satellite Conference, 2019.
[12] A. Parsons, “Radiation effects on cubesats,” The UNSW Canberra at ADFA Journal of Undergraduate Engineering Research, vol. 10, no. 2, 2018.
[13] C. Rodriguez-Solano, U. Hugentobler, and P. ˇStˇep´anek, “Comparison of earth radiation pressure models for doris satellites,” IDS Workshop, 2012.
[14] D. Heynderickx, B. Quaghebeur, J. Wera, E. Daly, and H. Evans, “Esa’s space environment information system (spenvis): A web-based tool for assessing radiation doses and effects in spacecraft systems,” Orfeo, the institutional Open Access repository for Federal Science Policy funded research, 2005.
[15] R. C. Singleterry Jr, S. R. Blattnig, M. S. Clowdsley, G. D. Qualls, C. A. Sandridge, L. C. Simonsen, T. C. Slaba, S. A. Walker, F. F. Badavi, J. L. Spangler et al., “Oltaris: On-line tool for the assessment of radiation in space,” Acta Astronautica, vol. 68, no. 7-8, pp. 1086–1097, 2011.
[16] M. Bandecchi, B. Melton, and F. Ongaro, “Concurrent engineering applied to space mission assessment and design,” ESA bulletin, vol. 99, no. Journal Article, 1999.
[17] D. Knoll and A. Golkar, “A coordination method for concurrent design and a collaboration tool for parametric system models,” Concurrent Engineering, vol. 26, no. 1, pp. 5–21, 2018.
[18] T. Gateau, L. Senaneuch, S. Salas Cordero, and R. Vingerhoeds, “Open-source framework for the concurrent design of cubesats,” in 2021 IEEE International Symposium on Systems Engineering (ISSE). IEEE, 2021, pp. 1–8.