sid.inpe.br/mtc-m21c/2021/02.21.20.48-TDI

CONOPS2M: CONCEPT OF OPERATIONS MODELLING FOR CUBESAT-BASED SPACE MISSIONS

Danilo Pallamin de Almeida

Master's Dissertation of the Graduate Course in Engineering and Space Technology/Space Systems Engineering and Management, guided by Drs. Maria de Fátima Mattiello Francisco, and Fabiano Luis de Sousa, approved in January 29, 2021.

URL of the original document:
<http://urlib.net/8JMKD3MGP3W34R/447TRJB>

INPE
São José dos Campos
2021

PUBLISHED BY:

Instituto Nacional de Pesquisas Espaciais - INPE
Coordenação de Ensino, Pesquisa e Extensão (COEPE)
Divisão de Biblioteca (DIBIB)
CEP 12.227-010
São José dos Campos - SP - Brasil
Tel.:(012) 3208-6923/7348
E-mail: pubtc@inpe.br

sid.inpe.br/mtc-m21c/2021/02.21.20.48-TDI

CONOPS2M: CONCEPT OF OPERATIONS MODELLING FOR CUBESAT-BASED SPACE MISSIONS

Danilo Pallamin de Almeida

Master's Dissertation of the Graduate Course in Engineering and Space Technology/Space Systems Engineering and Management, guided by Drs. Maria de Fátima Mattiello Francisco, and Fabiano Luis de Sousa, approved in January 29, 2021.

URL of the original document:
<http://urlib.net/8JMKD3MGP3W34R/447TRJB>

INPE
São José dos Campos
2021

Cataloging in Publication Data

Almeida, Danilo Pallamin de.

Al64c Conops2M: Concept of operations modelling for CubeSat-based space missions / Danilo Pallamin de Almeida. – São José dos Campos : INPE, 2021.

xxii + 77 p. ; (sid.inpe.br/mtc-m21c/2021/02.21.20.48-TDI)

Dissertation (Master in Engineering and Space Technology/Space Systems Engineering and Management) – Instituto Nacional de Pesquisas Espaciais, São José dos Campos, 2021.

Guiding : Drs. Maria de Fátima Mattiello Francisco, and Fabiano Luis de Sousa.

1. Concept of operations. 2. Modelling & Simulation. 3. Model based systems engineering. 4. CubeSat. I.Title.

CDU 629.783

MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÕES

INSTITUTO NACIONAL DE PESQUISAS ESPACIAIS

Serviço de Pós-Graduação-SEPGR

Pós-Graduação em ETE/Engenharia e Gerenciamento de Sistemas Espaciais.

DEFESA FINAL DE DISSERTAÇÃO DE DANILO PALLAMIN DE ALMEIDA

No dia 29 de janeiro de 2021, às 10h, por videoconferência, o aluno mencionado acima defendeu seu trabalho final (apresentação oral seguida de arguição) perante uma Banca Examinadora, cujos membros estão listados abaixo. O aluno foi APROVADO pela Banca Examinadora, por UNANIMIDADE, em cumprimento ao requisito exigido para obtenção do Título de Mestre em Engenharia e Tecnologia Espaciais/Eng. Gerenc. de Sistemas Espaciais. O trabalho precisa da incorporação das correções sugeridas pela Banca Examinadora e revisão final pelos ORIENTADORES.

Título novo: "Conops2M: Concept of Operations Modelling for CubeSat-Based Space Missions"

Eu, Walter Abrahão dos Santos, como Presidente da Banca Examinadora, assino esta ATA em nome de todos os membros.

Membros da Banca

Dr. Walter Abrahão dos Santos Presidente INPE.
Dra. Maria de Fátima Mattiello Francisco Orientador(a) INPE.
Dr. Fabiano Luis de Sousa Orientador(a) INPE
Dra. Ana Maria Ambrosio Membro da Banca INPE.
Dr. András Vörös Convidado(a) BME.

Documento assinado eletronicamente por **Walter Abrahão dos Santos**, **Tecnologista**, em 03/02/2021, às 17:24 (horário oficial de Brasília), com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.mctic.gov.br/verifica.html, informando o código verificador **6451029** e o código CRC **FDA1C286**.

Referência: Processo nº 01340.000741/2021-35 SEI nº 6451029

"We are a way for the cosmos to know itself".

Carl Sagan
em *"Cosmos"*, 1980

A meus pais ***Maria Inês*** *e* ***José Sérgio****, e à minha irmã* ***Juliana***

ACKNOWLEDGEMENTS

Agradeço à meus pais Inês e Sérgio e à minha irmã Juliana por toda a incomparável estrutura que me deram ao longo desses anos todos, e o imenso incentivo aos estudos. Sou muito privilegiado por ter vocês. Vocês são a base de tudo.

À minha orientadora Dra. Fátima, por ter me aceitado como aluno e me mostrar o caminho desde 2016. Obrigado por toda a orientação e conhecimento passado ao longo destes anos, pelo tempo investido e confiança depositada em mim. Obrigado pelas oportunidades únicas que a Sra. possibilitou e ofereceu.

Ao meu orientador Dr. Fabiano, por trazer o projeto para o ambiente do fantástico CPRIME e abrir um novo horizonte trazendo toda sua experiência e conhecimento. Obrigado pelo tempo, paciência, disponibilidade e por toda a orientação.

Ao Dr. Otávio Durão, por ter me confiado a incrível oportunidade em 2016 de fazer parte da missão NanosatC-Br2 e aberto as portas para o INPE e o setor espacial.

Ao Dr. Ronan pela abertura da excelente ferramenta ForPlan e toda a ajuda e apoio dado, sempre com incrível disposição.

Aos meus professores, que me permitiram enxergar mais longe.

Ao Carlos pela parceria diária (e muitas vezes noturna) na sala, no lab e nas atividades do Br2 e do SPORT.

A todos da equipe do NanosatC-Br2, pelo incrível projeto que realizamos.

À Dra. Jenny e o pessoal do LIT pelo apoio na campanha de AIT do Br2.

Aos colegas do CEI, em especial à Lídia, por todo o apoio e compartilhamento.

À Denise, por todo o trabalho neste processo desde 2018 que me ajudou a voltar a alcançar pelo cosmos.

Ao INPE e todos os funcionários, por terem provido toda a infraestrutura necessária para a realização deste trabalho.

A special thanks to Bence and Vince from the BME FTSRG for the huge contribution to this work through ADVANCE.

This work has been partially supported by the project ADVANCE - Ad-

dressing Verification & Validation Challenges in Future Cyber-Physical Systems (https://www.advance-rise.eu/) - call H2020-MSCA-RISE-2018, number 823788 - with contribuitions of Graics Bence and Molnár Vince from BME, during 1 month period of secondment at INPE.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) - Finance Code 001.

ABSTRACT

The increasing accessibility to space provided by small satellites, especially the CubeSat standard, with lower costs and shorter development time, has stimulated many new missions and possibilities. As a measure to reduce CubeSat mission failure rates, which are comparatively high, there is a need to tailor Systems Engineering practices and methodologies to fit the time and cost budgets of these kinds of missions. Throughout the entire life-cycle of space missions, modelling and simulation play a large role in supporting the engineering and operation activities. Early stage design activities, such as feasibility and performance analyses, trade-off studies, and requirement specifications, are commonly performed based on concurrent engineering practices in design offices, such as Concurrent Engineering Centers, and benefit from modelling and simulation. In this dissertation, the author proposes and demonstrates a modelling process, called Conops2M, that guides the construction of an initial mission architecture focused on the concept of operations, preparing for the simulation of operation scenarios to be used in early phase design trade-studies, through automatic model transformation and code generation. Conops2M transforms mission operation objectives and requirements into functions realized by the mission's Space and Ground Segments, highlighting the interactions and dependencies among them. Conops2M is demonstrated through an instantiation for a generic CubeSat mission, and then applied for the NanosatC-Br2, a scientific CubeSat mission developed by Brazil's National Institute for Space Research (INPE) and the Federal University of Santa Maria (UFSM). An example trade study analysis is conducted comparing the simulation of different operation scenarios generated using Conops2M, and the results are discussed.

Keywords: Concept of Operations. Modelling & Simulation. Model Based Systems Engineering. CubeSat.

CONOPS2M: MODELAGEM DO CONCEITO DE OPERAÇÕES PARA MISSÕES ESPACIAIS BASEADAS EM CUBESATS

RESUMO

A crescente acessibilidade ao espaço providenciada por pequenos satélites, em especial do padrão CubeSat, com menores custos e períodos de desenvolvimento mais curtos, tem estimulado várias novas missões e possibilidades. Como uma medida para reduzir as taxas de falhas em missões CubeSat, que são comparativamente altas, há uma necessidade de adaptar as práticas e metodologias de Engenharia de Sistemas para as adequar às disponibilidades de recursos financeiros e cronogramas deste tipo de missão. Ao longo de todo o ciclo de vida de missões espaciais, modelagem e simulação têm um grande papel em apoiar as atividades de engenharia e operações. Atividades de projeto iniciais, como análises de viabilidade e performance, estudos de *trade-off*, e especificação de requisitos, são comumente feitos baseados em práticas de engenharia simultânea em escritórios de projetos, como Centros de Engenharia Simultânea, e se beneficiam de modelagem e simulação. Nesta dissertação, o autor propõe e demonstra um processo de modelagem, denominado Conops2M, que guia a construção de uma arquitetura inicial de missão focada no conceito de operações, preparando para a simulação de cenários operacionais ser utilizada em estudos de *trade-off* em estágios iniciais de projeto, através de transformação automática de modelo e geração automática de código. Conops2M transforma objetivos e requisitos operacionais de missão em funções realizadas pelos segmentos Espacial e Solo da missão, destacando as interações e as dependências entre eles. Conops2M é demonstrado através de uma instanciação para uma missão CubeSat genérica, e em seguida é aplicado para o NanosatC-Br2, uma missão CubeSat científica desenvolvida pelo Instituto Nacional de Pesquisas Espaciais (INPE) e pela Universidade Federal de Santa Maria (UFSM). Um exemplo de análise de um estudo de *trade-off* é conduzido comparando a simulação de diferentes cenários ope-racionais gerados usando Conops2M, e os resultados são discutidos.

Palavras-chave: CONOPS. Modelo. MBSE. CubeSat.

LIST OF FIGURES

Page

LIST OF TABLES

Page

LIST OF ABBREVIATIONS

AIT	–	Assembly, Integration and Test
C2F	–	Capella To ForPlan
CDB	–	Class Diagram Blank
CE	–	Concurrent Engineering
CEC	–	Concurrent Engineering Center
CONOPS	–	Concept of Operations
COTS	–	Commercial Off-The-Shelf
CPRIME	–	Space Missions Integrated Design Center
CRM	–	CubeSat Reference Model
DSML	–	Domain Specific Modeling Language
ECSS	–	European Cooperation for Space Standardization
ESA	–	European Space Agency
INCOSE	–	International Council on Systems Engineering
INPE	–	National Institute For Space Research
LEO	–	Low Earth Orbit
MBSE	–	Model-Based Systems Engineering
NASA	–	National Aeronautics and Space Administration
NCBR2	–	NanosatC-Br2
OMG	–	Object Management Group
ROI	–	Region of Interest
SE	–	Systems Engineering
SysML	–	Systems Modeling Language
V&V	–	Verification and Validation

CONTENTS

Page

1 INTRODUCTION

The human space exploration endeavour has been immensely benefited by the technological advances of humankind, while being a key driver for research and development in countless fields of science and engineering. From the early space race to interplanetary exploration, humanity has developed numerous applications with many different motivations, resulting in a consolidated and still growing global space industry. Besides the direct impact of each application, such as remote sensing and connectivity satellites for example, human civilization benefits from many applications that were made possible in other areas and industries due to technologies developed for the space industry, which are called spin-offs.

To escape the pull of Earth's gravity and launch anything into space, an immense amount of energy is required to get each gram of mass up to the altitude and velocity required, not to mention beating the drag effects of the resistances of Earth's atmosphere. Up to today, the way humans have achieved this is through the use of rockets, which have limited payload volume and mass they can take. Each unit of mass and volume has a high cost and value for each launch.

Since getting into space is hard and expensive, spacecrafts are traditionally developed with a very high reliability, to guarantee as much as possible they will operate as expected once they reach their destination. This is assured through extensive amount of testing, verification, and validation, from components up to system level. Also, the space environment is extremely hazardous. The radiation and temperature levels outside Earth's atmosphere can damage the sensitive electronics, so they need to be shielded. In addition, the rocket launch itself is very violent and harsh. The combustion of rocket fuel generates vibration in the rocket, which is propagated mechanically through the structure of the rocket up to the nose cone, where the payloads are stored. The nose cone also interacts at high speeds with the gases of Earth's atmosphere, generating vibrations. These vibrations and acceleration are unfortunately transmitted to the payloads, resulting in high forces and stress the payloads have to endure. This requires spacecrafts to have structures much more resistant than what would be required for operation, just to survive the launch. All these factors contribute to the high complexity and cost of space systems.

Modern electronics and materials have revolutionized the space industry, allowing the development of smaller and lighter equipment. For the past 20 years, a meaningful amount of nanosatellite-class spacecrafts have been developed and launched, and this is largely due to the *CubeSat* standard, which has been highly adopted world-

wide both commercially and academically. The launch of over 1000 CubeSats over the last 10 years has sparked the development of a global industry with over 500 companies (KULU, 2019) that provide solutions based on the standard. CubeSats have brought a paradigm shift in the space industry, providing relatively simple and cost-effective access to space.

As a cost-reduction strategy, CubeSats typically are made with Commercial Off-The-Shelf (COTS) components, which commonly do not have space-grade ratings and consequently have a lower life-expectancy and robustness. Design teams (especially in university-class missions) commonly have little experience, and it is also common to see teams skipping some Systems Engineering, AIT and V&V procedures due to schedule constraints and cost budget limitations. Along with many other possible contributing reasons, CubeSat missions have high failure rates (SWARTWOUT; JAYNE, 2016) (VENTURINI et al., 2017).

To shorten development cycles while reducing failure rates, it is evident there is a need to tailor SE practices and methodologies in order to better fit and benefit from characteristics from the CubeSat standard such as platforming and reusability. Traditional approaches for large satellite development may not be suitable for CubeSats, therefore teams around the world are developing modern SE approaches such as Asundi and Fitz-Coy (2013), Waseem and Sadiq (2018) and Fischer et al. (2017).

Modelling and Simulation (M&S) have become essential throughout the entire life cycle of space missions. It is applied heavily from conception to operation, and even disposal (EICKHOFF; HENDRICKS, 2005) (EICKHOFF, 2009). Models help provide an understanding of the system, exchange of information and drive discussions among design team members. M&S can be used to allow customers to monitor and shape the evolution of the project, and to support the design, development and testing of the system. Customers can deliver part of the system specification as models, and the primary contractor can deliver models back as part of the review process (ECSS, b). This permits faster communication and reduces cost in reaching a better understanding of the system among stakeholders. Efforts are being made by many organizations to create an M&S infrastructure to permit reuse, technology evolution, and minimize costs (ECSS, b).

Regarding the application of models in the early stage design activities of space mission, teams from Concurrent Engineering Centers (CEC) around the world are bringing the formalized use of modeling to their processes of mission analysis at

Phase 0/A, often performed based on an optimized Concurrent Engineering (CE) approach (EICKHOFF, 2009). Model-Based Systems Engineering (MBSE) is a discipline that supports systems engineers in developing system models integrating design information from multiple domains. It evidences interfaces and overall system architectures, enabling the whole team to visualize concurrent activity and identify conflicts more efficiently (IWATA et al., 2015).

Regarding the application of simulation tools to support Phase 0/A, Kranz et al. (KRANZ et al., 2015) elaborate on the definition and contributions of a System Concept Simulator on the rapid evaluation of system design concepts, assisting in trade studies. System models such as those developed by Raif et al. (RAIF et al., 2010) can be used to simulate dynamic behaviour of the satellite.

Mission objectives, requirements and constrains express the expected behavior of satellite in operation, from the stakeholders point of view, driving the processes of mission analysis at Phase 0/A. The Concept of Operations (CONOPS) discipline plays an important role in the mission requirements analysis, because it traverses both ground and space segments, aiming to highlight dependencies among mission elements and the space environment to meet the mission operation requirements (WERTZ et al., 2011). Usually CONOPS models take into account major elements of the mission such as data and power budgets. Designing the mission CONOPS including multiple operation scenarios may reveal requirements and needed design functions. Simulating the operation of a satellite and the associated ground segment can be very useful for the design team and other stakeholders to improve the understanding of how the system solution achieves mission objectives. It can assist in trade-off studies, power and data budget analyses, and validating design choices.

Studies using modelling and simulation in early stage space mission design have shown positive impacts, such as the Virtual Satellite project (SCHAUS et al., 2010) by the German Aerospace Center (DLR) supporting their Concurrent Engineering Facility (CEF). The European Space Agency (ESA) has developed a highly integrated MBSE process related to systems modelling and simulation supporting the generation and maturation of system requirements, architectures and system budgets (ESTABLE et al., 2017). Also, the European Cooperation for Space Standardization (ECSS) has issued a technical memorandum defining the recommendations for model based data exchange for early phases of engineering design (ECSS, c). DLR has also developed executable models for early spacecraft design verification (FISCHER et al., 2013) and to check space mission feasibility in early design phases

(AKHUNDOV et al., 2016).

The International Council on Systems Engineering Space Systems Working Group began investigating the applicability of MBSE for designing CubeSats in 2011, developing a SysML model for a CubeSat mission (SPANGELO et al., 2012). Their approach has then been extended into developing a reference model for CubeSat-based missions, called the Cubesat Reference Model (CRM) (KASLOW et al., 2018). The CRM is to be a starting point for design teams, and will serve as the Object Management Group's specification for modelling CubeSat missions. The CRM developed an approach to modelling and simulating the operation of a CubeSat using SysML integrated with simulation tools (SPANGELO et al., 2013).

At the Brazilian National Institute for Space Research (INPE), the team at INPE's CEC Space Missions Integrated Design Center (CPRIME) intends to transition to the adoption of MBSE, inspired by CEC's around the world (IWATA et al., 2015). The product of this dissertation originates from the effort to expand CPRIME's M&S infrastructure, laying groundwork for the adoption of MBSE into their design processes. The product is a modelling process, called Conops2M, for CubeSat-based missions that guides the construction of a primary mission architecture focused on the mission concept of operations, in order to simulate operation scenarios, assisting in early-stage design. Along with the modelling process, a plugin for the modelling tool was developed to integrate the modelling with the operation scenario simulation, automatically transforming the model into an input for the Satellite Simulator tool Forplan developed and used at CPRIME (CHAGAS et al., 2018).

The contribution of Conops2M is to guide the construction of the CONOPS model, based on the functionalities the system shall perform to reach the stakeholder objectives. It follows a top-down flow in abstraction levels translating the stakeholder objectives into functions each equipment aboard the spacecraft will realize, to determine when the equipment shall operate, resulting in a simulatable architecture.

Conops2M was applied to an example study to support the creation of a CONOPS model for INPE's NanosatC-Br2 CubeSat mission, generating operation scenarios to be simulated using ForPlan for an example trade study.

1.1 Problem formulation

In space mission design, it is common for new requirements and dependencies to show up when the operations analysis is made only after the early stage design

phases, leading to large rework efforts, power and data budget issues, equipment redesign or replacement in late stages of the mission, or even partially compromise the mission if the problem cannot be solved in time. This is especially true with CubeSat missions, where the standardized form factor and interfaces of equipment facilitate the equipment selection process, giving a false "plug-and-play" notion. It is common for scientific missions to fit as many experiments/payloads inside the spacecraft as possible to make use of all the available interior space and mass budget, in an effort to minimize costs with other launches. This is done without considering the CONOPS at this point, which may lead to future problems in coordinating the operation of the payloads and issues regarding power and data budgets.

During design analysis activities conducted at CPRIME, the use of operation scenarios simulation for trade studies in the early stages of design contributes significantly to the reduction of the appearance of these unexpected dependencies, resulting in initial versions of mission designs less prone to later stages rework.

In order to simulate operation scenarios, it is needed to configure the simulation tool with the parameters that describe the mission, requiring the team to share a same mental model of the system. It is seen that developing a modelling infrastructure with a guided methodology to develop a system model centralizing information from multiple areas with a focus on the concept of operations, and using that model as an input for a simulator tool to simulate the operation scenarios can be very beneficial.

There is research being developed in many places around the world on Model-Based System Engineering practices to develop reference models for space missions and simulate certain aspects of them (with some focused on CubeSats). The author proposes an approach to a modelling process to develop early-stage design concept of operations for CubeSat-based missions, preparing for operation scenario simulation.

1.2 Dissertation objectives

The primary objective of this dissertation is:

- Propose a modelling process developed to guide the modelling of CubeSat-based missions and their CONOPS for early-stage design studies.

The secondary objectives of this dissertation are:

- Present the model transformation plugin for Forplan simulation.

- Exemplify the application of the modelling process through an instantiation for a real CubeSat mission, with an example simulation trade study.

1.3 Methodology

This section details the methodology in which the work presented in this dissertation inserts itself into.

The International Council on Systems Engineering (INCOSE) defines *methodology* as a collection of related processes, methods and tools (ESTEFAN et al., 2007). This study from INCOSE defines each part as:

- A Process (P): "is a logical sequence of tasks performed to achieve a particular objective. A process defines “WHAT” is to be done, without specifying “HOW” each task is performed. The structure of a process provides several levels of aggregation to allow analysis and definition to be done at various levels of detail to support different decision-making needs."

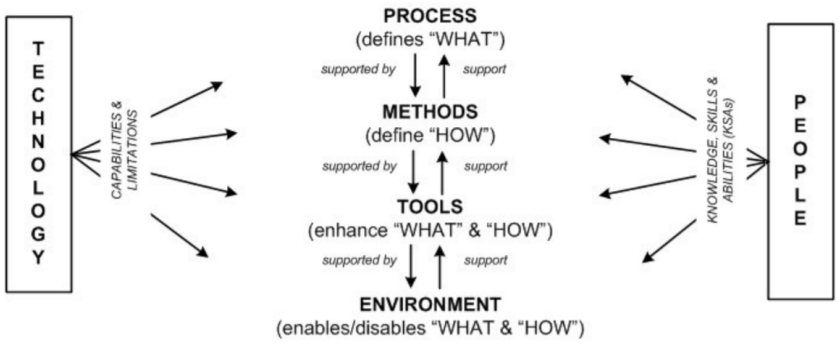
- A Method (M): "consists of techniques for performing a task, in other words, it defines the “HOW” of each task. (In this context, the words “method,” “technique,” “practice,” and “procedure” are often used interchangeably.)"

- A Tool (T): "is an instrument that, when applied to a particular method, can enhance the efficiency of the task; provided it is applied properly and by somebody with proper skills and training. The purpose of a tool should be to facilitate the accomplishment of the “HOWs.” In a broader sense, a tool enhances the “WHAT” and the “HOW"". Most tools used to support systems engineering are computer- or software-based, which also known as Computer Aided Engineering (CAE) tools.

Associated with an Environment (E), which is all the surroundings, external objects and conditions that influence on the group or individual. Figure 1.1 illustrates these elements and their relationships and effects on people and technology.

By reviewing INCOSE standards (INCOSE, 2019), recent published studies in journals and conferences such as (SPANGELO et al., 2012), (WASEEM; SADIQ, 2018), (LANGE et al., 2018), and the MBSE methodology survey by INCOSE (ESTEFAN et al., 2007), the most common approach for MBSE applications is to use OMG’s

Figure 1.1 - The PMTE Elements and Effects of Technology and People.



Source: Estefan et al. (2007).

SysML language supported by software tools such as *MagicDraw (NoMagic), Rational Rhapsody (IBM), or Enterprise Architect (Sparx Systems).*

However, these approaches have considerable barriers to entry, such as a lack of methodological guidance, a large initial effort with the structure, and subscription requirements for the tools.

Driven by the previous experiences and recommendations of previous student colleagues in the Systems Engineering academic department at INPE that employed MBSE in their projects ((CERQUEIRA, 2018) and (BURGER,)), the author has adopted an approach to use the Capella tool, developed by engineers at Thales Alenia, which is an open-source MBSE tool that incorporates the Arcadia method and a domain-specific modeling language very close to SysML (ROQUES, 2017). The main reasons for adopting Capella/Arcadia are:

- Capella is an open-source software, requiring no license and allowing for customized modifications, which resulted in C2F.
- Being an open-source software there are less barriers for peers to replicate, use and expand this work.
- Capella features a Domain-Specific Modelling Language (DSML) which is similar to SysML, yet the author found it to be more intuitive to learn.
- The methodological guidance of Arcadia fit well the purpose of a modelling process to guide users in the development of their own model, which is the object of this work.

By adopting Capella and Arcadia, the author considers the aforementioned barriers to entry are significantly reduced, and making the end product accessible is of essential value to the author.

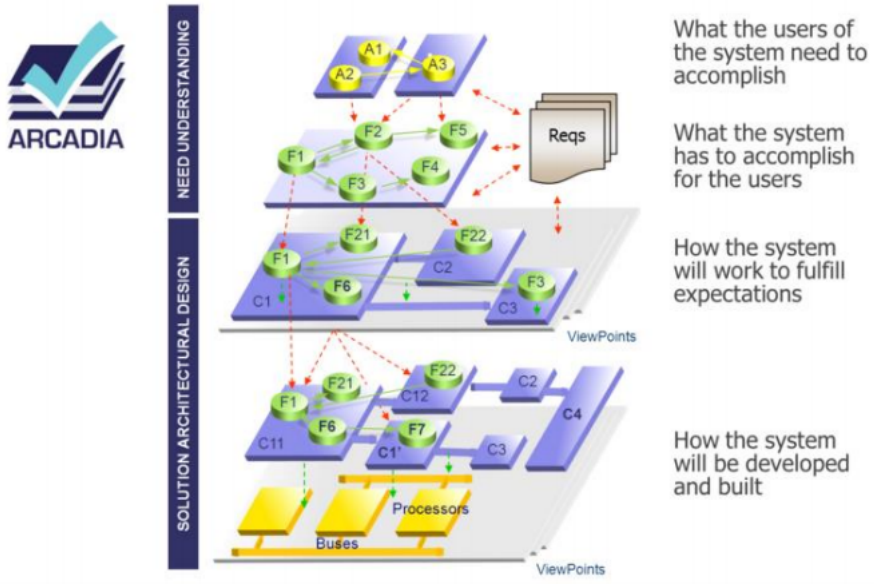
The four parts of the methodology of this work are explained in the following subsections:

1.3.1 Method

ARCADIA is a structured engineering method aimed at defining and validating the architecture of complex systems developed by Thales Alenia between 2005 and 2010 (ROQUES, 2017). It has a Domain Specific Modeling Language (DSML) that is inspired by UML/SysML and shares many concepts with these languages. ARCADIA is based on functional analysis and function allocation to components (ROQUES, 2016).

It follows an approach structured on successive engineering phases that first tackles need understanding, and then the solution architectural design, illustrated by Figure 1.2.

Figure 1.2 - ARCADIA Engineering Levels.



Source: Roques (2016).

The highest level of the method is the **Operational Analysis**, which captures *"what the users of the future system need to accomplish"*. At this level, the focus is on what

roles the users will have to perform to capture their needs.

The second level is the **System Analysis**, that involves the identification of the Capabilities and Functions of the system that will satisfy the operational needs, which captures *"what the system must accomplish for the users"*.

The third level is the **Logical Architecture**, which captures *"how the system will work to fulfill expectations"* and aims to identify Logical Components inside the system, their relations and their content, independently of any considerations of technology or implementation.

The fourth level is the **Physical Architecture**. The objective of this level is similar to the third level except defining the final architecture of the system (*"how the system will be built"*)

The final level is the **End Product Breakdown Structure (EPBS)**, which deduces the conditions each component must satisfy (*"what is expected from the provider of each component"*).

1.3.2 Process

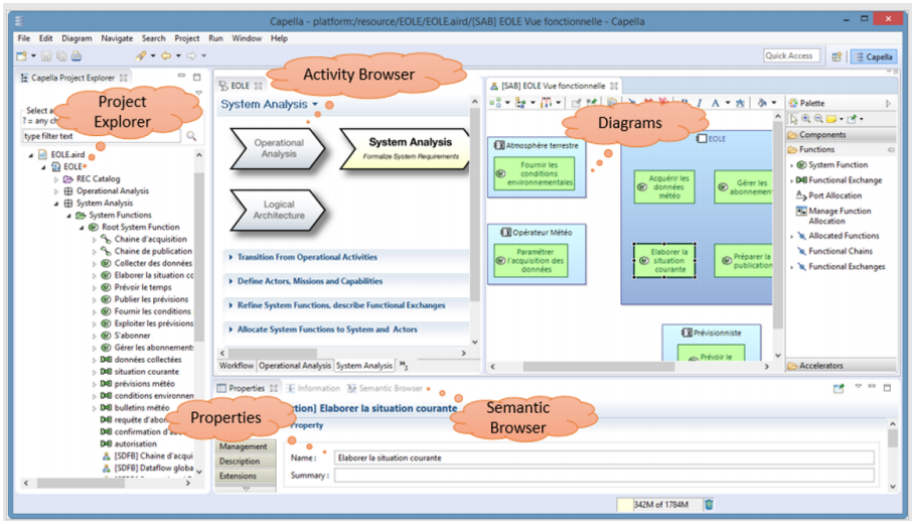
The Process of the methodology is the modelling process product of this dissertation, detailed in chapter 3. The process is composed of a series of steps to take in order to model the CubeSat mission and its concept of operations, and prepare the operation scenarios simulations.

1.3.3 Tool

Capella is an open-source Eclipse application that follows ARCADIA and implements their DSML. The workspace (Figure 1.3) provides a navigation throughout the diagrams through a tree diagram called the "Project Explorer", a "Properties" tab that describe properties of the model elements, a "Semantic Browser" which allows the user to browse through the model presenting the references surrounding the item, relations and diagrams it appears in, and an "Activity Browser" which exposes the user to the steps in the ARCADIA method used to create the diagrams in each phase.

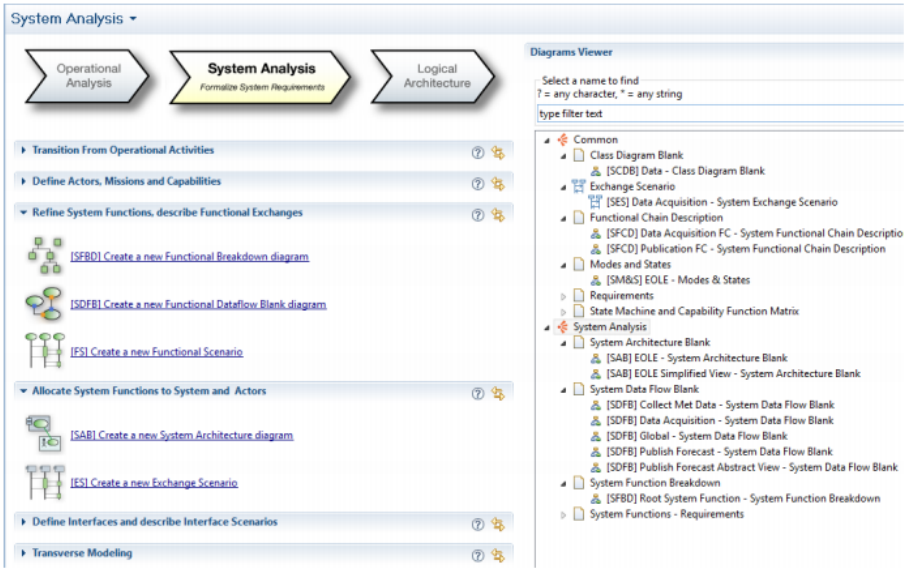
The Activity Browser guides the user throughout the method providing access to all the key activities of Capella and creation of all main diagrams, level by level, which can be seen in Figure 1.4.

Figure 1.3 - Overview of the Capella Interface



Source: Roques (2017).

Figure 1.4 - Capella Methodological Activity Browser.



Source: Roques (2016).

Because Capella is open-source, the development of a plugin enhancing the tool to convert resulting models into ForPlan simulator input configuration scripts was possible, and resulted in a significant part of the end product of this dissertation. The plugin is detailed in chapter 4.

1.3.4 Environment

The environment for the methodology is INPE's Concurrent Engineering Center CPRIME. This work was originated to expand CPRIME's Modelling and Simulation infrastructure and lay the groundwork for the adoption of MBSE into CPRIME's design processes.

1.4 Document structure

The remainder of this document is structured in the following manner:

- Chapter 2: Theoretical Background. In this chapter the main concepts that underlie and drive the work done in this dissertation are discussed. These concepts are: CubeSats; Modelling & Simulation; The ForPlan Simulator Tool; Model-Based Systems Engineering; The CubeSat Reference Model; Concurrent Engineering Centers; and Concept of Operations.
- Chapter 3: The Conops2M Modelling Process. In this chapter the modelling process is detailed. It is initially detailed through a general description, and then followed with an example application of the model for a generic CubeSat mission.
- Chapter 4: The CapellaToForPlan (C2F) Plugin. In this chapter, the plugin developed to transform the resulting model into the ForPlan simulator input is detailed.
- Chapter 5: Case Study. In this chapter the case study application of the modelling process is shown. Initially a description of the NanosatC-BR2 mission is provided, then the generated model is detailed, and finally some example simulation trade studies are shown.
- Chapter 6: Conclusion. This chapter discusses the results of the simulations in the different scenarios and how the modelling process affects the mission design; presents the future work for Conops2M and C2F envisioned by the author; and concludes the dissertation.

2 THEORETICAL BACKGROUND

2.1 CubeSats

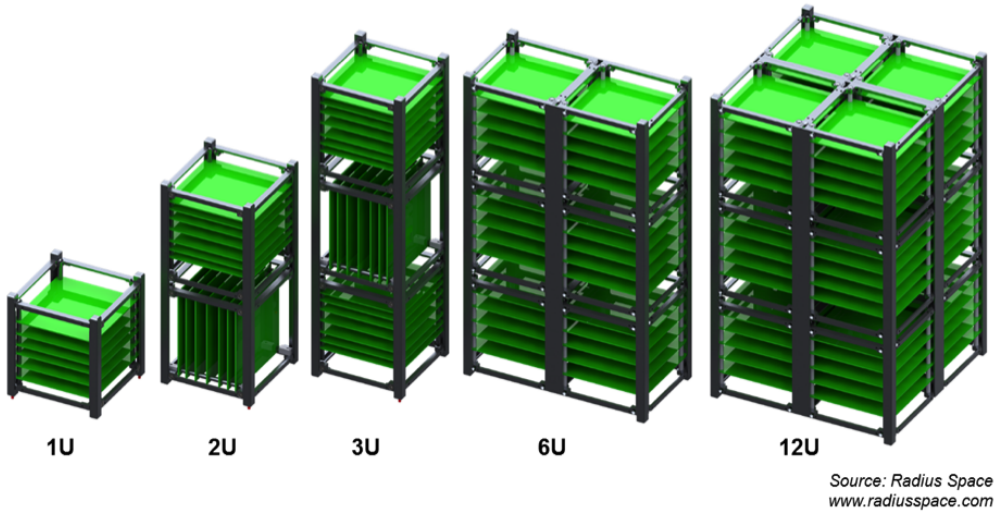
CubeSats are satellites typically in the nanosatellite class (from 1 to 10 kg) that follow the design specifications determined by the CubeSat standard. The standard was initially proposed in the late 1990s in a joint collaboration between Stanford University and the California Polytechnic Institute at San Luis Obispo, CA, as an effort to establish a reference design for the universities' small satellite programmes (HEIDT et al., 2000), helping students have experience in satellite missions, which are traditionally expensive. The first six CubeSats were launched together in 2003 (DAVID, 2004), and since then over 1200 have been launched (KULU, 2019), including the first two interplanetary CubeSats (AERONAUTICS; NASA, 2019). The standard proved itself effective for low-cost missions including: technological validation; dedicated "simple" science missions; and low-bandwidth communications, establishing itself among universities, institutes and governmental agencies. Subsequently, a niche market evolved with over 500 companies focused on CubeSats created worldwide (KULU, 2019). Future applications in high-speed communication networks and remote sensing are gaining interest of the industry.

The basic design of a CubeSat is a 10-centimeter cube, referred to as a 1-Unit (1U) structure. The cubes can be stacked together forming 2, 3 or 6-unit structures and so on, as seen in Figure 2.1.

The intent of the CubeSat Project was to "reduce cost and development time, increase accessibility to space, and sustain frequent launches" (MEHRPARVAR et al., 2014). The CubeSat standard reduces costs of space missions in many ways. The standardized form factors and equipment design parameters allow manufacturers to produce equipment in higher volumes, reducing individual cost. The standard allows design teams to opt for equipment from different manufacturers with reduced complications by having standardized interfaces among them.

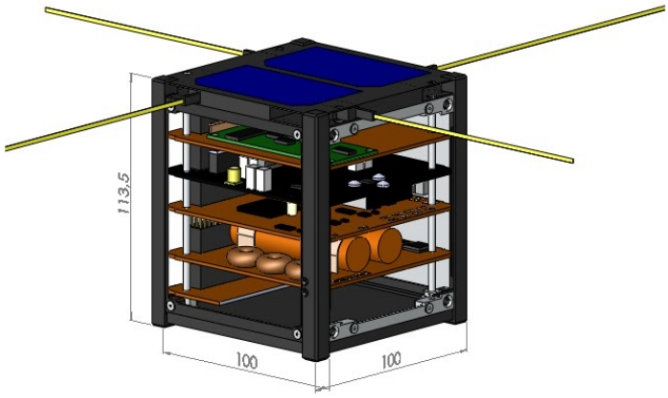
CubeSat equipment typically follow the PC/104 form factor, stacked vertically through the 104-pin CubeSat Kit Bus (CSKB) connectors. Through these connectors, most of the inter-module electrical interfaces are provided, such as power supply and digital communication (typically I2C). Figure 2.2 shows an example illustration depicting a 1U CubeSat, it's internal equipment stacked through the CSKB connectors, a top-mounted solar panel, and the UHF/VHF antennas.

Figure 2.1 - CubeSat Size Comparison.



Source: Radius Space (2018).

Figure 2.2 - 1U CubeSat Example Illustration.



Source: NanosatC-Br1, accessed (2014).

In comparison to traditional satellite missions, CubeSat missions have much lower costs, shorter schedules and smaller teams. Because of the smaller teams, schedule constraints and low financial budgets, it is common that some development teams skip some systems engineering, AIT and V&V procedures developed in the traditional space industry to guarantee a high mission reliability, and therefore CubeSat missions present high failure rates (SWARTWOUT; JAYNE, 2016) (VENTURINI et al., 2017). To assist developers with little experience, NASA has developed a technical report called *"CubeSat 101: Basic Concepts and Processes for First-Time CubeSat*

Developers", to guide the developers through the main processes involved concerning CubeSat Missions (NASA, 2017).

There are some downsides to the mass, volume and cost constraints. Due to these constraints, and also to reduce complexity, CubeSats generally have either static solar panels on the external surfaces and don't have Sun-pointing to maximize energy conversion, or have small deployable solar panel arrays. Either way, electrical power generation is limited generally to a couple of Watts per U. The form factor also limits the size of the battery packs available for each mission. The limited power generation and storage limits the equipment power consumption, which consequently impacts other design variables such as equipment processing power and other performance parameters. This limits the power available for radio communications, limiting the rate and volume of data that can be downloaded, which finally limits the amount of data the payloads can generate. Mass and volume also have a direct impact on equipment limitations. For example, in large Earth observation satellites, high resolution cameras may require large and heavy lenses and optic components that are virtually impossible to equip into a CubeSat.

This means CubeSats do not replace traditional large satellites.

2.2 Modelling and simulation in space systems

With the advances and evolution of information technology and development of more modern and precise tools, modelling and simulation (M&S) has become essential throughout the entire life-cycle of space missions, from conception through design and operation (EICKHOFF; HENDRICKS, 2005) (EICKHOFF, 2009).

M&S can be used to allow customers to monitor and shape the evolution of the project, and to support the design, development and testing of the system. Customers can deliver part of the "system specification" as models, and the prime contractor can deliver models back as part of the review process (ECSS, b). This permits faster communication and reduces cost in reaching a better understanding of the system among stakeholders. Efforts are being spent by many organizations to create an M&S infrastructure to permit reuse, technology evolution, and minimize costs (ECSS, b).

Throughout the entire development phase, M&S is heavily employed in space missions in many disciplines for equipment sizing, performance verification and design validation (BODIN et al., 2012) (LOWE; MACDONALD, 2014) (EICKHOFF, 2009) (EICK-

HOFF; HENDRICKS, 2005) (WANG et al., 2017). Each discipline relies on specific Computer Assisted Design (CAD) / Computer Assisted Engineering (CAE) softwares to support their activities.

In the early stages of the space system life-cycle (phase 0 and phase A), where the basic system characteristics are elaborated and defined, M&S is mainly used for requirement specification and trade-off analyses to develop the system concept and configuration alternatives, including budget and orbit analyses. At these stages, a System Concept Simulator (SCS) can be used to allow the rapid evaluation of system design concepts, assisting in trade studies (ECSS, b).

Phase 0/A analyses often are performed based on an optimised, Concurrent Engineering (CE) approach in design offices such as Concurrent Engineering Centers (CEC) (EICKHOFF, 2009). System simulation can significantly impact these activities, as shown by Kranz et al. (KRANZ et al., 2015) with a System Concept Simulator (SCS). Raif et al. show simulation of the dynamic behaviour of small satellites from a System Model and how that can benefit the design (RAIF et al., 2010).

2.2.1 Modelling

A model is a simplified representation of reality.

A consensual definition in literature given by (ROTHENBERG et al., 1989) is: *"Modeling, in the broadest sense, is the cost-effective use of something in place of something else for some cognitive purpose. It allows us to use something that is simpler, safer or cheaper than reality instead of reality for some purpose. A model represents reality for the given purpose; the model is an abstraction of reality in the sense that it cannot represent all aspects of reality. This allows us to deal with the world in a simplified manner, avoiding the complexity, danger and irreversibility of reality."*

Modeling is one of the most fundamental processes of the human mind. It underlies our ability to think and imagine, to use signs and language, to communicate, to use patterns, to predict, to describe and understand. For instance, numbers are models to represent quantities, and equations are models to describe natural phenomenons.

In engineering, each discipline has their own domain-specific models to represent different perspectives of the system. For instance, mechanical engineers use 3D CAD models to represent the physical design and layout of components, and thermal models to describe the propagation of heat and temperature distribution.

When correctly modeling a behaviour of the system, it is possible to predict the future outcome of the real operation within the accuracy of the model. By modeling the orbital dynamics of a satellite, for example, one can predict its location at any given point in time.

Designing complex systems often demands employing multiple models to represent different perspectives and phenomenons. Integrating the information held within these models is one of the key tasks system engineers perform.

2.2.2 Simulation

Simulation is the imitation of the operation of a real-world process or system over a period of time (BANKS, 2005). Eickhoff and Hendricks (2005) provide a definition for Simulation as: "Simulation is an approach for analyzing a dynamic system for gaining an insight to its dynamic behavior. Simulation implies conducting experiments on a model of the system."

When one provides a set of assumptions that define how a system or process works, and mathematically describes the phenomena involved (modelling), one can predict the state of the system or process at a given time by solving the equations. By solving the equations at multiple and incremental time steps, one can observe the variations caused in the system. By changing inputs and observing the resulting outputs, valuable insight may be obtained into which variables are most important and how variables interact.

Modern computers handle computations with precision and at extremely high speeds, and therefore computer simulations are getting faster and more faithful to reality with every advance in computation. Computer simulations are used in many ways, from graphical animations to gaming. Simulation software is applied in all domains of engineering and is a large part in modern engineering. Simulating engineering designs can save a lot of investment and effort by verifying and validating design choices without having to develop physical prototypes.

In space systems engineering, simulators are used along the entire life cycle of missions, assisting in the design phase, operation phase, and disposal. Simulation is an important form of validation in every step. For the early stage design phase, INPE has developed a satellite simulator to simulate the space environment with a satellite operation to verify and validate system design choices. The engineering team at the concurrent engineering center CPRIME perform trade-off studies by changing the

design choices and analyzing the simulation outputs. The simulator is described in the following Section 2.2.3.

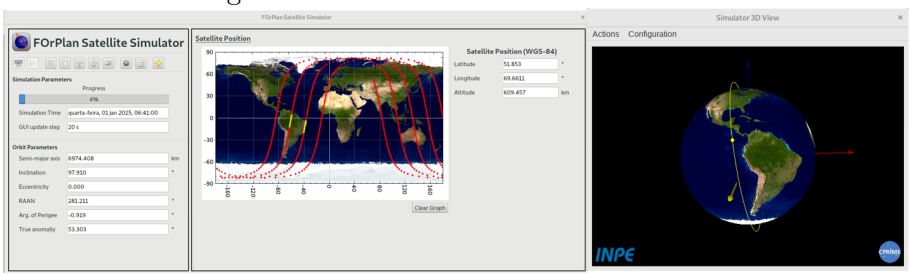
2.2.3 INPE/CPRIME's ForPlan simulator

INPE's Concurrent Engineering Center, the Space Missions Integrated Design Center (CPRIME), has developed a satellite simulator with the main objective of performing a functional simulation of satellites and associated ground segment to reflect operational scenarios of the mission under analysis, called *ForPlan Simulator* (CHAGAS et al., 2018).

Designed for verification of mission concept of operations during studies carried out mainly in Pre-Phase A at CPRIME, it is a simulation tool focused on the dynamics of data and power usage, and data exchanged between the satellite and the ground stations (CHAGAS et al., 2016).

The simulator is divided into a simulation core and the graphical user interface (GUI). The simulation core handles all of the simulation computations. The resulting data of the computations are then displayed to the user in a user-friendly way through the GUI, which can also handle inputs to the simulation, such as adjusting the time-step of each iteration. Figure 2.3 shows the main GUI window of the simulator.

Figure 2.3 - ForPlan Main GUI Window



Source: CPRIME (2020).

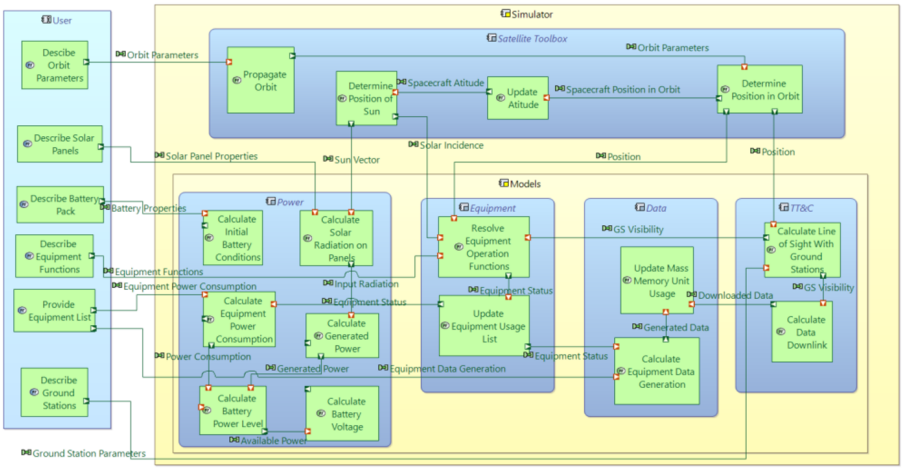
The simulation core is composed of the following features:

- A Space Environment module, responsible for orbit propagation, determining the Sun position, determining the satellites position and attitude, and checking if the satellite is above a specific country and if it's inside the visibility circle of the configured ground stations.

- An Equipment module, responsible for simulating the behaviour of the satellite equipment in terms of electrical power consumption and data volume generation.
- A Power Subsystem module, that calculates the generated power by the solar panel array, factors the power consumption of all the on-board equipment, and calculates the resulting battery charge.
- The On-Board Data Handling and TT&C Modules, which calculate the on-board mass memory usage and data transfer to the ground stations.

The simulation core is written in Julia language (JULIA..., 2020) and is based on a toolbox developed by Chagas (CHAGAS et al., 2018) called Satellite Toolbox, developed to aid in CPRIME studies with orbit analyses. Figure 2.4 shows a Capella diagram developed by the author while studying ForPlan's source code to understand how each module works and their interfaces with the user. It shows the functions each module and the user must perform, and the interfaces among them.

Figure 2.4 - ForPlan Functions/Interfaces Capella Diagram.



Source: Author.

To define a mission and prepare it for simulation on ForPlan, users (who will run a simulation using the software) must write a Julia language configuration script where all of the simulation parameters (associated to the User in Figure 2.4) are declared. The definition and declaration of simulation parameters follow predetermined rules and methods/functions according to the development of ForPlan's software, and for this reason, it is preeminent that the users must have a working knowledge of this

part of the source code, and be able to write the functions themselves.

Therefore, in order to make ForPlan a more accessible tool for distribution, it could be of great advantage to have a graphical, user-friendly tool to define the parameters and automatically generate the Julia script. This is the main driver for the last step of the modelling process shown in this dissertation, described in Section 3.1.1 and Chapter 4.

2.3 Model-Based Systems Engineering (MBSE)

In traditional Systems Engineering (SE), documents are used to record and store mission information, such as requirements, concept of operations, interfaces and other specifications. This results in creating, updating, reviewing and managing several different documents and their configurations, which for complex systems can become numerous.

Model-Based Systems Engineering is a SE approach that brings models as the primary source of information, transitioning from a document-centric to a model-centric methodology. The International Council on Systems Engineering (INCOSE) defines it as *"the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases."* (INCOSE, 2007). In other words, it is an approach that aims to synthesize the many documents that can be generated throughout the SE process into an integrated System Model (SM), which can result in a more modern, efficient and organized medium to record and store information of the referenced system.

Due to the trend in engineering disciplines in model-centric approaches, in the 2007 INCOSE International Workshop, the MBSE Initiative was initiated to *"Promote, advance, and institutionalize the practice of MBSE to attain the MBSE 2020 Vision through broad industry and academic involvement in Research; Standards; Processes, Practices and Methods; Tools & Technology; and Outreach, Training & Education"* (INCOSE,).

Implementing MBSE in small satellite missions is a relatively new topic, with the first found study from the INCOSE team in 2012 (SPANGELO et al., 2012), teams around the world are contributing to the topic using different methods and tools such as the ones shown by Guo et al. (2014), Fischer et al. (2017) and Waseem and Sadiq (2018). They show MBSE to be a promising approach for increased productivity and

quality, and lower development risk in these types of missions. Artifacts developed can be used as "single-source-of-truth" and therefore can lower the complexity of managing multiple documents and disperse information.

Following the platforming trend, a recent study regarding reuse with MBSE in space systems development (LANGE et al., 2018) shows a systematic approach for enhancing reuse with MBSE in space systems. It also raises a reuse infrastructure that MBSE provides with its possible benefits and some example applications.

For an ESA mission called e.Deorbit, Estable et al. (2017) developed an integrated and collaborative MBSE process to support the development and maturation of system requirements, architectures and system budgets. This work shows an example application of systems modelling integrated with analytical modelling and simulation tools using simulation execution to perform trade off studies.

2.4 The CubeSat reference model

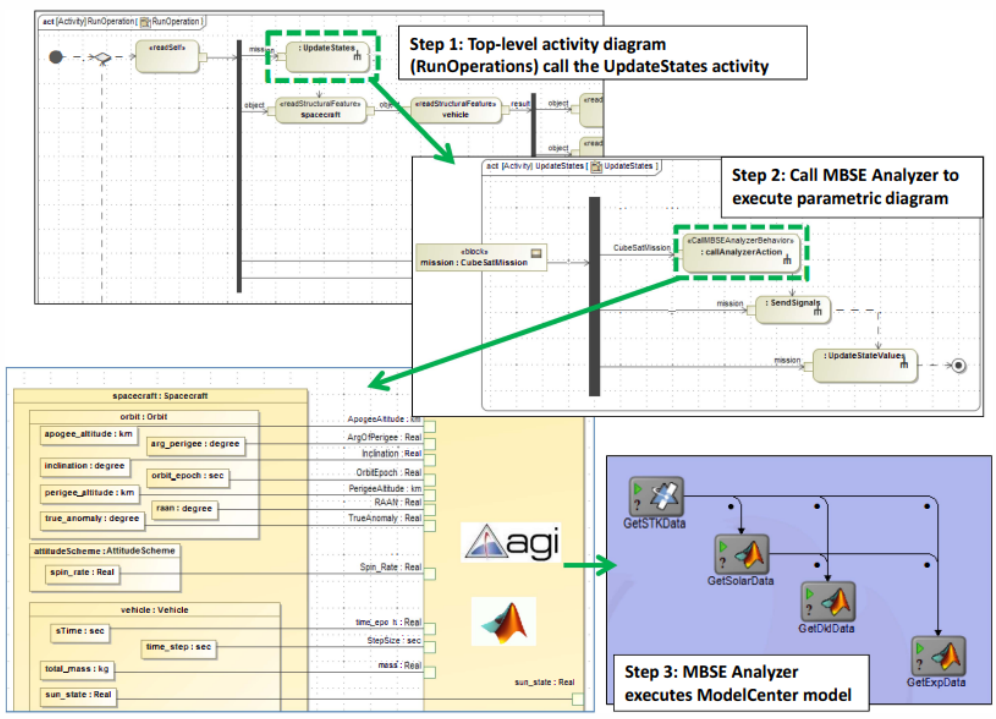
This section introduces the line of work from the INCOSE working group that is structuring MBSE into the CubeSat domain and served as a reference and starting point for this dissertation.

As a measure to reduce the failure rate of CubeSat missions, which is around 50% for university-class missions, when launch failures are factored out (SWARTWOUT; JAYNE, 2016), the INCOSE Space Systems Working Group (SSWG) began investigating the applicability of MBSE for designing CubeSats in 2011. They began by demonstrating MBSE in a CubeSat mission by creating a SysML model of a CubeSat and applying it to the Radio Aurora Explorer (RAX) mission (SPANGELO et al., 2012).

Further studies from this group employed parametric simulation of CubeSat missions and operational scenarios, integrating system models with simulation software such as Matlab and STK (KASLOW et al., 2014) (SPANGELO et al., 2013). This integrated M&S environment enables users to extract feasibility, performance and robustness metrics by visualizing representations of physical and functional states. The system modelling was performed using SysML in the MagicDraw tool, where the behavioural diagrams are made. The parametric modelling and integration with external tools was made using ModelCenter's MBSE Analyzer. Figure 2.5 shows an example of the steps taken from executing the behavioural models in SysML to the parametric models calling the external simulation tools, which is the process the

authors implemented.

Figure 2.5 - CRM Activity Simulation Steps Example.



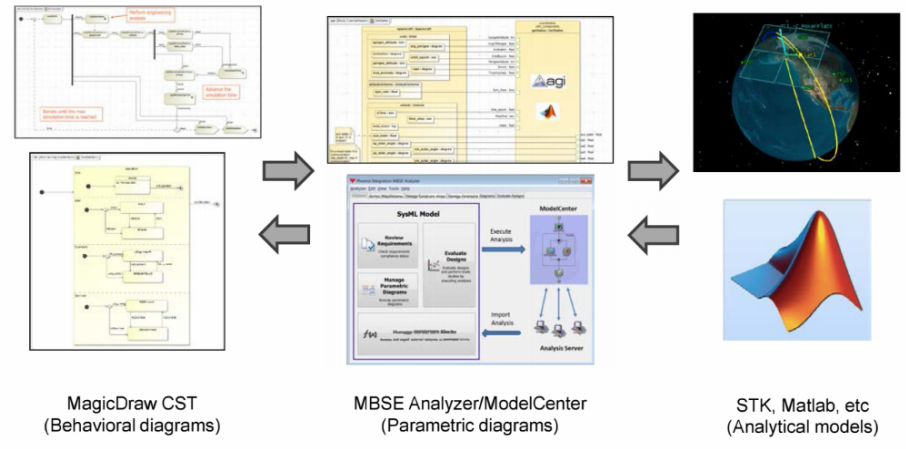
Source: Kaslow et al. (2014).

The relationship between the modelling and simulation tools to execute the system models and perform the analyses is shown in Figure 2.6. MBSE Analyzer calls the updates on the analytical models of the external simulators and integrate the results with each step on the behaviour SysML models. These simulations are very helpful in defining performance characteristics, budgets and constraints in subsystems and components specifications.
Further phases of this project have investigated the application of MBSE on defining the behaviour of CubeSats (KASLOW et al., 2017), and on the concept lifecycle phase (KASLOW, 2015).

The end product of this group is the under development CubeSat Reference Model (CRM), which they intend to release to the public as a reference model for other missions to use when defining the mission-specific model so teams can have a starting point when designing a mission, which will also serve as an Object Management Group (OMG) specification (KASLOW et al., 2018).

The developers define their objective as: *"The purpose of the CubeSat Reference*

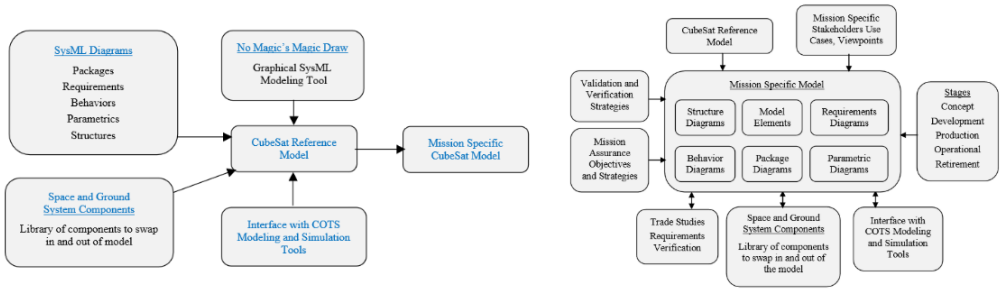
Figure 2.6 - CRM Modelling and Simulation Tools Integration.



Source: Kaslow et al. (2014).

Model is to provide a logical architecture, which serves as a guide and provides the building blocks for any CubeSat mission. The goal is to provide an object-oriented architecture framework so that teams can easily compose their CubeSat system and mission from the elements and objects found in the reference model" (KASLOW et al., 2016). The scope of the CRM is displayed in Figure 2.7, showing its relations with modelling and mission design elements.

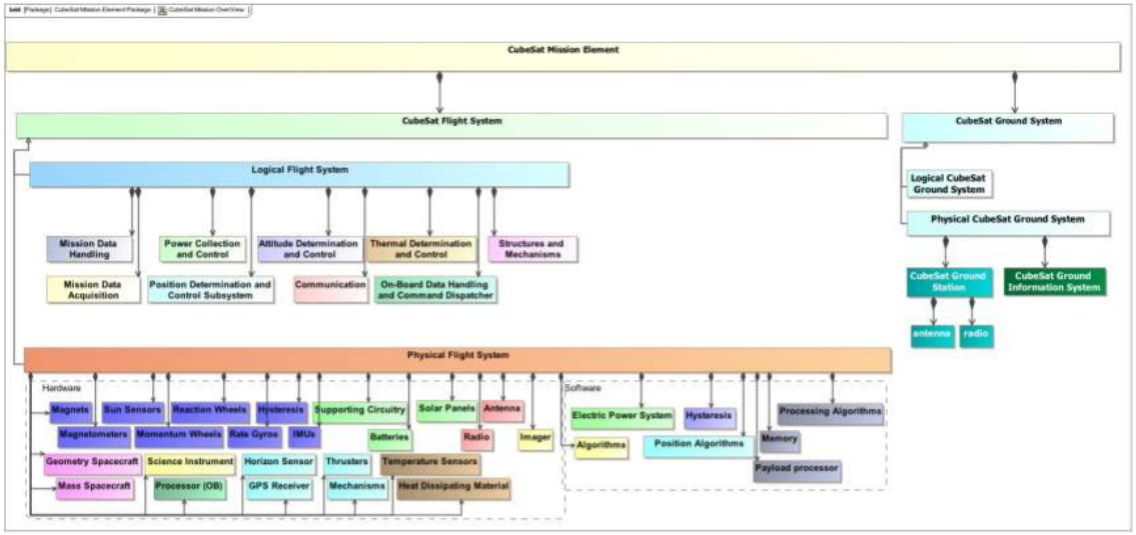
Figure 2.7 - CubeSat Reference Model Scope.



Source: Kaslow (2015).

The CRM consists essentially of a framework of multiple SysML packages and diagrams to describe the Logical and Physical Architecture of a standard CubeSat mission, along with behavioural models. An example of one of the diagrams provided by the CRM can be seen in Figure 2.8 that shows the logical architecture mission element of a generalized CubeSat system.

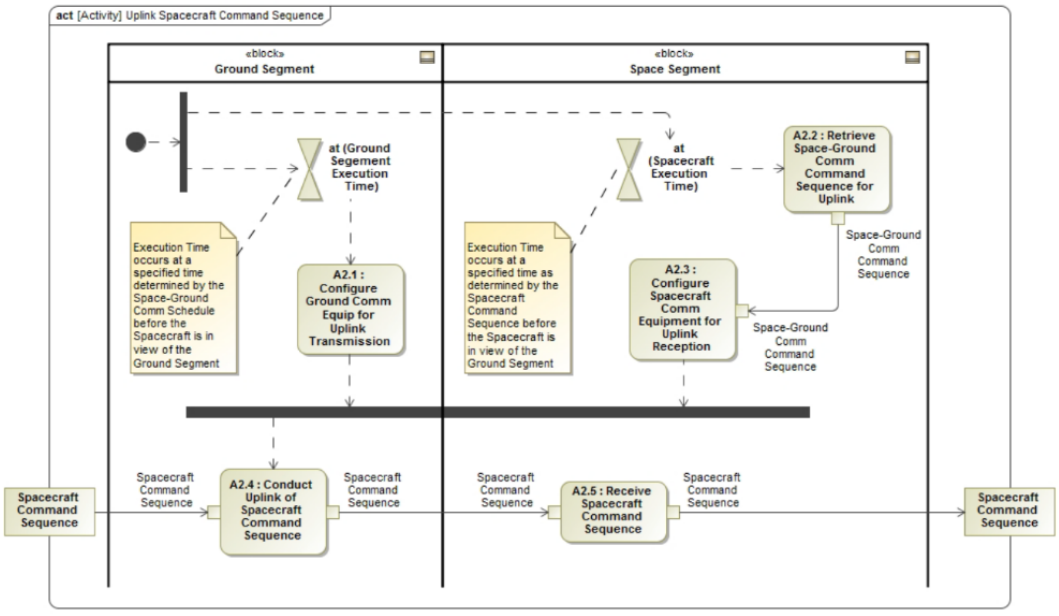
Figure 2.8 - CubeSat Mission Element Logical Architecture.



Source: Spangelo et al. (2012).

Activity diagrams as shown in Figure 2.9 are used to determine specific activities of the system. Together with Modes & States and Sequence diagrams, the behaviour of the system given determined scenarios can be modelled.

Figure 2.9 - Uplink Spacecraft Command Sequence Activity



Source: Kaslow et al. (2017).

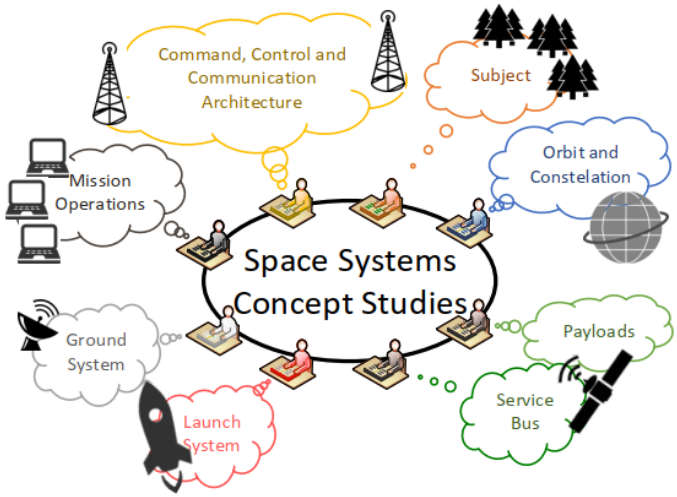
The CRM is still in development and is not yet available to the public. The papers provide good insight to the development process and the work shows to be very useful in the demonstrated applications. However, along with the barrier of entry of commercial software tools used, the author considers the project could benefit of open-source modelling and simulation tools and a guided methodological guidance for the m&s process. The author hopes the work of this dissertation can be beneficial in this sense.

2.5 Space missions concurrent engineering centers and concept studies

Space mission development typically begins with an initial phase called "Concept Studies". As stated by NASA's Systems Engineering Handbook (KAPURCH, 2010), it describes the study of stakeholder needs, demonstrates the feasibility of the desirable mission, and provides analyses of future resource allocation. These concept studies aim to establish mission goals, high-level requirements and functional descriptions, along with the concept of operations.

Concept studies are multidisciplinary studies and require exchange of information between specialists from many different disciplines that compose a space mission (WERTZ et al., 2011). The main disciplines are shown in Figure 2.10. The exchange of information between each discipline can become very complex since many parameters have multiple dependencies in different disciplines, raising the difficulty of design choices at the early stages and leading to multiple trade studies and design iteration loops.

Figure 2.10 - Space Systems Concept Study Main Disciplines.



Source: Cerqueira (2018).

Because of the elevated complexity and necessity of information exchange, concept studies are usually performed in facilities called Concurrent Engineering Centers (CEC). CECs provide the necessary infrastructure and team disposition to support the concept studies using the Concurrent Engineering approach.

Concurrent Engineering (CE) is a *systematic approach by diverse specialists collaborating simultaneously in a shared environment, real or virtual, to yield an integrated design* (HIHN et al., 2011). The approach was first applied to space system designs in 1995 JPL and since then it has been widely adopted in the aerospace industry and in government agencies due to its many benefits.

The Concurrent Engineering approach provides a collaborative, co-operative and simultaneous working environment (ESA, 2018) which by ESA standards is based on five key-elements (SCHUMANN et al., 2008):

- a process
- a multi-disciplinary team
- an integrated data/design model
- an appropriate facility
- a software/hardware infrastructure

Concurrent Engineering Centers are facilities that allow a physical allocation of a multidisciplinary team of experts along with tools and equipment that allow for rapid development of complex systems designs, based on the concurrent engineering approach. The co-location permits faster design iterations and better communication between the team members in comparison with separate offices, which shortens the decision making cycle and the overall design time (IWATA et al., 2015).

The facilities generally have a main room equipped with a network of computer stations, multimedia devices, whiteboards and software tools. The computers are dedicated to each discipline with their specific CAD tools and disciplines that are closely related and exchange more information (for example Thermal and Structural) are located closer to improve communication.

Many agencies and companies around the world have adopted the concept and built their own CEC, each having their own focuses and differences. Some examples are:

- ESA - Concurrent Design Facility (CDF)

- NASA/JPL - Team X, Team Xc, and Product Design Center (PDC)
- NASA/GSFC - Integrated Design Center (IDC)
- NASA/GRC - COMPASS
- The Aerospace Corporation - Concept Design Center (CDC)
- RAL Space - Concurrent Design Facility (CDF)
- INPE - Space Missions Integrated Design Center (CPRIME)

Figures 2.11 (a) and 2.11 (b) show studies being conducted at ESA's Concurrent Design Facility's (CDF) and INPE's CPRIME main rooms with team members allocated at their work stations.

Studies conducted in CECs are generally in the early stage of development (Phase 0), focusing on the conceptual design where there is a higher range of freedom and uncertainty and consequently a greater need for exchange of information between the disciplines. The studies generally last a couple of weeks, which is a dramatic reduction in cost and time in comparison with traditional concept studies which could take up to several months.

It is common for customers to participate in CEC sessions in order to provide inputs such as requirements and desired operation characteristics. Typical outputs from CEC studies are lists of diverse concepts, trade studies analysis, a more refined or detailed design, a set of requirements for a proposal, or even an evaluation of a specific concept.

Models are heavily employed in CECs to represent many different parts and subsystem of the system being designed (IWATA et al., 2015). Models help provide understanding of the system, exchange of information and drive discussions among the team members. Essentially, each discipline has their domain-specific model, and to group all the information there is an Integrated Design Model (IDM), which generally consists of spreadsheets.

Although Systems Engineering (SE) encompasses the entire life cycle of the system while CECs are tailored to focus on the early phases, the similarities between SE and CE allow for the advances in SE disciplines such as Model-Based Systems Engineering (MBSE) to make their way into CEC practices, as seen in the MBSE-CE integration survey by Iwata et al. (2015). This study shows that CECs are gradually adopting MBSE as the methodology and its tools evolve.

Figure 2.11 - Concurrent Engineering Centers.

(a) *ESA CDF.*

(b) *INPE CPRIME.*

At ESA's European Space Technology and Research Centre (ESTEC), the engineers have developed a system simulator to assist at early stage design studies at the CDF, called the System Concept Simulator (SCS) Kranz et al. (2015). By simulating the functional architecture of the system, it supports the trade-off studies for the system concept in an iterative manner, providing a quantitative assessment of the performance of the system for different mission and spacecraft concepts (ECSS, b).

2.6 Concept of operations

The Concept of Operations (CONOPS) is a description of the system's characteristics from an operational perspective. It describes how the system will operate to meet stakeholder expectations.

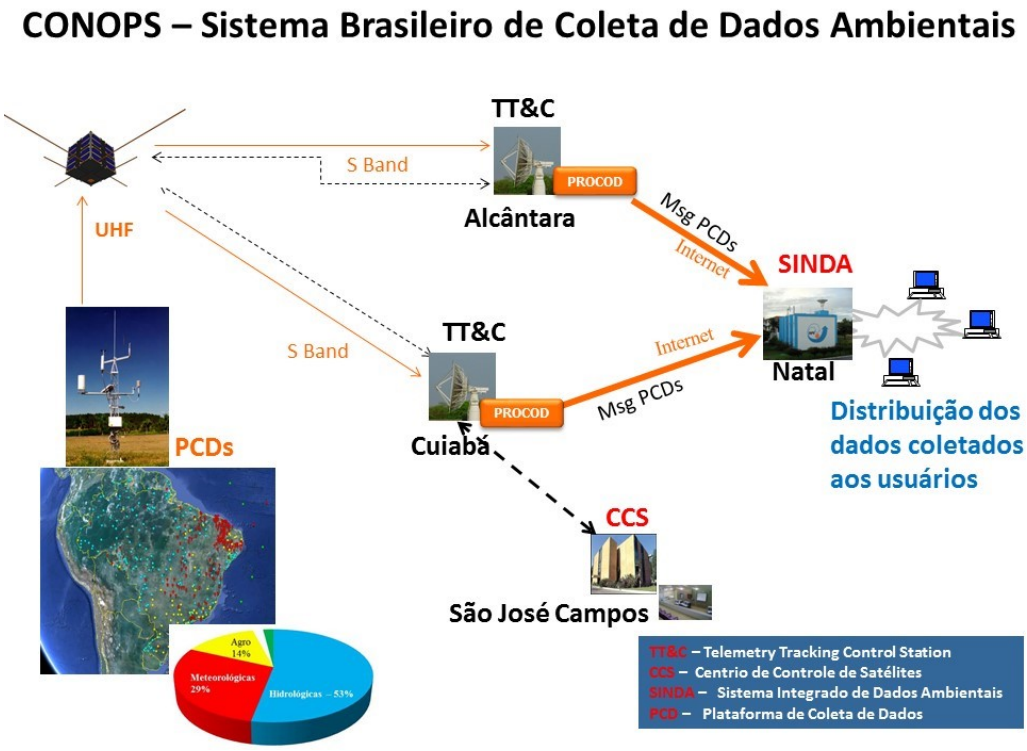
Defining the mission CONOPS in the early stages of development include: an initial physical and logical architecture of the space and ground segments; the interfaces between the elements of the architecture; mission objectives and constraints analysis; operation timelines, modes and scenarios; end-to-end communications strategy and data-flow; and especially power and data-budgets analysis. Addressing these items at this point is important to capture stakeholder expectations and elaborate system requirements.

Different institutions use different methods to construct the mission CONOPS. For the ECSS, the CONOPS is consolidated in the operations engineering process through a series of documents (ECSS, a). In the early stages, the CONOPS elements we desire to describe are addressed in the Operations Concept Document (MOCD), the Mission Analysis Report (MAR), and the Space Segment User Manual (SSUM).

Figure 2.12 shows an example CONOPS architecture of the Brazilian Data Collection System (SBCD) describing how the data retrieved from the multiple platforms across the Brazilian territory flows, starting by a UHF up-link to the spacecraft when in line-of-sight, that re-sends the mission data to the two Ground Stations at Alcântara and Cuiabá (the latter also receiving platform (housekeeping) data) through an S-band down-link. Mission data is then sent through the internet to the Integrated Environment Data System (SINDA) which archives and distributes the data to the end-users. The Cuiabá Ground Station is also remotely connected to the Satellite Control Center (CCS) at INPE in São José dos Campos, for satellite control purposes. Commands scheduled into the flight plan are uploaded during satellite visibility over that ground station.

Although in Figure 2.12 is presented an overview of the CONOPS, to describe it completely it is still necessary to describe its operation modes and scenarios, and other system characteristics, which are important in the design phase. This is often described through multiple figures with different and sometimes redundant viewpoints, and a large volume of text generated through many documents. The unification power of models provide an appealing approach to consolidate the information in a centralized manner, which is a key driver to the study of this dissertation.

Figure 2.12 - Brazilian Environment Data Collection System CONOPS Example - High Level Data Flow.



Source: Mattiello-Francisco (2019).

The satellite mission conception and definition guided by CONOPS is very useful for trade studies in Pre-Phase A and Phase A studies to establish spacecraft expected behaviour and initial design parameters and requirements, such as in the example study shown by Chagas et al. (2018) where INPE's Space Missions Integrated Design Center (CPRIME) counts on a simulation software that supports the engineering team to clarify some CONOPS aspects of the mission under analysis.

In the MBSE applications studied for this dissertation, such as (ESTABLE et al., 2017) and (KASLOW et al., 2014), among others, the concept of operations is generally modelled through use cases, activity, sequence and other behavioural diagrams. The implementations studied using SysML do not intuitively connect the equipment selection to the functions and capabilities performed, allowing for direct analysis in simulation trade-off studies. This is the direction explored in the approach of this study.

3 THE CONOPS2M MODELLING PROCESS

This chapter describes the modelling process Conops2M, the main product of this dissertation.

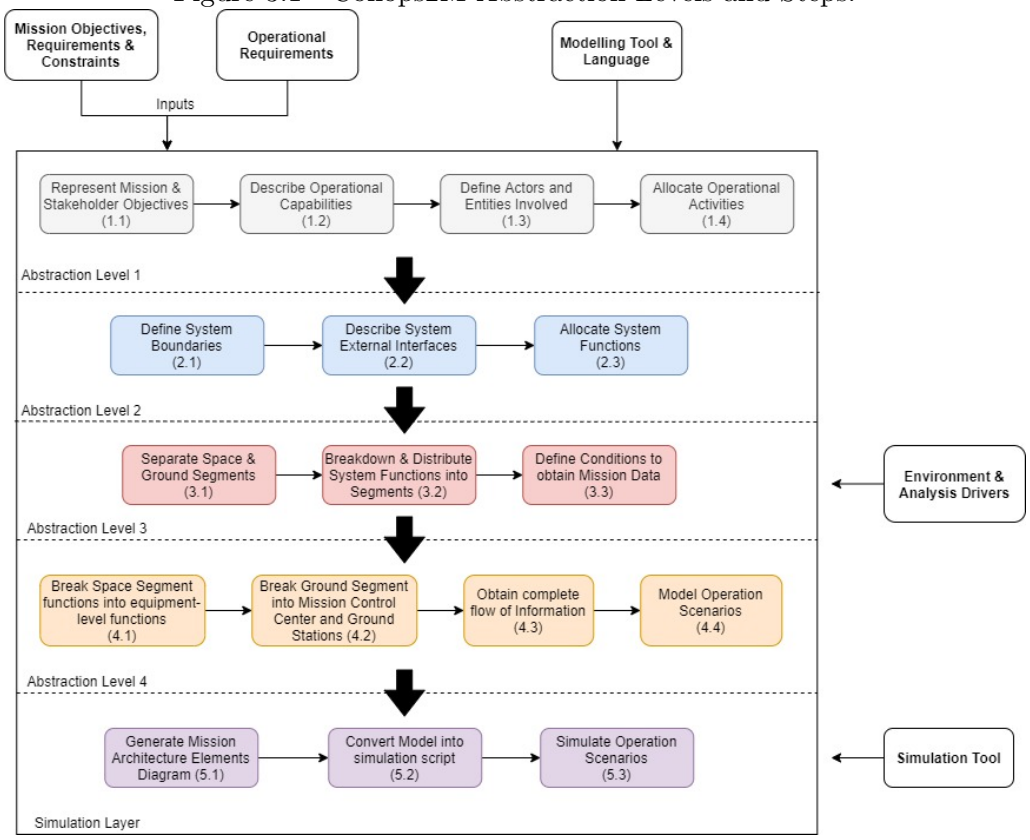
3.1 Modelling process description

The Conops2M modelling process consists of a set of sequential steps to be followed in order to generate a model of space mission concept of operations and prepare operation scenarios for simulation. The general idea behind the process is to begin at a high level of abstraction defining the mission, and then decompose activities and functions iteratively into lower levels of abstraction, reaching the equipment level aboard the spacecraft and ground stations. The steps are separated into different levels representing abstraction levels. The steps and levels can be seen in Figure 3.1, along with the external dependencies. The process requires as inputs the mission objectives, requirements and constraints, along with the operational requirements. The process requires a modelling tool with a modelling language to be executed. The simulation requires a simulation tool. The tool choices are left to the user, since the process is tool-independent. However, it is important to select both tools together in the beginning of the process in order to understand the interfaces between them and consequently how to export the model into the simulator.

Conops2M (especially the final level of abstraction) was developed on top of the "simplicity" and facilities provided by the CubeSat standard, bringing into context the functionalities of general subsystems common in CubeSats. However, the process could be applied for any class of single spacecraft space mission.

In the highest level of abstraction, the goal is to describe the mission itself, modelling as operational activities the events and phenomena that will happen for the mission to be achieved. The first step is to represent the mission & stakeholder objectives (1.1), inserting them into the modelling environment. Next, the user must derive and describe operational capabilities (1.2), what needs to be done in order to reach the mission/stakeholder objectives. The user must then define the actors and entities involved (1.3) in the operational capabilities and then define and allocate the operational activities (1.4) which the actors and entities perform to achieve them, describing the interfaces. Entities are things that have effects on the mission and will interact with the system, but do not have human elements (as Actors), such as the Earth, Sun, etc.

Figure 3.1 - Conops2M Abstraction Levels and Steps.



Source: Author.

In the second level of abstraction, the goal is to define the scope of the system solution and understand which of the operational activities of the previous steps will be converted to functions the system will perform, and model the interfaces to the external environment and end users. First the user must define the system's boundaries (which operational activities the system will perform – from data collection to data distribution, considering both space and ground segments together - 2.1), describe the system's external interfaces (how the system interacts with the world and the users - 2.2), and then allocate system functions (2.3) to the system by decomposing the relevant operational activities and creating new ones.

In the third abstraction level, the objectives are to separate and define the functions each of the space and ground segments will perform, and have an initial representation of how the mission data will be obtained and distributed. Initially the user separates the system into space and ground segments (3.1) and then must breakdown and distribute system functions into functions according to each segment (3.2). At this stage, the user must describe when/where the space (and/or ground) segment

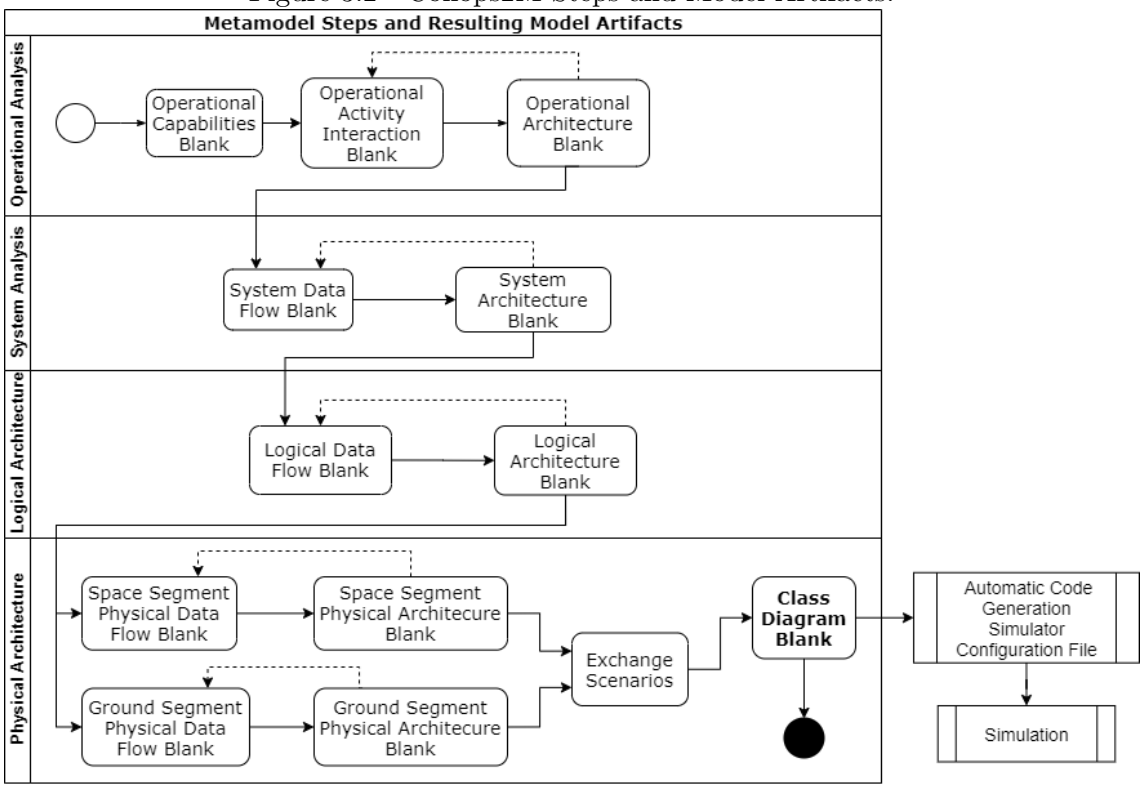
will collect the data relevant to the mission, or in other words, define conditions to obtain mission data (3.3).

The purpose of the fourth level of abstraction is to reach a representation of the space segment with all the equipment aboard and the functions they realize, a representation of the ground segment divided into the mission control center and the ground stations along with the functions they realize, and then model the sequence and conditions all of the functions will be performed routinely, describing operation scenarios. First, the user must decompose the space segment functions into functions individual equipment or subsystems will perform (4.1), depending on the grouping level the user wants to define and simulate the equipment. This step is usually iterative, leading to raising necessity of adding new equipment to handle unforeseen functions. The ground segment functions are then decomposed into functions for a mission control center and ground stations (4.2). The user must then connect all of these functions defining the information exchanged among them in order to obtain the complete flow of information involved in the functions sequences (4.3), from data collection to data distribution to the system users. This way, the user can finally model operation scenarios, describing sequences and conditions the functions are performed to meet the mission objectives (4.4).

In the last level of the process, the goal is to represent the final mission elements relevant to CONOPS simulation, transform the model into an input for a satellite simulator and then simulate the operation scenarios. The user may then create an architecture diagram (5.1) listing the spacecraft equipment along with other mission architecture elements that compose the mission CONOPS. At this point, the user should be able to configure the chosen simulator with the mission architecture elements according to the previously described operation scenarios. In the specific application shown in this paper, the author developed a plugin for the modelling tool that automatically transforms the created model's architecture diagram into a configuration script (5.2) for the chosen simulator considering each operation scenario, following the simulator's specific parameters and interfaces. The user can then finally simulate the operation scenarios (5.3) in order to validate the power and data budgets, equipment selection and usage, and other elements regarding the concept of operations.

The author's choice for modelling tool was the open-source software Capella (ROQUES, 2016), which comes with an embedded modelling method called Arcadia. Conops2M works very well within Capella/Arcadia since it follows a similar

Figure 3.2 - Conops2M Steps and Model Artifacts.



Source: Author.

abstraction level structure, allowing for separate diagrams for each abstraction level to be made. The process is adapted to the Arcadia method and is inserted into the four main steps of Arcadia, described by Roques (ROQUES, 2017). The author also finds the structured methodological guidance of Arcadia to have a good combination with the general facilitating idea of the CubeSat standard.

Capella comes with a domain-specific modelling language. Capella runs on Eclipse, and because it is open-source, the development of a plugin enhancing the tool to transform the model into the simulator configuration script was possible, and is described in 3.1.1. If another user decides to use another modelling tool, they will have to adapt the modelling process to use the diagrams enabled by the tool and language of choice. The artifacts that are generated inside Capella/Arcadia by following Conops2M can be seen in Figure 3.2, where each artifact is separated by the Arcadia steps where it is created. The artifacts are detailed in Section 3.2, and they are named following their names defined by Capella. The terms used to refer to Capella elements follow the definitions used by the Capella developers and can be found in the reference book (ROQUES, 2017).

3.1.1 Generating the simulation script

After the model is created, the user may proceed to simulate the operation scenarios. In order to do so, the user must configure the chosen simulator with the parameters that describe the mission, the spacecraft and its operation. In step 5.1 of Conops2M, the user generates an architecture diagram with the mission architecture elements based on the elements of the mission that are relevant for the simulation, so the diagram must be designed according to the inputs required for the chosen simulator. In our case, we create a Class Diagram with the mission elements modelled as Class instances, and translate it into a Julia language (JULIA..., 2020) script that is structured according to the input parameters needed for the ForPlan simulator.

Capella is built upon Eclipse, an open integrated development environment (IDE) which conveniently supports the development of plug-ins to extend the environment. To generate the simulation script, the author have developed, in partnership with the Fault Tolerant Systems Research Group (FTSRG) from the Budapest University of Technology and Economics (BME), through the ADVANCE project (LARANJEIRO et al., 2019), an Eclipse plugin for the Capella tool, called CapellaToForPlan (C2F). The plugin works by generating the configuration file and Julia code from specified *Class instances* inside the Capella Project, which are created using SysML-based Class diagrams.

Each mission architecture element is created as a Class instance, and has rules and internal elements that are detailed in Chapter 4. The plugin converts the classes along with their parameters, functions and properties into specific lines of codes according to how ForPlan is supposed to be initialized, when the file is executed. When the plugin is executed inside Capella, the configuration Julia language file for ForPlan is generated into a destination folder.

To create the class instances, we must use the Class Diagram Blank [CDB] in Capella. The reference template for a class diagram blank to configure ForPlan through C2F is shown in Figure 4.1.

3.2 Example model - Generic CubeSat mission

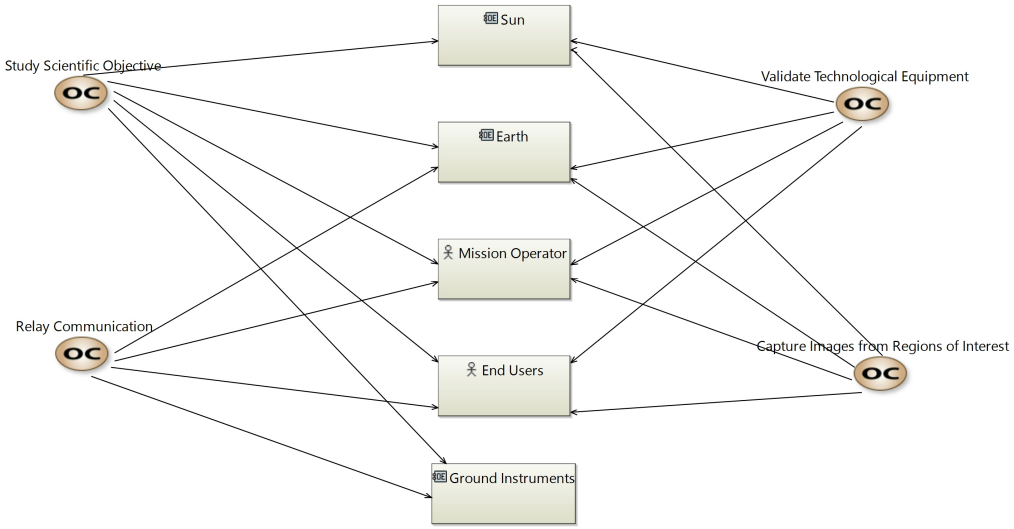
To illustrate the usage of Conops2M, in this section the author goes through the modelling of an example generic CubeSat mission, with the main traits of common missions of this type. The following subsections are divided according to each layer of the Arcadia method and show the resulting artifacts expected by following each

step of the modelling process.

3.2.1 Operational Analysis

In the first step, the Operational Analysis, the objective is to raise the operational goals of the stakeholders involved, or, in other words, "what the users of the system need to accomplish" (ROQUES, 2017). In this step, three diagrams are used: Operational Capabilities Blank [OCB], Operational Activities Interaction Blank [OAIB] and Operational Architecture Blank [OAB].

Figure 3.3 - Example Model Operational Capabilities Diagram.



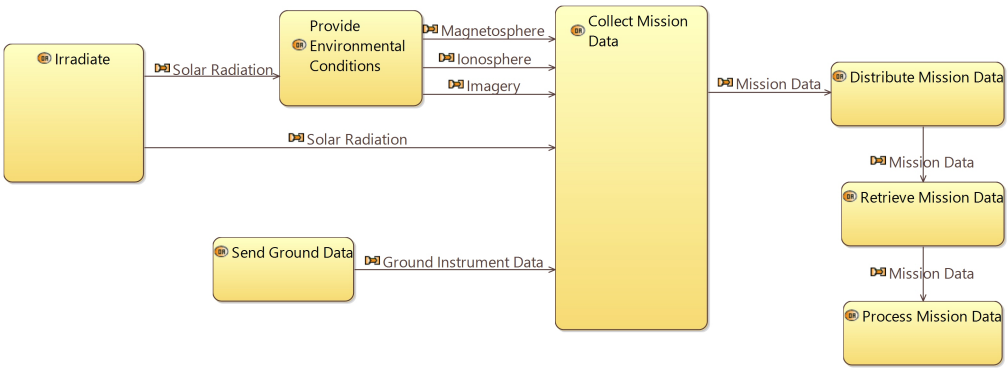
Source: Author.

In the OCB, the goal is to translate the mission objectives and list them as Operational Capabilities (OC), and link them to whichever actors and entities involved. In the generic model OCB, shown in Figure 3.3, four example OCs are provided, representing common LEO CubeSat missions: scientific (Study Scientific Objective OC), technological validation (Validate Technological Equipment OC), communication (Relay Communication OC) and remote sensing / imaging (Capture Images from Regions of Interest OC). It is important to represent the end-users, mission operator and other stakeholders involved, along with external entities that influence the phenomena under investigation or exploited.

After listing the OCs, the user must create and connect the operational activities, which are activities that will be carried out by the actors and entities previously listed, and then allocate them to the actors and entities using the OAIB and OAB di-

agrams, respectively. The operational activities and their interfaces should represent the flow of events of the phenomena of interest.

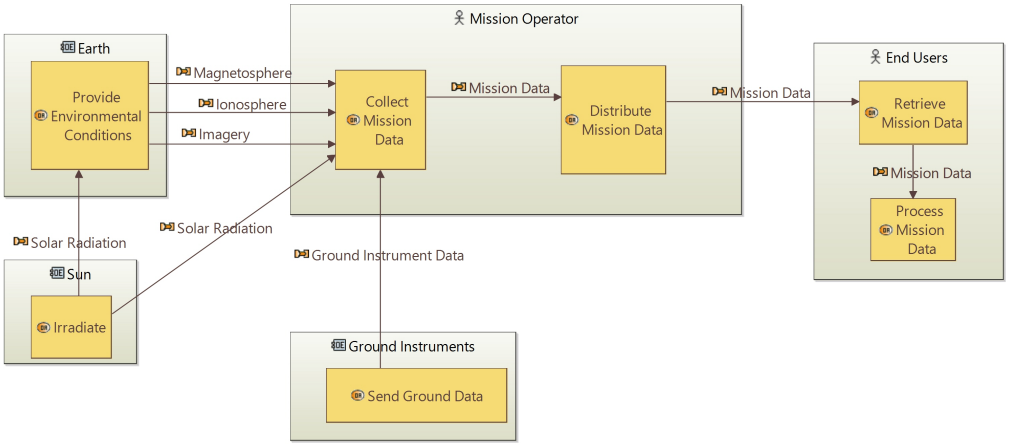
Figure 3.4 - Example Model Operational Activity Interaction Diagram.



Source: Author.

For the example model, Figures 3.4 and 3.5 show some activities common in LEO CubeSat missions, allocated to entities and actors. The Sun and the Earth are modelled as the entities which provide the interactions to be investigated. The activities which are not done by external entities or by the end users, are then allocated to the figure/Actor of the Mission Operator. This is the way which functions may be assigned to the system are defined for the next steps.

Figure 3.5 - Example Model Operational Architecture Diagram.

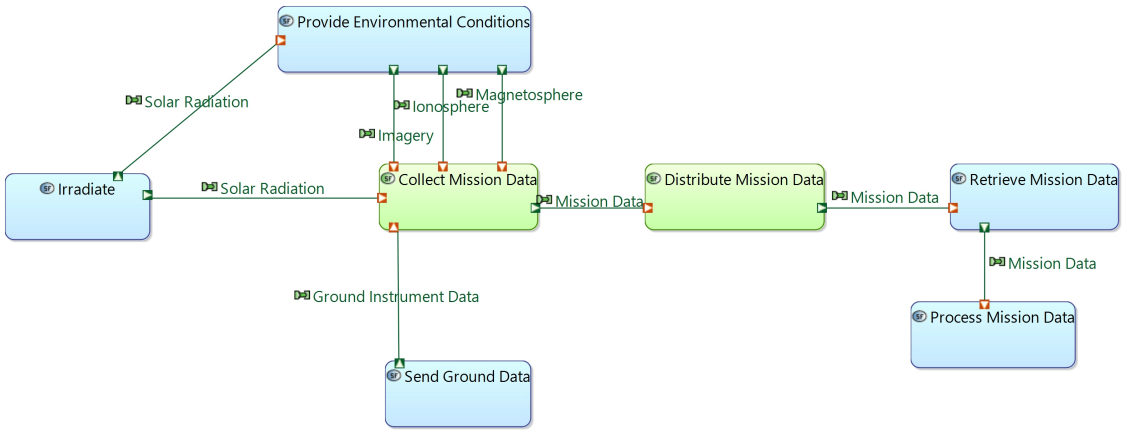


Source: Author.

3.2.2 System Analysis

The second step is a conversion from the operational activities to define the scope and borders of the system. The objective is to identify the functions the system will perform, and their interfaces with the external environment. To create the functions, the System Data Flow Blank [SFDB] diagram is used. The functions are then allocated to the system and actors using the System Architecture Blank [SAB]. The system is still treated as a black-box. In the example model, the System must collect mission data coming from interactions with the Earth environment, Sun, or ground-based sensors, and then distribute it to the end users, shown in Figures 3.6 and 3.7. The System is what we will begin describing in the next chapter.

Figure 3.6 - Example Model System Data Flow Diagram.

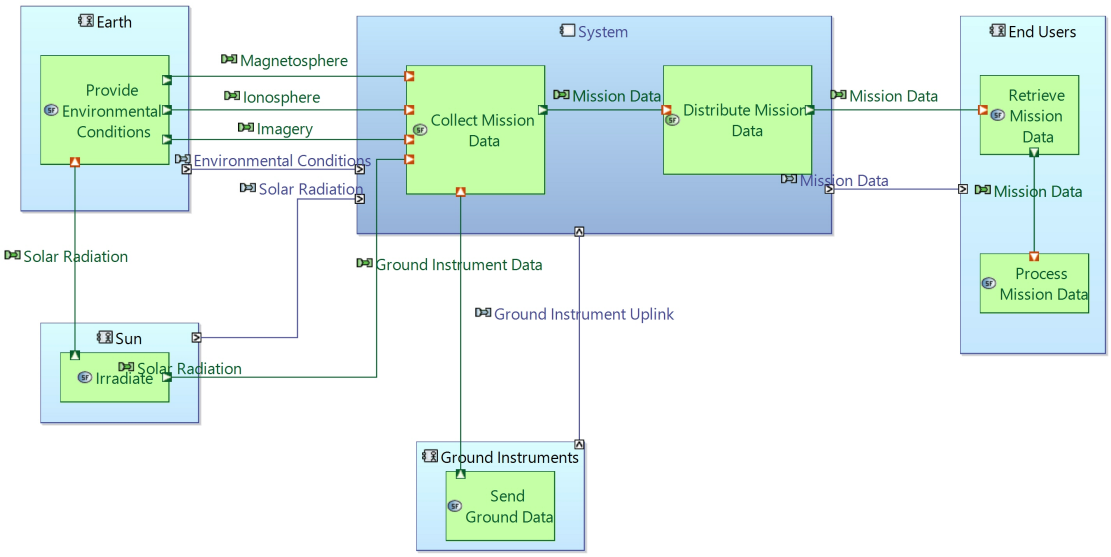


Source: Author.

3.2.3 Logical Architecture

In the logical architecture step, we begin to open up the black-box system to identify how the system will work to fulfil stakeholder expectations. Initially, the system is separated into Space Segment and Ground Segment. If the user decides to accept the ForPlan built-in functions to use in C2F (detailed in Section 4), the user now must identify which "data collection" or other functions will be done periodically, and which will be done over regions of interest (ROI). Otherwise, the user must detail the functions that will follow the simulator operation functions they will create. At this point, it is important to identify which functions of the space segment will need attitude control / pointing. The LDFB and LAB for the example model are shown in Figures 3.8 and 3.9, with the functions and the segments to which they

Figure 3.7 - Example Model System Architecture Diagram.



Source: Author.

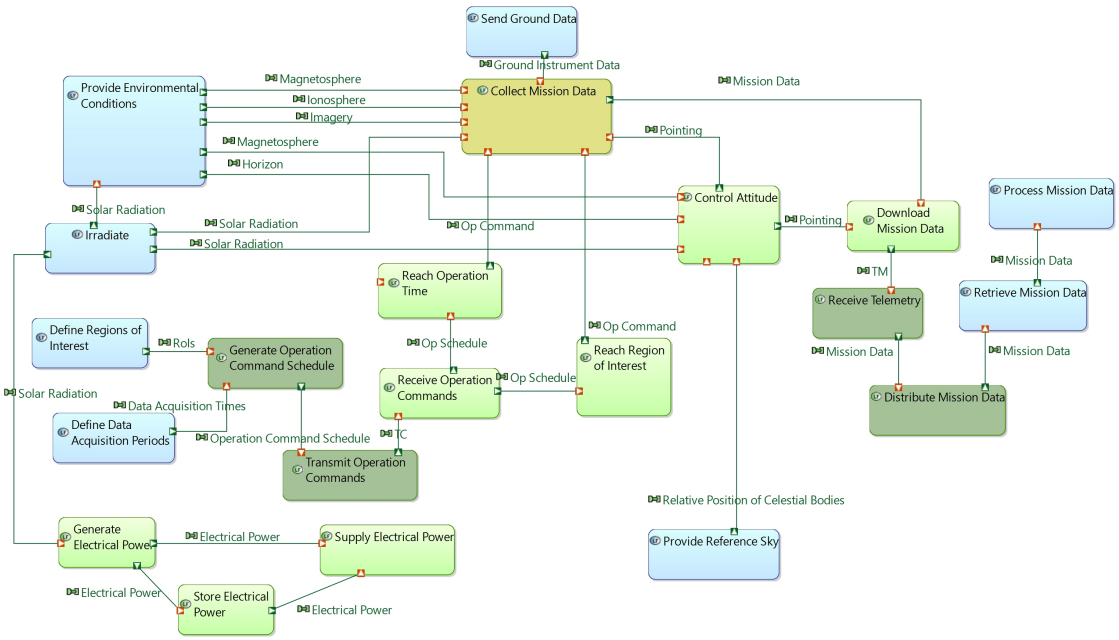
are allocated. In an instantiated mission model, each function of collecting a form of data or running an experiment/payload, for example, shall be declared individually, expanding the "Collect Mission Data" function in yellow in Figure 3.8, which is displayed as an intention to summarize functions of that kind.

3.2.4 Physical Architecture

In the Physical Architecture step, the final layer of abstraction, the concrete components of the system are defined. For each segment, functions are broken down into the relevant level of abstraction at this conception phase using the Physical Data Flow Blank [PDFB], which are then allocated to the equipment declared as behaviour components in a Physical Architecture Blank [PAB]. For the Space Segment, the functions must be broken down to the level of the equipment that will be on-board the spacecraft. For the Ground Segment, the functions will be assigned to either Ground Stations, or the Mission Control Center.

In the Space Segment PDFB, it is important to represent details such as how the spacecraft will detect when it enters operation regions of interest of periods (manage time & position), how it will determine and control it's attitude, how it will generate and supply electrical power, how it will receive and transmit operation commands, and other basic platform functions. In this diagram, the user must break down the "collect mission data" functions into lower level functions by describing the payload operation to the equipment level. Figure 3.10 shows an example set of equipment-

Figure 3.8 - Example Model Logical Data Flow Diagram.



Source: Author.

level functions for a CubeSat mission.

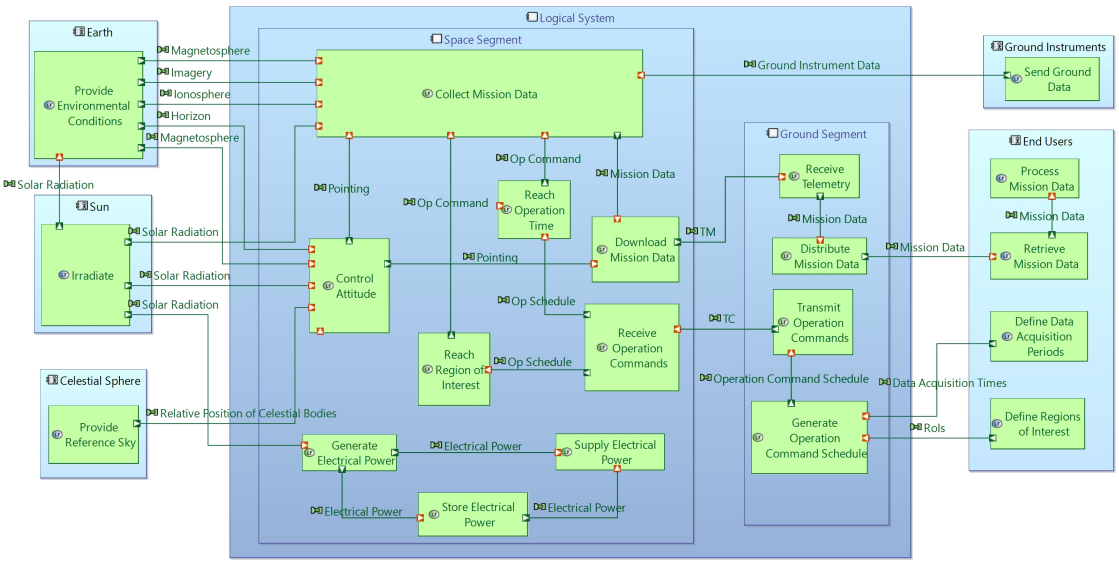
The PAB for the Space Segment is then used to create components for each equipment and allocate to them the functions previously created. This can lead to an iterative process while discovering the need to break functions into lower levels to assign them to individual equipment. In the end, the objective is to have a visual layout / architecture of the equipment and which functions they will perform. An example architecture artifact of a basic CubeSat platform is shown in Figure 3.11.

A second PDFB is then created to represent the ground segment functions. This step is important in order to understand how the commands are generated and sent to the space segment, and how the telemetry is received and distributed to the end users.

The Ground Segment PDFB is used to create the functions, and the Ground Segment PAB is used to allocate these functions to the Ground Stations or Mission Control Center components. Figures 3.12 and 3.13 show an example PDFB and PAB with basic functions of a Mission Control Center and Ground Station relevant for this level of analysis for a CubeSat Mission.

As a last element to model how the system will operate, an Exchange Scenario [ES]

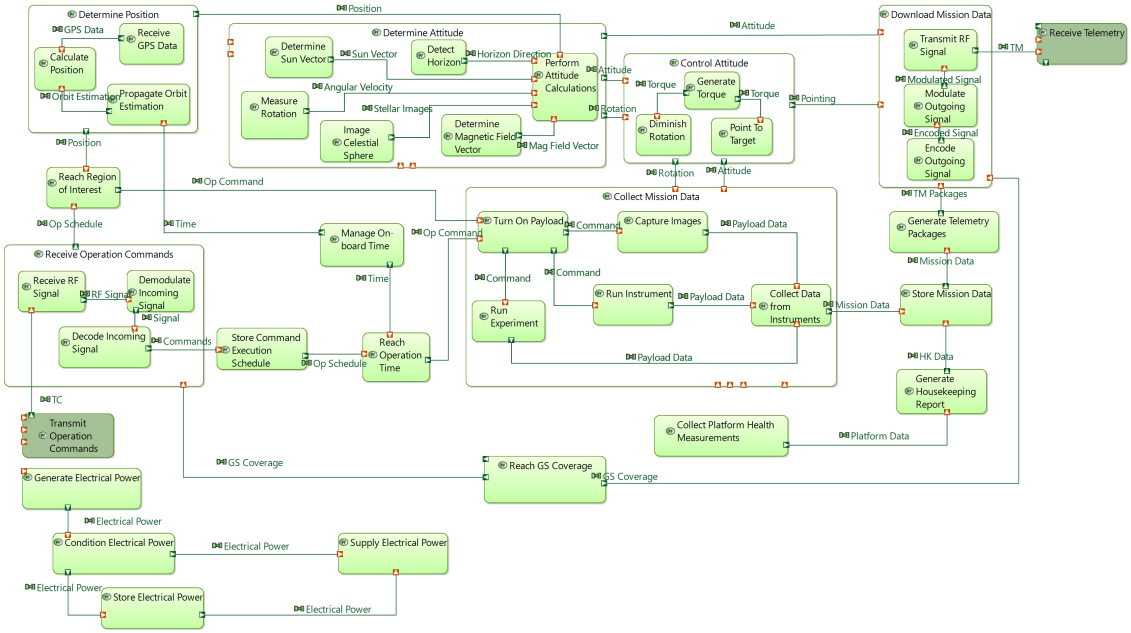
Figure 3.9 - Example Model Logical Architecture Diagram.



Source: Author.

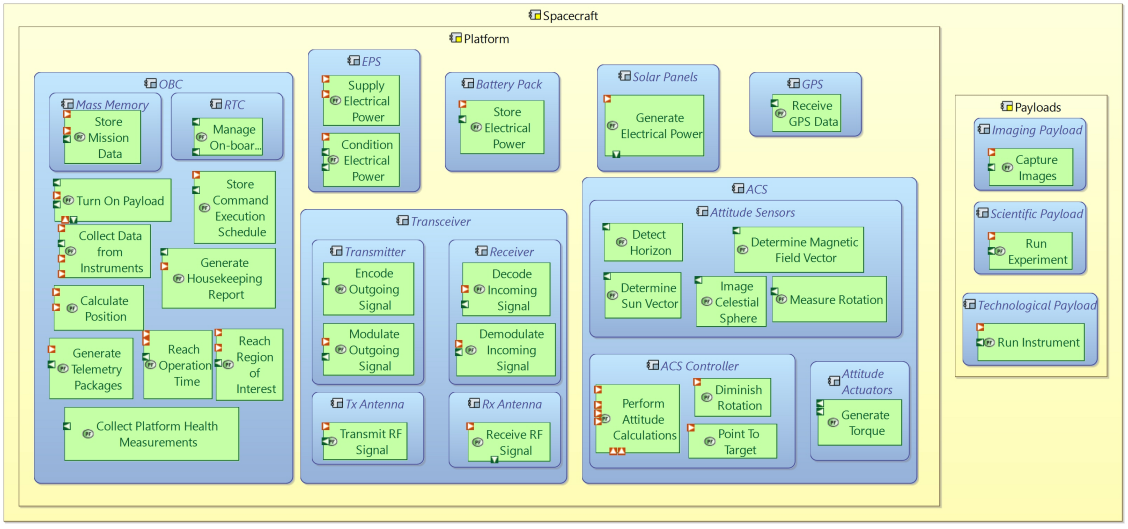
is proposed to describe the sequence of functions for each operation scenario expected. The user should include the relevant elements for the operation scenario, and detail sequentially the functions and the interactions among them. The idea is to represent how the spacecraft would operate routinely along the multiple orbits, with respect to platform and payload operation. An example ES is shown in Figure 3.14. The ES should show the complete flow of information in an operation scenario, beginning in the ground segment by generating the operation schedule, passing through the spacecraft alternating between payload operation depending on which region of interest or operation period is reached, and then all the way back to the ground segment to be retrieved from the database by the end user. By creating an ES as such, sequential and/or temporal constraints of equipment or payloads can be described and better understood.

Figure 3.10 - Example Model Space Segment Physical Data Flow Diagram.



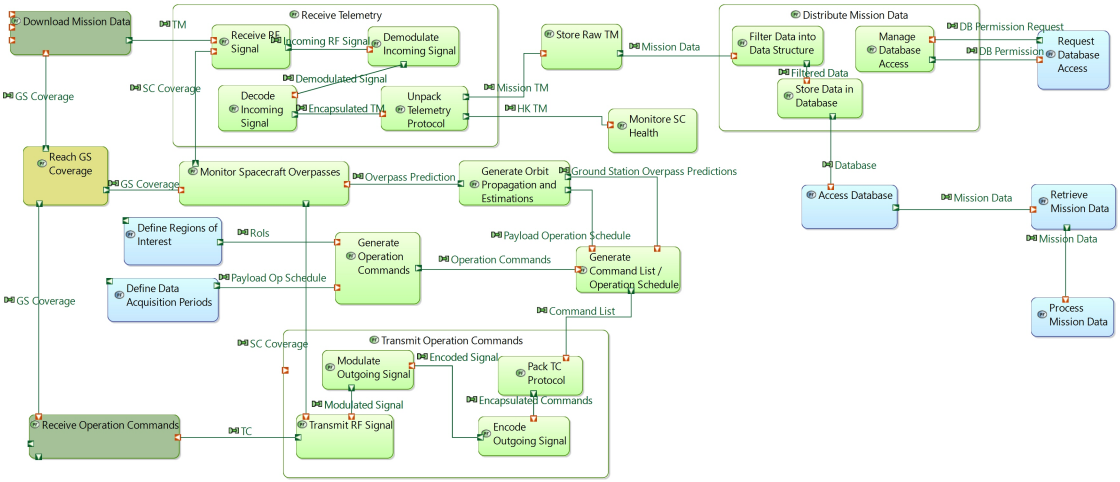
Source: Author.

Figure 3.11 - Example Model Space Segment Physical Architecture Diagram.



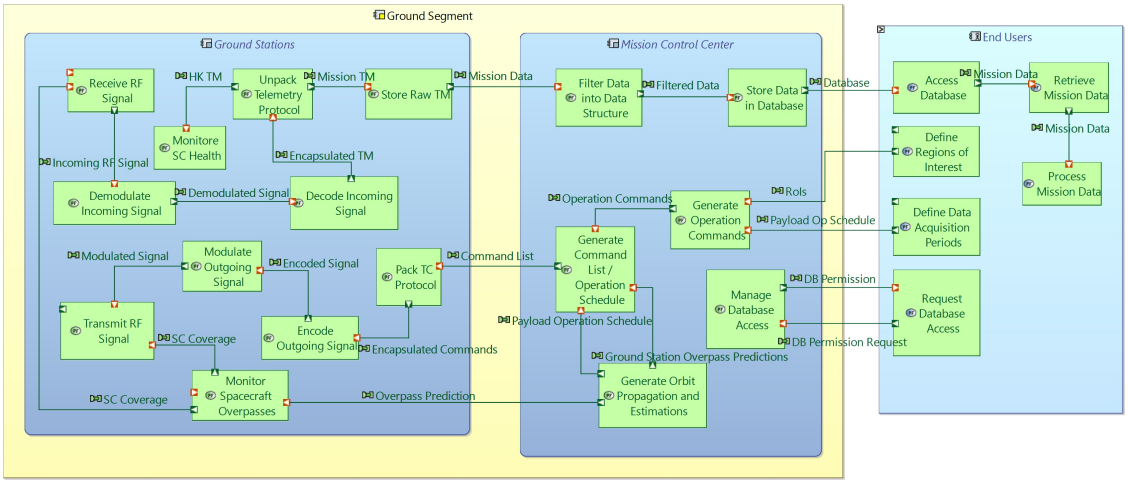
Source: Author.

Figure 3.12 - Example Model Ground Segment Physical Data Flow Diagram.



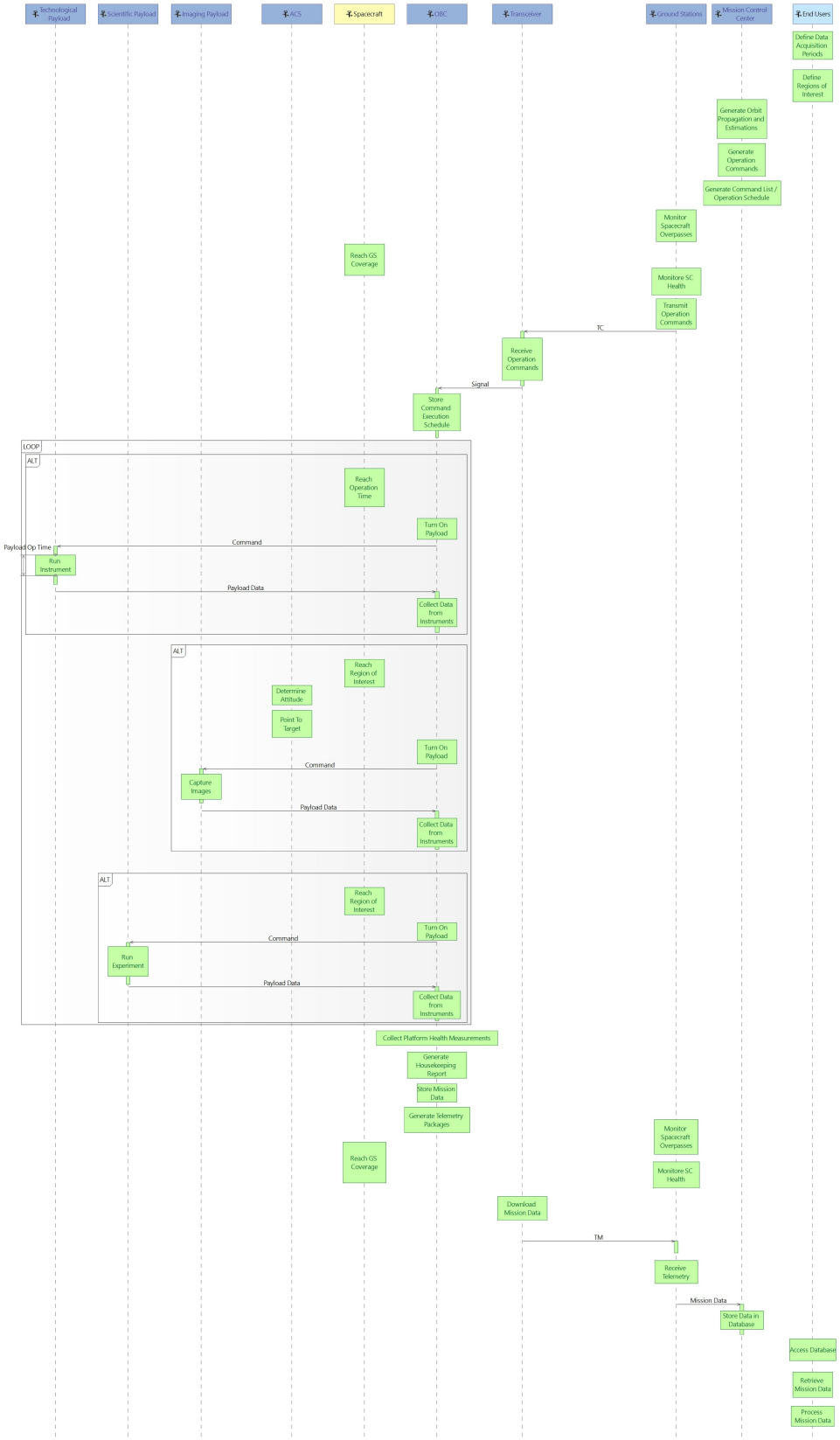
Source: Author.

Figure 3.13 - Example Model Ground Segment Physical Architecture Diagram.



Source: Author.

Figure 3.14 - Example Model Operation Scenario Example Exchange Scenario.



Source: Author.

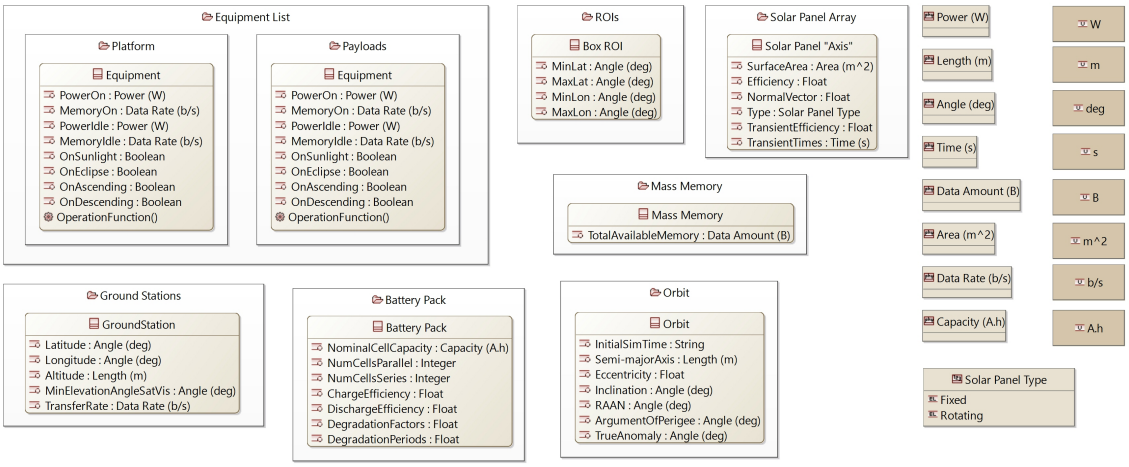
4 THE CAPELLATOFORPLAN (C2F) ECLIPSE PLUGIN

This chapter describes the way the CapellaToForPlan (C2F) plugin processes the Capella artifacts and transforms the model into the Julia language configuration file. As aforementioned, C2F searches for Class-type instances inside the project. The Class instances must be inside the Data folder under the Physical Architecture step, and are separated into specific *data packages*.

Each package is dedicated to a specific set of configuration parameters for the For-Plan Simulator which must be declared in the configuration file. In order to create the *class instances*, a Class Diagram Blank [CDB] is used. Inside the CDB, each *data package* must be created with the predefined hard-coded names. Using hard-coded names is a limitation due to Capella not supporting multiple meta-levels for modelling. The best choice would have been to define a metamodel where these kinds of elements (types) could be defined, and then these elements could be instantiated with the appropriate attribute values (instances). This is not supported in Capella, therefore we are unable to include selection of previously defined packages, classes and properties, obligating the user to create each object with the hard-coded names.

The data packages can be seen in Figure 4.1 as the larger objects containing the class elements. A template Class Diagram to use to configure ForPlan, showing all the data packages and classes with their respective properties, is shown in Figure 4.1. The user must simply instantiate all the equipment and other classes, following the guidelines provided.

Figure 4.1 - Simulation Configuration Class Diagram Blank.



Source: Author.

Each class has a set of specific *properties*, also with hard-coded names. The value of each *property* is defined in its *Default Value* parameter. To guide user instantiation, the template comes with pre-defined units for each of the physical quantities used, as seen in Figure 4.1. For example, all power consumption properties should be declared in Watts (W), angles in degrees (deg), and so on. These units are the units used in the ForPlan simulator. If changed, the user might find errors.

Every equipment on-board the spacecraft, must be declared inside the *Equipment List* data package, either in the *Platform*, or in the *Payloads* data package. The equipment has properties which define: power consumption when turned on (*PowerOn*); data generation when turned on (*MemoryOn*); power consumption when idle (*PowerIdle*); data generation when idle (*MemoryIdle*); if equipment works when exposed in sunlight (*OnSunlight*); if equipment works when in eclipse (*OnEclipse*); if equipment works on ascending part of orbit (*OnAscending*); if equipment works on descending part of orbit (*OnDescending*).

Each equipment also has an *OperationFunction*. This function, which is the central piece of defining the spacecraft's operation, defines when the equipment will be turned on, consuming electrical power and generating mission data. The simulator has four built-in operation functions from which the user can choose to define the equipment. If one of these functions does not satisfy the specific operation of an equipment, the user with access to the simulator source code can create their own specific function to better define that operation. These functions are generally defined internally in the equipment models of the simulator. However, it is possible to define new functions in the configuration script, which access to the variable workspace of the current state of the simulation. For the C2F plugin to add new user-defined functions, at this point it would be necessary to write them into the C2F plugin's source code.

The built-in functions are described as following:

- **AlwaysOn** Equipment is always turned on in a constant state.
- **OnGroundStation** Equipment is turned on whenever above the ground stations listed as parameters. If no parameters are listed, then all ground stations are used.
- **TimedOp** Equipment is periodically turned on. Parameters define on-time and off-time.
- **RoiOP** Equipment is turned on whenever above regions of interest, which

are defined as a parameter array of ROIs defined in the ROIs data package.

The Regions Of Interest (ROIs) are defined as classes inside the *ROIs* data package. For now, ROIs can only be defined as "box" areas defined by maximum and minimum latitudes and longitudes properties. Future implementations are expected to expand this functionality by adding interfaces with map APIs, for example.

The *Solar Panel Array* data package is where the solar panels must be instantiated. Each panel must be instantiated through an individual class with properties such as *SurfaceArea*, conversion *Efficiency*, its *Type* (defining if it's static or rotating). If the panel is static, its plane is defined by its *NormalVector* with respect to the satellite's reference frame. If the panel is defined as Rotating, it is defined by *RotatingVector*, which in this case the simulator positions the panel to maximize Solar incidence respecting the vector of rotation.

Each Ground Station is defined as a class instance inside the *Ground Stations* data package. Their properties are *Latitude*, *Longitude*, *Altitude*, minimum elevation angle for satellite visibility (*MinElevationAngleSatVis*), and data transfer baud rate (*TransferRate*).

The amount of mass memory available in the spacecraft to store on-board data is defined as a single class with a single property *TotalAvailableMemory* in the *Mass Memory* package.

The spacecraft's battery is defined in the *Battery Pack* package. A single class instance is used. Its properties are the cells' capacity, the number of cells in parallel and in series, the charge and discharge efficiency, and the degradation factors and periods. In future developments, it will be possible to define parameters for the battery charge & discharge functions (which is already editable in the simulator source code). In the meantime, it is limited to a common lithium-ion type function described in (CHAGAS et al., 2018), since this is a very common type of battery used in CubeSat-based platforms.

Finally, the *Orbit* package is where the Orbit class is defined, where the orbital elements and initial simulation time are inserted as properties.

After all the configuration elements are instantiated and their property values are filled in, the user may generate the Julia file for the simulator configuration script through a button in the Eclipse Project Explorer generated by the C2F plugin.

Trade-off studies can be made by generating different configuration scripts for different equipment or properties under analysis. For example, if an analysis on the relationship between the amount of data that is generated by a payload and the amount that can be downloaded to earth if required, we can define different operation scenarios by determining different regions of interest or operation periods for that specific payload, where each of these operation scenarios will result in different values for the operation function parameters (different ROIs or on-time/off-time). For each condition in the trade-off analysis, the user defines these parameters in the CDB.

5 MODEL INSTANTIATION CASE STUDY: NANOSATC-BR2

5.1 Mission description

NanosatC-BR2 (NCBR2) is the second satellite of the NanosatC-Br program, developed at INPE in cooperation with the Federal University of Santa Maria (UFSM) (SCHUCH et al., 2017). NCBR2 is a 2U CubeSat that will carry three scientific/technological payloads (SLP, SDATF & SMDH).

NCBR2 is a scientific and technological mission, aiming to:

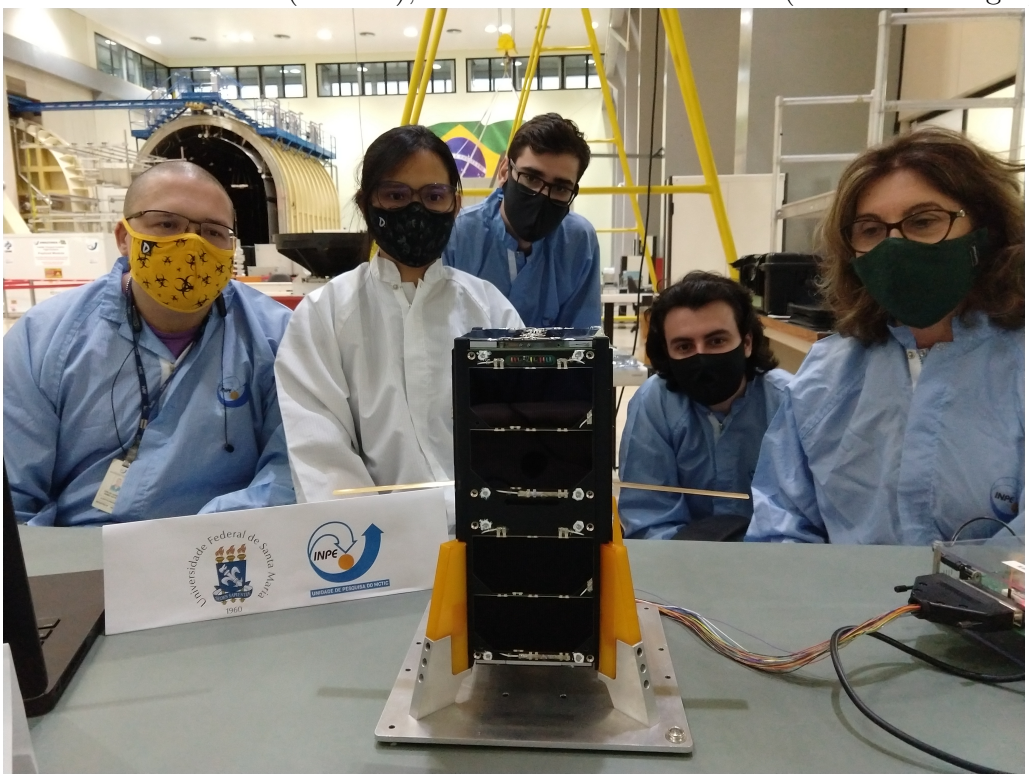
- collect data to better understand the Magnetic Anomaly of the Southern Atlantic (SAMA) (SLP Payload & Magnetometer)
- collect data to better understand the formation of plasma bubbles in the ionosphere (SLP Payload)
- validate in-orbit a fault tolerant attitude determination system (SDATF Payload)
- validade in-orbit a radiation tolerant FPGA and ASIC system (SMDH Payload)
- develop human resources with experience in space mission development

The mission will reuse the ground segment infrastructure inherited from INPE's NCBR program, which is a ground station at Santa Maria, providing UHF/VHF downlink and uplink capabilities, and a mission control center. The mission will also include a ground station at Natal, offering the same capabilities to act as an additional channel for data download and command upload.

Figure 5.1 shows a picture of the NanosatC-BR2 moments after completing the Assembly, Integration and Tests (AIT) campaign at INPE's AIT facility LIT, with some of the team members. Shortly after, it was sent to the launch service provider for preparation and future integration on the launch vehicle.

The spacecraft is divided into two modules: the Bus module and the Payload module. The Bus module contains all the subsystems necessary for the operation of the spacecraft, such as: the On-Board Computer (OBC), Electrical Power Supply (EPS), Telecommunications (Transceiver + UHF/VHF Antennas), and the Attitude Determination and Control Subsystem (ADCS). The Payload module contains the three physical payloads, which are: a Langmuir Probe (SLP), a Fault-Tolerant Attitude Determination System (SDATF), and an experiment board carrying a rad-hard

Figure 5.1 - NanosatC-BR2 moments after the successful AIT campaign - Dec 2020
From Left to Right: Carlos Batista, Dr. Jenny Asencio (LIT), Lucas Hein, Danilo Pallamin de Almeida (Author), and Dr. Fatima Mattiello (Mission Manager).



Source: Author.

ASIC chip and a fault-tolerant FPGA (SMDH Payload). The two software payloads on-board the OBC are a telecommunications protocol and service for the Brazilian amateur radio community, and the attitude stabilization algorithm.

In CubeSat missions, the power and data budgets are common bottlenecks for equipment design and operations due to the constraints brought by the standard's simplifications: CubeSats typically have limited size solar panels on the external surfaces of the spacecraft, instead of sun-pointing deployable arrays, limiting the amount of power harvested from the Sun. With respect to communications, CubeSats typically use amateur radio frequencies, limiting the baud rate capabilities.

The NCBR2 mission has 6 solar panels, 1 on each side of the spacecraft (4 2U side panels and 2 1U top/bot panels). The 2U side panels generate a maximum of 4.8W with the expected incidence of Solar Radiation, while the top and bottom solar panels generate a maximum of 2.4W. With all equipment and payloads turned on, NCBR2 consumes 4W of electrical power, which would demand 6Wh of energy

considering a 90 minutes orbit which is expected for the NCBR2 (LEO polar). Considering an average 60 minutes of Solar exposure time for each 90 minute polar orbit, at peak generation (which is highly unrealistic considering spacecraft rotation and NCBR2 does not have Solar pointing capability) the spacecraft would harvest 4.8Wh of energy, leading to a negative power balance and raising the necessity of specific operations for the payloads, and therefore the simulation of operation scenarios to reach a viable solution, especially considering the rotation of the satellite leading to different incidence angles of Solar radiation on the solar panels.

Regarding the data budget, NCBR2 can produce per orbit over 600KB of data with maximum generation of all payloads and housekeeping telemetry. Considering 90 minutes LEO orbits, that would result in 9.6 MB of data per day. An initial estimate for downlink capability can be made considering a LEO polar orbit, which would have per day around 4 overpasses with an average time of 10 minutes each above INPE's ground station at Santa Maria - Brazil, based on the similar orbit of NCBR1. NCBR2's transmitter can operate at 1200, 2400, 4800 or 9600 bps, leading to a maximum of 2.8MB per day that theoretically could be downloaded to a single ground station. Considering the mission's other ground station at Natal, they would have overlapping zones, so it is important to simulate operation scenarios to limit payload data generation and generate an estimate of data to be downloaded.

Table 5.1 - Maximum Energy Demand & Data Volume Generation and Existing Budgets

Full Operation Daily Energy Demand [Wh]	6
Max Daily Energy Generation [Wh]	4.8
Full Operation Data Generation [KB]	9600
Max Daily Downlink Volume - 1 GS [KB]	2800

Table 5.1 shows the comparison between expected energy demands for an operation considering all payloads turned on with the expected data generation (Full Operation), and expected maximum daily power generation and maximum daily downlink volume. Since the budgets aren't met, it is needed to balance payload operation to see if the budgets can be met considering the stakeholder objectives.

5.2 The NanosatC-Br2 CONOPS model

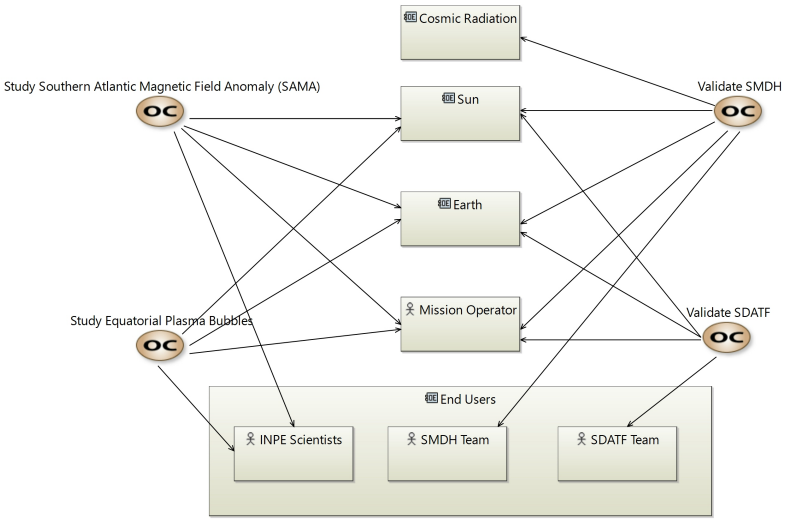
In this section, the model for the NCBR2 mission generated following Conops2M is described. The model was created with the purpose of generating trade studies an-

alyzing viable operation duration of each payload regarding power and data budget constraints.

5.2.1 Operational Analysis

We begin by creating the Operational Capabilities Blank (OCB), shown in Figure 5.2, where we represent the four mission objectives that use the physical payloads, and the actors and entities involved. Since the attitude control and amateur radio software payloads are only secondary objectives and are not relevant to the main objectives of the mission, we leave them out of this model for simplification purposes.

Figure 5.2 - Instantiated Model Operational Capabilities Diagram.



Source: Author.

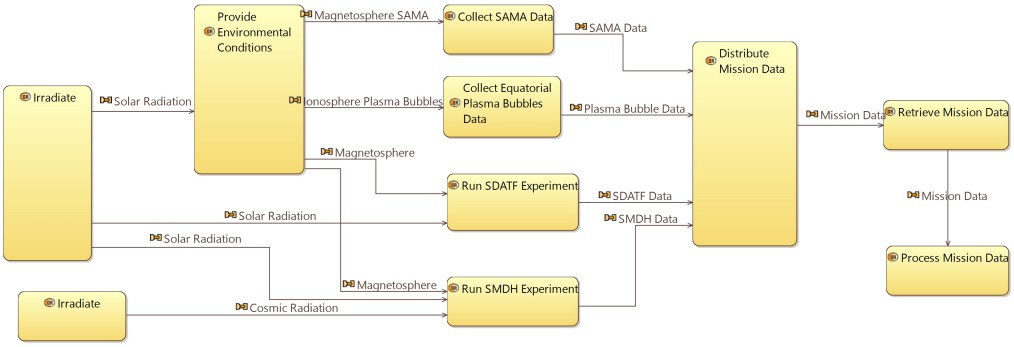
We then model the activities expected from the actors and entities and their interfaces using an Operational Activities Interaction Blank, shown in Figure 5.3.

The activities are then allocated to the entities and actors using the Operational Architecture Blank (OAB), as shown in Figure 5.4.

5.2.2 System Analysis

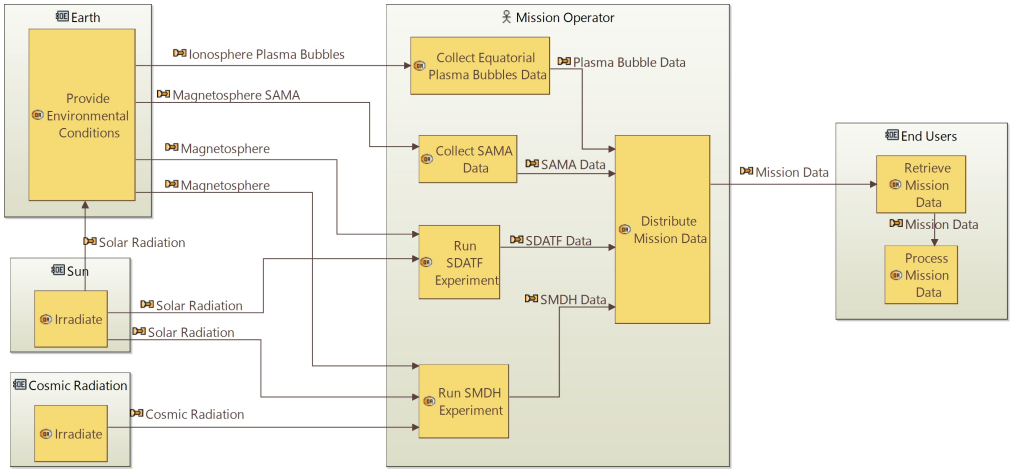
The mission operational activities are converted into system functions in the second step, using the System Data Flow Blank (SDFB). We allocate the functions to the actors, entities and components involved using the System Architecture Blank, shown in Figure 5.5. We now have the boundaries delimiting what functions the system will perform, and the interfaces with the external environment.

Figure 5.3 - Instantiated Model Operational Activities Interaction Diagram.



Source: Author.

Figure 5.4 - Instantiated Model Operational Architecture Diagram.



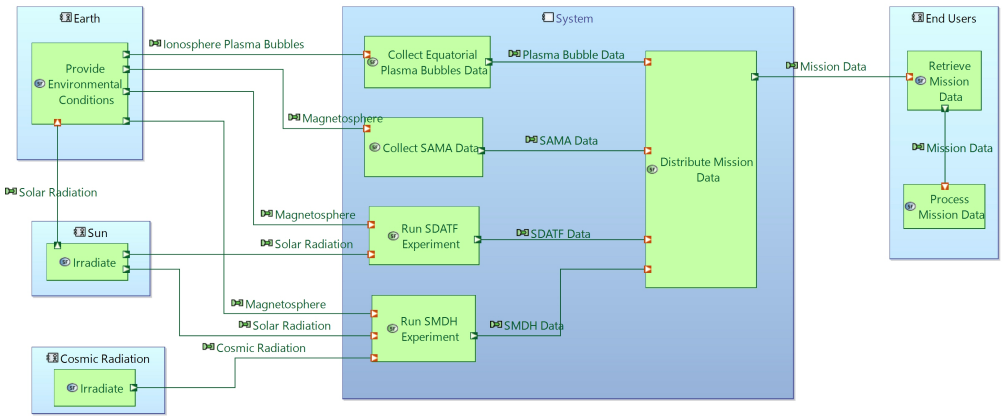
Source: Author.

5.2.3 Logical Architecture

In the third step, we describe the logical functions the system will perform using the Logical Data Flow Blank (LDFB) shown in Figure 5.6. It is important to show which functions will be executed periodically and which will be through regions of interest. This is done by connecting them to the "Reach Operation Period" or "Reach Region of Interest" functions.

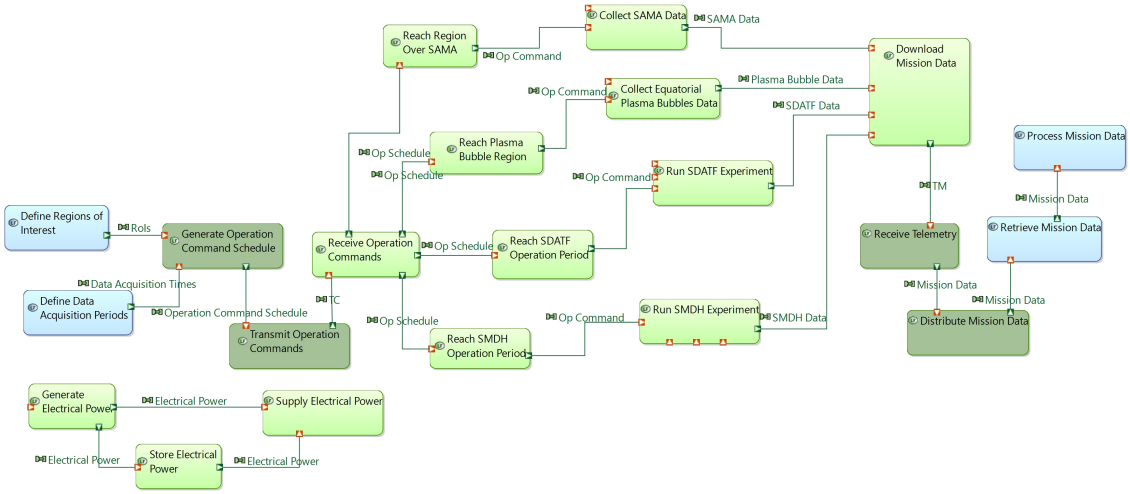
The functions are then allocated into to space and ground segments, using the Logical Architecture Blank (LAB), shown in Figure 5.7.

Figure 5.5 - Instantiated Model System Architecture Diagram.



Source: Author.

Figure 5.6 - Instantiated Model Logical Data Flow Diagram.

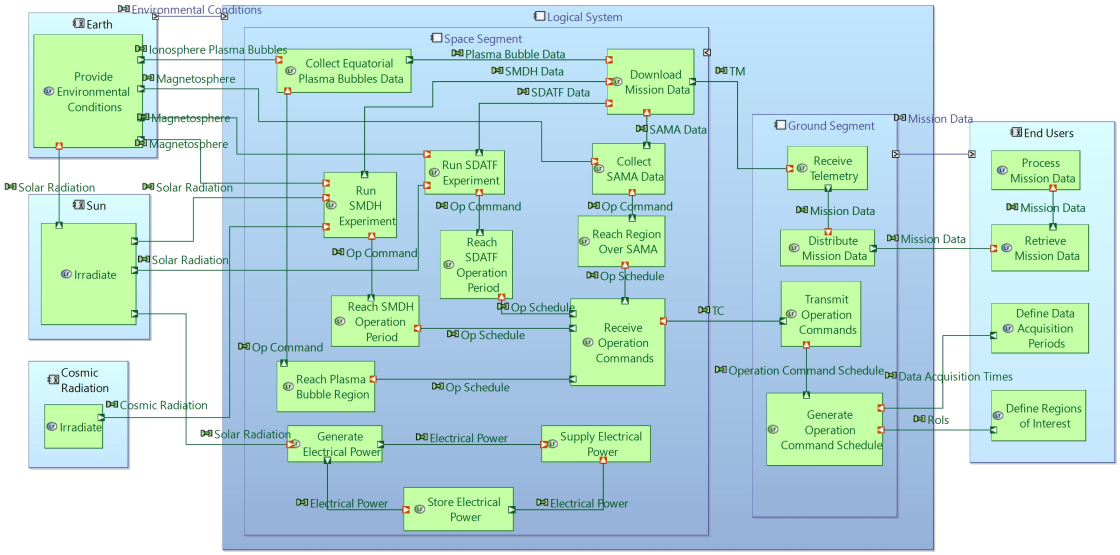


Source: Author.

5.2.4 Physical Architecture

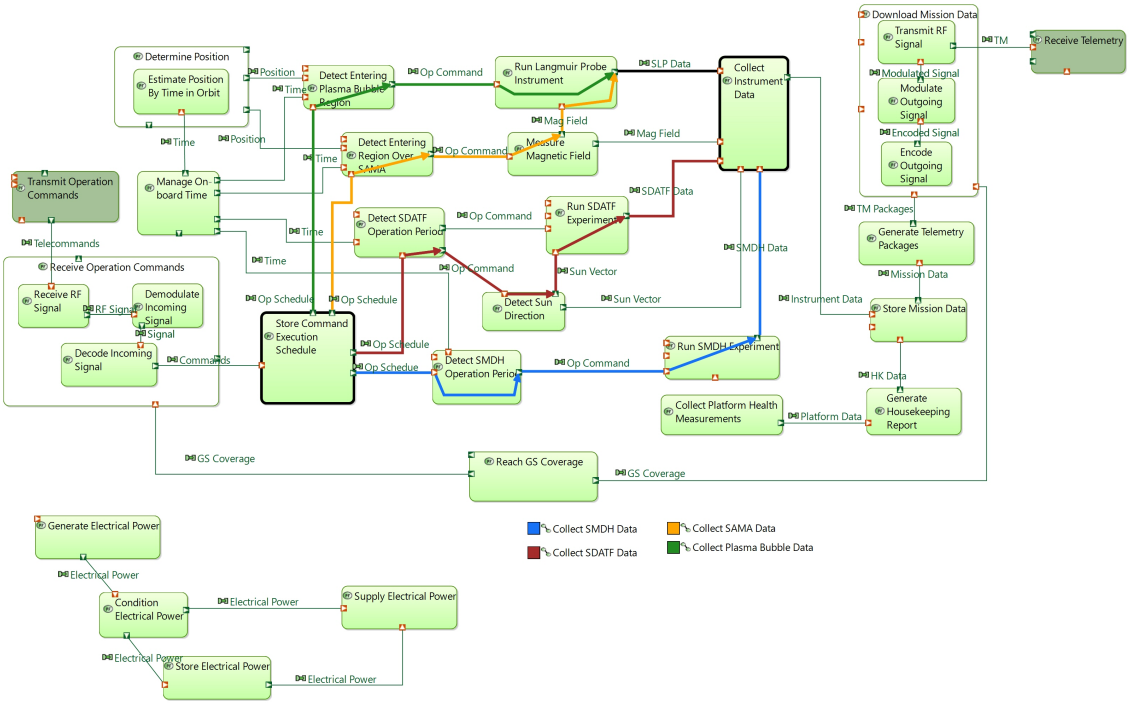
In the Physical Architecture step, the functions are broken down to the level each equipment will realize using the Physical Data Flow Blanks (PDFB), one for each segment. In the space segment, we evidence how time and position will be managed. This was especially important for this mission, since at this point it raised the awareness of having to implement a robust method of correlating the spacecraft's position with its on-board time. Since the NCBR2 does not have a GPS or any other means to detect its relative position and velocity in space, it will be necessary to predict on ground when it will enter the target regions of interest, and then generate a command list based on time, and have a reliable time keeping method on-board.

Figure 5.7 - Instantiated Model Logical Architecture Diagram.



Source: Author.

Figure 5.8 - Instantiated Model Space Segment Physical Data Flow Diagram.



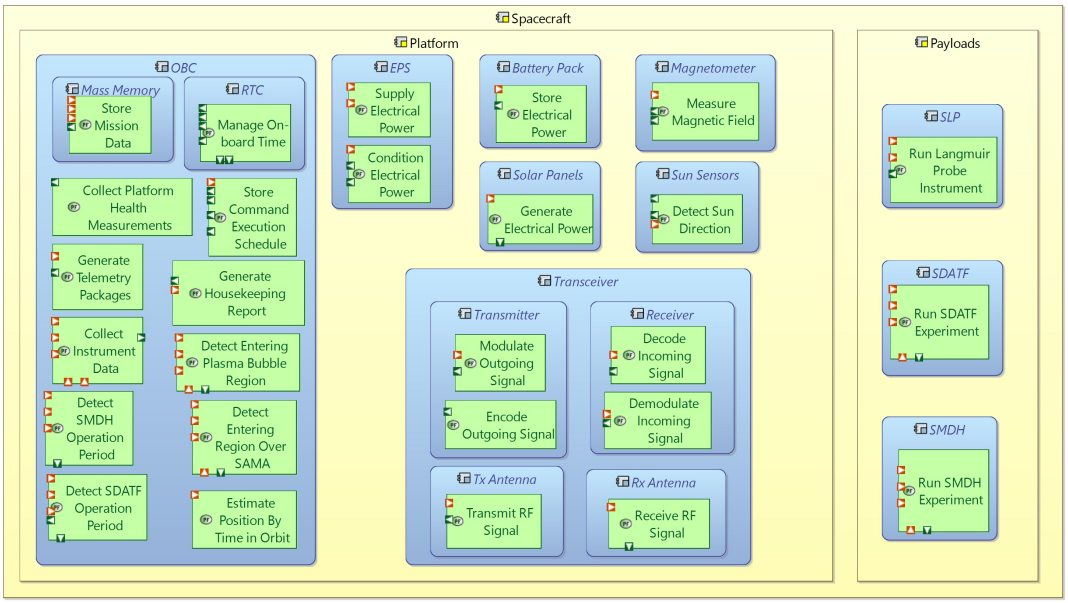
Source: Author.

Since ForPlan does not yet have the attitude control module implemented, we have suppressed the attitude control functions for this example. By using Functional Chains, a Capella functionality, the flow of information to collect each type of mission

data can be made clearer, as can be seen in the Space Segment PDFB in Figure 5.8. It can be seen that two types of mission data are collected by the same payload, by running in different conditions.

The functions are then allocated to the created components for each piece of equipment, using the Physical Architecture Blanks (PAB). The end result for the physical architecture can be seen in Figure 5.9. We have simplified the internal operation of each payload as a single black-box function, since at this point we are not concerned with their complex individual operations. Representing the payload operation this way is enough to determine when they will operate, their interfaces with other equipment, inputs and outputs.

Figure 5.9 - Instantiated Model Space Segment Physical Architecture Diagram.



Source: Author.

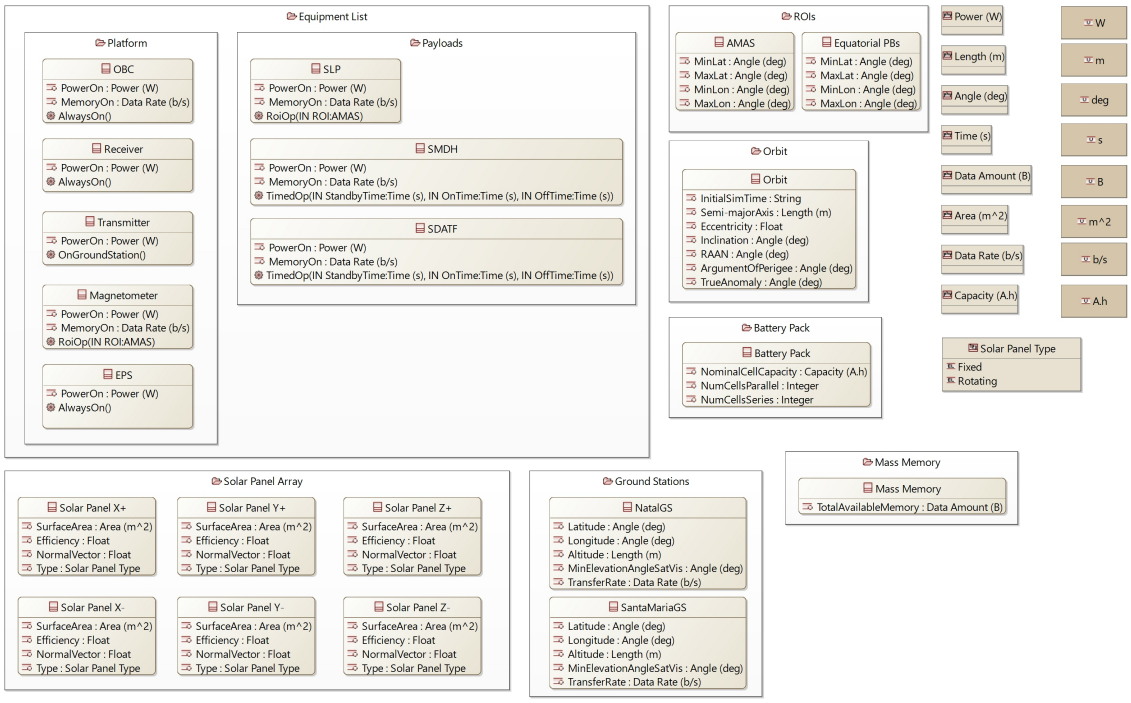
For this mission, the ground segment functions and components are considered exactly the same as the generic example model, seen in Figure 3.13.

5.2.4.1 Generating the simulation script

Finally, we proceed to the Class Diagram Blank (CDB) to generate the simulation configuration file. All of the classes are instantiated into the data packages and the result can be seen in Figure 5.10. The equipment has been defined with only a few properties, since the ones omitted have the same values as the default values for

those properties. The On-board Computer (OBC), Receiver and Electrical Power Supply (EPS) will always remain turned on, so we instantiate their *Class Operation* as *AlwaysOn()*. The transmitter will work above both of the ground stations, so it's function is *OnGroundStation()*. For the simulation configuration shown, we want to simulate the operation of the Magnetometer and SLP only when over the Southern Atlantic Magnetic Anomaly (SAMA / AMAS in portuguese), therefore, their operation function is *RoiOp()* with an input parameter that points to the AMAS ROI class. The SMDH and SDATF have periodic operation through the *TimedOp()* function with on- and off-time parameters, along with a *StandByTime* parameter which is a way to declare initial off-time.

Figure 5.10 - Instantiated Model Simulation Configuration Class Diagram Diagram.



Source: Author.

The resulting ForPlan configuration script generated running C2F with this configuration is attached as Appendix A.

5.3 Simulation trade studies

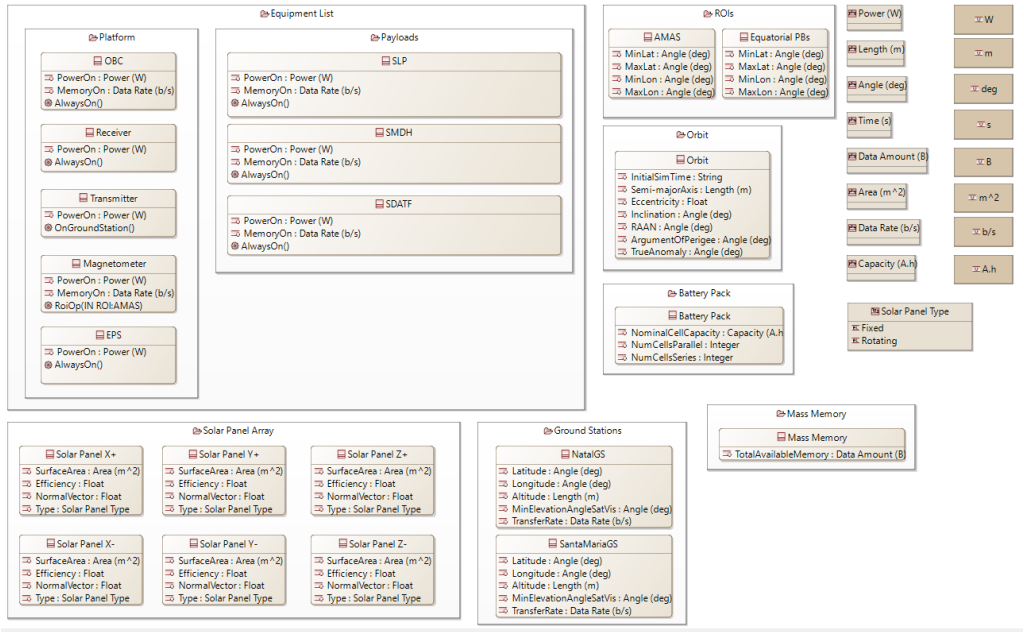
During the period this dissertation was developed, NCBR2 had already passed the early design phase, and is a direct example to what was mentioned in the beginning of the problem formulation: the mission stakeholders prioritized the usage of the

available internal volume without analyzing the mission conops, leading to power and data budget issues. This section presents a trade-study simulating operation scenarios using parameters that were still flexible at the time, with a later discussion on how the impact could be different if the study was carried out earlier.

For this analysis, the author used the resulting model created through Conops2M along with C2F to generate different operation scenarios comparing different payload operation periods and regions so that the simulation can assist in defining scenarios where the power and data budgets can be closed out with different ROIs and operation periods for the payloads.

In the first scenario, we take a first look at how the system operation would turn out with all payloads constantly operating at maximum rates. For that we set their operation function as *"AlwaysOn()"*, and the configuration diagram is set as shown in Figure 5.11, which is then converted to the Julia script and executed by the simulator.

Figure 5.11 - First Scenario Configuration Class Diagram.

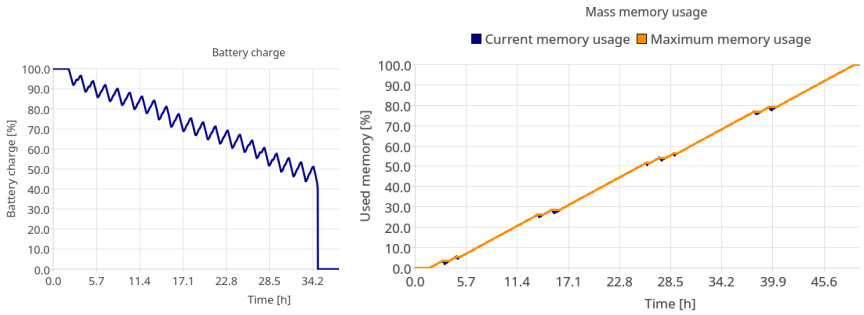


Source: Author.

After running the simulation it can be see in the output Figure 5.12 that the batteries do not charge as much as they are being demanded, and after 34 hours they fall below 40% of charge, for which the simulator then considers as depleted. In this

scenario, data collection from all payloads would be maximized, but would not be able to be downloaded, since it is much higher than the downlink capability. Figure 5.12 shows the mass memory usage for this scenario where it can be seen that the memory usage hardly ever lowers until it reaches the on-board capacity after 45 hours.

Figure 5.12 - First Scenario Simulation Results.



(a) *Battery Charge* (b) *Memory Usage*

Source: Author.

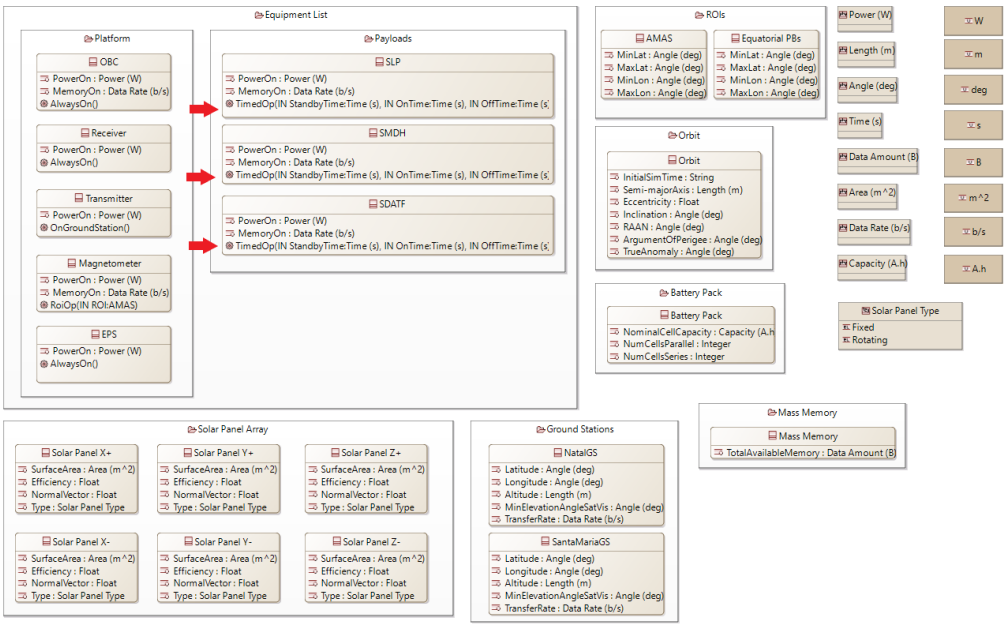
For the second scenario, we operate the payloads separately and one every orbit, alternating evenly and sequentially through them. We configure the payload classes with *TimedOp()* functions in the class diagram and generate another Julia script to be executed, as seen in Figure 5.13.

This time, the power balance is met, as can be seen in Figure 5.14. However, the amount of data generated still exceeds what can be downloaded, as can be seen in the rising mass memory usage in Figure 5.14.

After a couple of iterations balancing operation time prioritization and the data generation rate of each payload, it is possible to reach a scenario by operating the SLP over its regions of interest and adjusting the other two payloads' data generation rate, compromising the total amount of data generated by the three payloads, but managing to meet the power and data budgets, as can be seen in Figures 5.15 A and B.

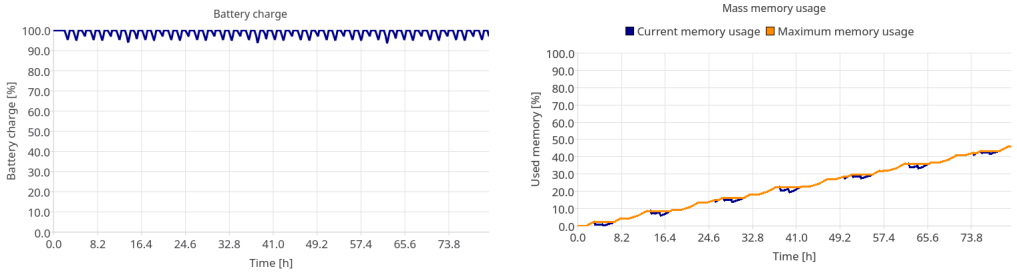
Table 5.2 shows a summary of the three scenarios, comparing the different operation functions for each of the three payloads. Table 5.3 shows the main pros and cons for each scenario.

Figure 5.13 - Second Scenario Configuration Class Diagram.



Source: Author.

Figure 5.14 - Second Scenario Simulation Results



(a) *Battery Charge* (b) *Memory Usage*

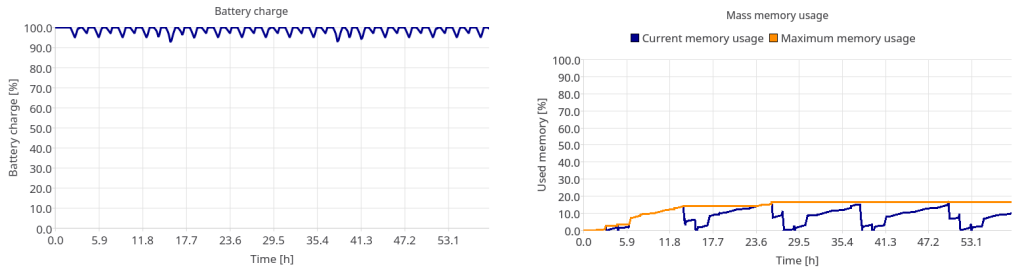
Source: Author.

Table 5.2 - Trade Scenarios Operation Functions Summary.

Scenario	SLP	SDATF	SMDH
1	AlwaysOn()	AlwaysOn()	AlwaysOn()
2	TimedOp(1on/2off)	TimedOp(1on/2off)	TimedOp(1on/2off)
3	RoiOP(AMAS)	TimedOp(1on/1off)	TimedOp(1on/1off)

Reducing and adjusting operation periods of the payloads result in less data generated compared to initially though, which leads to a longer mission operation period in order to obtain the data initially planned. The mission objectives will still be met,

Figure 5.15 - Third Scenario Simulation Results



(a) *Battery Charge* (b) *Memory Usage*

Source: Author.

Table 5.3 - Trade Scenarios Main Pros and Cons.

Scenario	Pros	Cons
1	Maximum operation time	Too high power consumption and data generation
2	Evenly balanced operation time	Too high data generation
3	Power and data budgets met	Low operation time

but it will take longer in order to do so, depending on a longer mission life.

If the definition of the concept of operations along with operation scenario simulation were performed in the early design phase, the mission could use the modelling and simulation to analyze, for example:

- the use of different communications equipment with higher bandwidth.
- limitations for payloads and request hardware/software alterations.
- the adoption of a different form factor with larger solar panels for higher power generation.
- the impact of using other partner ground stations.
- the impact of having a larger battery pack.

Trade studies considering these options could provide the information for the mission directors to opt for alternatives in the mission architecture that would have better impacts on the mission objectives.

6 CONCLUSION

In this dissertation a process to guide the modelling of the concept of operations of space missions was introduced. Called Conops2M, it was developed in the context of a Model-Based Systems Engineering methodology, to assist in early stage design studies. A plugin (C2F) developed to automatically transform the model generated by Conops2M, to an entry for a simulation tool (ForPlan), was also presented. Finally, the usability of the modelling process was exemplified by its application on the design and verification, by simulation, of operation scenarios of a CubeSat mission.

Using the Conops2M modelling process, it was possible to model a primary version of the concept of operations of the NanosatC-BR2 mission, and the simulation trade studies were of high importance in defining the requirements for on-board and ground-station software to prepare the operation of the system. Luckily the operation of the payloads were adjusted without major implications in the mission, only an extended operation period. The trade studies provided insight into the limitations regarding dedicated operation time and amount of data to be obtained, and provided data for system-level decision making. Future missions with even more flexibility on mission design parameters and equipment selection can benefit from this process.

With the evolution of the NanosatC-BR2 mission, hardware limitations and software implementation constraints were found which lead to differences in power consumption, data generation and download volume. The resulting conops model was then once again used to simulate the new operation scenarios to assist in the operations planning and final on-board software development.

Conops2M provides a different approach from the behaviour modelling of the CRM and other aforementioned system models by providing the user with a methodological guidance in decomposing the initial stakeholder and mission objectives into functionalities that are tied to low-level equipment / subsystems, depending on how far the user would like to decompose the functions. It is in a straight-forward and intuitive way, facilitating the construction of an initial architecture for a *"simple"* CubeSat mission and its concept of operations, and preparing it for simulation using the ForPlan simulator, without requiring the user to understand ForPlan's source code or internal operation.

Using Conops2M for an initial concept system model highly reduces the barriers to

entry found in the CRM and other system models that all used commercial software that require paid licenses, by using open-source tools such as Capella and soon-to-be ForPlan. Capella's modelling language is also simpler to use than SysML.

It is worth mentioning that generating the different trade study simulations using the visual model was simpler than editing lines of code and manually altering functions in the configuration script (that's what automatic code generation is for!)

6.1 Future work

For future work, the author envisions:

- The use of Modes & States Diagrams along with Exchange Scenarios inside Capella to describe specific operation functions of equipment, and then implement the automatic transformation in C2F to write the custom functions into ForPlan.
- Transition from the use of hard-coded names in the Class Diagram Blanks to pre-defined types in higher meta-levels. This would allow the users to create the objects in a more direct and fail-proof manner. This would require a change inside Capella.
- Implement Conops2M using other modelling tools and language, such as ModelCenter using SysML for example, and compare the differences in the resulting models and available resources.
- Adapt C2F to generate inputs for other mission design software, such as AGI's Systems Tool Kit (STK) or NASA's General Mission Analysis Tool (GMAT).
- Extend "Region of Interest" operation in ForPlan with regions different than square boxes.

REFERENCES

AERONAUTICS, N.; NASA, S. A. **MarCO (Mars Cube One)**. 2019. Available from:
<https://solarsystem.nasa.gov/missions/mars-cube-one/in-depth/>. 13

AKHUNDOV, J.; WERNER, M.; SCHAUS, V.; GERNDT, A. Using timed automata to check space mission feasibility in the early design phases. In: IEEE AEROSPACE CONFERENCE, 2016. **Proceedings...** [S.l.]: IEEE, 2016. p. 1–9. 4

ASUNDI, S. A.; FITZ-COY, N. G. Cubesat mission design based on a systems engineering approach. In: IEEE AEROSPACE CONFERENCE, 2013. **Proceedings...** [S.l.]: IEEE, 2013. p. 1–9. 2

BANKS, J. **Discrete event system simulation**. [S.l.]: Pearson Education, 2005. 17

BODIN, P.; NYLUND, M.; BATTELINO, M. Satsim—a real-time multi-satellite simulator for test and validation in formation flying projects. **Acta Astronautica**, v. 74, p. 29–39, 2012. 15

BURGER, E. E. **A conceptual MBSE framework for satellite AIT planning**. Thesis (PhD in Space Engineering and Technology - Space Systems Engineering and Management) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos. 7

CERQUEIRA, C. S. **Tangible collaboration applied into space systems concurrent engineering concept studies**. Thesis (PhD in Space Engineering and Technology - Space Systems Engineering and Management) — Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, 2018. 7, 25

CHAGAS, R. A.; SOUSA, F. L. de; LOURO, A. C.; SANTOS, W. G. dos. Modeling and design of a multidisciplinary simulator of the concept of operations for space mission pre-phase a studies. **Concurrent Engineering**, p. 1063293X18804006, 2018. 4, 18, 19, 30, 47

CHAGAS, R. A. J.; LOURO, A. C.; SOUSA, F. L. de; SANTOS, W. G. dos. Satellite simulator for verification of mission operational concepts in pre-phase a studies. In: INTERNATIONAL CONFERENCE ON SYSTEMS & CONCURRENT ENGINEERING FOR SPACE APPLICATIONS, 7., 2016. **Proceedings...** [S.l.], 2016. 18

DAVID, L. **Cubesats: tiny spacecraft, huge payoffs**. 2004. Available from: <https://www.space.com/308-cubesats-tiny-spacecraft-huge-payoffs.html>. Access in: 15/03/2021. 13

EICKHOFF, J. **Simulating spacecraft systems**. [S.l.]: Springer Science, 2009. 2, 3, 15, 16

EICKHOFF, J.; HENDRICKS, R. The significant role of simulation in satellite development and verification. **Aerospace Science and Technology**, v. 9, n. 3, p. 273–283, 2005. 2, 15, 16, 17

ESA, E. S. A. **What is concurrent engineering**. May 2018. Available from: <https://www.esa.int/Our_Activities/Space_Engineering_Technology/CDF/What_is_concurrent_engineering>. 26

ESTABLE, S.; GRANGER, T.; LOCHOW, T.; ZOEBELEIN, T.; BRAUER, N.; TOLCHINSKY, I.; GENERÉ, S.; KONING, H. de. Systems modelling and simulation of the esa e.deorbit space debris removal mission. 2017. Available from: <https://www.phoenix-int.com/tech-papers/systems-modelling-simulation-esa-e-deorbit-space-debris-removal-mission/>. 3, 21, 30

ESTEFAN, J. A. et al. Survey of Model-Based Systems Engineering (MBSE) methodologies. **Incose MBSE Focus Group**, v. 25, n. 8, p. 1–12, 2007. 6, 7

EUROPEAN COOPERATION FOR SPACE STANDARDIZATION - ECSS. **ECSS-E-ST-10C**: Space engineering: ground systems and operations. [S.l.]. 29

_____. **ECSS-E-TM-10-21A**: space engineering: system modelling and simulation. [S.l.]. 2, 15, 16, 28

_____. **ECSS-E-TM-10-25a**: space engineering: engineering design model data exchange (cdf). Netherlands. 3

FISCHER, P.; LÜDTKE, D.; LANGE, C.; ROSHANI, F.-C.; DANNEMANN, F.; GERNDT, A. Implementing model-based system engineering for the whole lifecycle of a spacecraft. **CEAS Space Journal**, v. 9, n. 3, p. 351–365, 2017. 2, 20

FISCHER, P. M.; LÜDTKE, D.; SCHAUS, V.; GERNDT, A. A formal method for early spacecraft design verification. In: IEEE AEROSPACE CONFERENCE, 2013. **Proceedings...** [S.l.]: IEEE, 2013. p. 1–8. 3

GUO, J.; GILL, E.; FIGARI, S. Model-based systems engineering to support the development of nano-satellites. In: INTERNATIONAL ASTRONAUTICAL CONGRESS, 2014. **Proceedings...** [S.l.]: IAC, 2014. v. 10, p. 6991–7003. 20

HEIDT, H.; PUIG-SUARI, J.; MOORE, A.; NAKASUKA, S.; TWIGGS, R. Cubesat: a new generation of picosatellite for education and industry low-cost space experimentation. In: ANNUAL USU CONFERENCE ON SMALL SATELLITES. **Proceedings...** [S.l.], 2000. 13

HIHN, J.; KARPATI, G.; CHATTOPADHYAY, D.; MCGUIRE, M.; BORDEN, C.; PANEK, J.; WARFIELD, K. Aerospace concurrent engineering design teams: current state, next steps and a vision for the future. In: AIAA SPACE 2011 CONFERENCE & EXPOSITION. **Proceedings...** [S.l.], 2011. 26

INCOSE. Vision 2020 (incose-tp-2004-004-02). 2007. 20

_____. **INCOSE MBSE standards**. 2019. Available from: <https://www.omgwiki.org/MBSE/doku.php?id=mbse:standards>. 6

INCOSE, I. C. O. S. E. **MBSE initiative**. Available from: <http://www.omgwiki.org/MBSE/doku.php>. 20

IWATA, C.; INFELD, S.; BRACKEN, J. M.; MCGUIRE, M.; MCQUIRCK, C.; KISDI, A.; MURPHY, J.; COLE, B.; ZARIFIAN, P. Model-based systems engineering in concurrent engineering centers. In: AIAA SPACE CONFERENCE AND EXPOSITION, 2015. **Proceedings...** [S.l.], 2015. 3, 4, 26, 27

JULIA Homepage. June 2020. Available from: <https://julialang.org/>. 19, 35

KAPURCH, S. J. **NASA systems engineering handbook**. [S.l.]: Diane Publishing, 2010. 25

KASLOW, D. Cubesat model based system engineering (mbse) reference model-application in the concept lifecycle phase. In: AIAA SPACE 2015 CONFERENCE AND EXPOSITION. **Proceedings...** [S.l.], 2015. 22, 23

KASLOW, D.; AYRES, B.; CAHILL, P. T.; HART, L.; YNTEMA, R. A model-based systems engineering (mbse) approach for defining the behaviors of cubesats. In: IEEE AEROSPACE CONFERENCE, 2017. **Proceedings...** [S.l.], 2017. 22, 24

KASLOW, D.; AYRES, B.; CAHILL, P. T.; HART, L.; LEVI, A. G.; CRONEY, C. Developing an mbse cubesat reference model–interim status# 4. In: AIAA SPACE AND ASTRONAUTICS FORUM AND EXPOSITION, 2018. **Proceedings...** [S.l.], 2018. 4, 22

KASLOW, D.; AYRES, B.; CHONOLES, M.; GASSTER, S.; HART, L.; MASSA, C.; YNTEMA, R. Cubesat model-based systems engineering (mbse) reference model–model distribution and application–interim status# 2. In: AIAA SPACE AND ASTRONAUTICS FORUM AND EXPOSITION, 2016. **Proceedings...** [S.l.], 2016. 23

KASLOW, D.; SOREMEKUN, G.; KIM, H.; SPANGELO, S. Integrated model-based systems engineering (mbse) applied to the simulation of a cubesat mission. In: IEEE AEROSPACE CONFERENCE, 2014. **Proceedings...** [S.l.], 2014. 21, 22, 23, 30

KRANZ, S.; GUTIERREZ, B. G.; MATTHYSSEN, A.; FIJNEMAN, M. System concept simulation for concurrent engineering. In: WORKSHOP ON SIMULATION FOR EUROPEAN SPACE PROGRAMMES, 2015. **Proceedings...** [S.l.], 2015. 3, 16, 28

KULU, E. **Cubesat database**. 2019. Available from: <https://www.nanosats.eu>. 2, 13

LANGE, C.; GRUNDMANN, J. T.; KRETZENBACHER, M.; FISCHER, P. M. Systematic reuse and platforming: application examples for enhancing reuse with model-based systems engineering methods in space systems development. **Concurrent Engineering**, v. 26, n. 1, p. 77–92, 2018. 6, 21

LARANJEIRO, N.; GOMEZ, C.; SCHIAVONE, E.; MONTECCHI, L.; CARVALHO, M. J.; LOLLINI, P.; MICSKEI, Z. Addressing verification and validation challenges in future cyber-physical systems. In: LATIN AMERICAN SYMPOSIUM ON DEPENDABLE COMPUTING. **Proceedings...** [S.l.], 2019. 35

LOWE, C.; MACDONALD, M. Rapid model-based inter-disciplinary design of a cubesat mission. **Acta Astronautica**, v. 105, n. 1, p. 321–332, 2014. 15

MATTIELLO-FRANCISCO, F. **Addressing verification and validation challenges in future cyber-physical systems (ADVANCE)**. 2019. Available from: <https://www.advance-rise.eu/>. 30

MEHRPARVAR, A.; PIGNATELLI, D.; CARNAHAN, J.; MUNAKAT, R.; LAN, W.; TOORIAN, A.; HUTPUTANASIN, A.; LEE, S. Cubesat design specification rev. 13. 2014. Available from: <https://blogs.esa.int/philab/files/2019/11/RD-02_CubeSat_Design_Specification_Rev._13_The.pdf>. 13

NASA. **CubeSat 101: basic concepts and processes for first-time CubeSat developers**. [S.l.], 2017. 15

RAIF, M.; WALTER, U.; BOUWMEESTER, J. Dynamic system simulation of small satellite projects. **Acta Astronautica**, v. 67, n. 9-10, p. 1138–1156, 2010. 3, 16

ROQUES, P. Mbse with the arcadia method and the capella tool. In: 8TH EUROPEAN CONGRESS ON EMBEDDED REAL TIME SOFTWARE AND SYSTEMS, 2016. **Proceedings...** [S.l.], 2016. 8, 10, 33

_____. **Systems architecture modeling with the Arcadia method: a practical guide to Capella**. [S.l.]: ISTE Press - Elsevier, 2017. 7, 8, 10, 34, 36

ROTHENBERG, J.; WIDMAN, L. E.; LOPARO, K. A.; NIELSEN, N. R. **The nature of modeling**. [S.l.: s.n.], 1989. 16

SCHAUS, V.; FISCHER, P.; LÜDTKE, D.; BRAUKHANE, A.; ROMBERG, O.; GERNDT, A. Concurrent engineering software development at german aerospace center-status and outlook. In: INTERNATIONAL WORKSHOP ON SYSTEM & CONCURRENT ENGINEERING FOR SPACE APPLICATIONS, 4., 2010. **Proceedings...** [S.l.], 2010. 3

SCHUCH, N. J.; DURãO, O. S. C.; SILVA, M. R. da; MATTIELLO-FRANCISCO, F.; SILVA, A. L. da. Nanosatc-br status - a joint cubesat-based program developed by inpe and ufsm. In: IAA CONFERENCE ON UNIVERSITY SATELLITE MISSIONS AND CUBESAT WORKSHOP, 2017, 4. **Proceedings...** [S.l.], 2017. 49

SCHUMANN, H.; BRAUKHANE, A.; GERNDT, A.; GRUNDMANN, J.; HEMPEL, R.; KAZEMINEJAD, B.; ROMBERG, O.; SIPPEL, M. Overview of the new concurrent engineering facility at dlr. In: WORKSHOP ON SYSTEM & CONCURRENT ENGINEERING FOR SPACE APPLICATIONS, 2008, 3., 2008. **Proceedings...** [S.l.], 2008. 26

SPANGELO, S. C.; KASLOW, D.; DELP, C.; COLE, B.; ANDERSON, L.; FOSSE, E.; GILBERT, B. S.; HARTMAN, L.; KAHN, T.; CUTLER, J. Applying

model based systems engineering (mbse) to a standard cubesat. In: IEEE AEROSPACE CONFERENCE, 2012. **Proceedings...** [S.l.], 2012. 4, 6, 20, 21, 24

SPANGELO, S. C. et al. Model based systems engineering (mbse) applied to radio aurora explorer (rax) cubesat mission operational scenarios. In: IEEE AEROSPACE CONFERENCE, 2013. **Proceedings...** [S.l.], 2013. 4, 21

SWARTWOUT, M.; JAYNE, C. University-class spacecraft by the numbers: Success, failure, debris.(but mostly success.). In: SMALL SATELLITE CONFERENCE, 2016, 2016. **Proceedings...** [S.l.], 2016. 2, 14, 21

VENTURINI, C. C.; BRAUN, B.; HINKLEY, D.; BERG, G. Improving mission success of cubesats. **El Segundo: Aerospace Corporation**, 2017. 2, 14

WANG, K.; ZHANG, B.; XING, T. Preliminary integrated analysis for modeling and optimizing space stations at conceptual level. **Aerospace Science and Technology**, v. 71, p. 420–431, 2017. 16

WASEEM, M.; SADIQ, M. U. Application of model-based systems engineering in small satellite conceptual design-a sysml approach. **IEEE Aerospace and Electronic Systems Magazine**, v. 33, n. 4, p. 24–34, 2018. 2, 6, 20

WERTZ, J. R.; EVERETT, D. F.; PUSCHELL, J. J. **Space mission engineering: the new SMAD**. [S.l.]: Microcosm Press, 2011. 3, 25

APPENDIX A - Example C2F Output ForPlan Configuration Script

```
using SatelliteToolbox
using Pkg

Pkg.activate("../")

using ForplanSimulatorCore

const deg2rad = pi / 180
const Kib = 1024
const Mib = 1024 * 1024
const Gib = 1024 * 1024 * 1024


function inside_ROIs(ROI_list, lat, lon)
for ROI in ROI_list
minLat = ROI[1] * deg2rad
maxLat = ROI[2] * deg2rad
minLon = ROI[3] * deg2rad
maxLon = ROI[4] * deg2rad
if (lat > minLat) && (lat < maxLat) && (lon > minLon) && (lon < maxLon)
return true
end
end
return false
end

function TimedOp(equip, sim_workspace)
params  = equip.params
deltaT   = sim_workspace.deltaT
#conditions = params[8]
eclipse = sim_workspace.eclipse
OnTime = params[6]
OffTime = params[7]

# Wait for the initial stand-by time.
```

```
if params[1] > 0.0
params[1] -= deltaT
return false, 0.0, 0.0
end

if params[2] > 0.0 #On time
params[2] -= deltaT
if params[2] <= 0.0
params[3] = OffTime
end
return true, params[4], params[5]
end

if params[3] > 0.0 #off time
params[3] -= deltaT
if params[3] <= 0.0
params[2] = OnTime
end
return false, 0.0, 0.0
end

#default
return false, 0.0, 0.0
end

function RoiOp(equip, sim_workspace)
params = equip.params
deltaT = sim_workspace.deltaT
ROI_list = params[4]
lat = sim_workspace.lat
lon = sim_workspace.lon
eclipse = sim_workspace.eclipse

# Wait for the initial stand-by time.
if params[1] > 0.0
params[1] -= deltaT
return false, 0.0, 0.0
```

```
end

#if (check_conditions(conditions, eclipse))
if inside_ROIs(ROI_list, lat, lon)
return true, params[2], params[3]
end
#end

#default
return false, 0.0, 0.0
end

# 1. Initial orbital parameters
# ================================================================================

orb = Orbit(
DatetoJD(2020, 01, 01, 12, 0, 0),# Initial simulation time.
6974408.0,# Semi-major axis [m].
0,# Eccentricity.
97.910 * deg2rad,# Inclination [rad].
280.932 * deg2rad,# RAAN [rad].
0.0 * deg2rad,# Argument of perigee [rad].
0.0 * deg2rad) # True anomaly [rad].

torb = period(orb)

# 2. Equipment list
# ================================================================================


equip_1= ForplanSimulatorCore.Equipment{Float64}(
name = "OBC",
f! = ForplanSimulatorCore.equip_always_on!,
params = [torb, 0.0, 0.383, 30.0])
equip_2= ForplanSimulatorCore.Equipment{Float64}(
name = "Receiver",
f! = ForplanSimulatorCore.equip_always_on!,
```

```
params = [torb, 0.0, 0.193, 0.0])
equip_3= ForplanSimulatorCore.Equipment{Float64}(
name = "Transmitter",
f! = ForplanSimulatorCore.equip_on_ground_station!,
params = [torb, 0.0, 0.0, 1.078, 0.0, 0.0, 0])
equip_4= ForplanSimulatorCore.Equipment{Float64}(
name = "Magnetometer",
f! = RoiOp,
params = [torb, 0.016, 96.0,
[[-60.0, 0.0, -90.0, -20.0]]])
equip_5= ForplanSimulatorCore.Equipment{Float64}(
name = "EPS",
f! = ForplanSimulatorCore.equip_always_on!,
params = [torb, 0.0, 0.250, 0.0])



payload_1= ForplanSimulatorCore.Equipment{Float64}(
name = "SLP",
f! = RoiOp,
params = [torb, 0.873, 800.0,
[[-60.0, 0.0, -90.0, -20.0]]])
payload_2= ForplanSimulatorCore.Equipment{Float64}(
name = "SMDH",
f! = TimedOp,
params = [torb, torb, torb, 1.00, 5.0, torb, torb])
payload_3= ForplanSimulatorCore.Equipment{Float64}(
name = "SDATF",
f! = TimedOp,
params = [2*torb, 1*torb, 1*torb, 0.264, 512, 1*torb,*torb])



equipment_list = [
equip_1
equip_2
equip_3
```

```
equip_4
equip_5
payload_1
payload_2
payload_3
]
equipment_group = [
create_equipment_group("Platform", 1:1:5)
create_equipment_group("Payloads", 6:1:8)
]

# 3. Solar panels
# ========================================================================

sag = [
# Solar Panel X+
create_static_solar_panel(0.01804,#Solar panel area [m].
0.161,# Solar panel efficiency.
[1.0, 0.0, 0.0],# Normal Vector of the solar panel.
[0.0],# Transient efficiency.
[0.0]),# Transient times [s].
# Solar Panel Y+
create_static_solar_panel(0.01804,#Solar panel area [m].
0.161,# Solar panel efficiency.
[0.0, 1.0, 0.0],# Normal Vector of the solar panel.
[0.0],# Transient efficiency.
[0.0]),# Transient times [s].
# Solar Panel Z+
create_static_solar_panel(0.01804,#Solar panel area [m].
0.161,# Solar panel efficiency.
[0.0, 0.0, 1.0],# Normal Vector of the solar panel.
[0.0],# Transient efficiency.
[0.0]),# Transient times [s].
# Solar Panel X-
create_static_solar_panel(0.01804,#Solar panel area [m].
0.161,# Solar panel efficiency.
[-1.0, 0.0, 0.0],# Normal Vector of the solar panel.
```

```
[0.0],# Transient efficiency.
[0.0]),# Transient times [s].
# Solar Panel Y-
create_static_solar_panel(0.01804,#Solar panel area [m].
0.161,# Solar panel efficiency.
[0.0, -1.0, 0.0],# Normal Vector of the solar panel.
[0.0],# Transient efficiency.
[0.0]),# Transient times [s].
# Solar Panel Z-
create_static_solar_panel(0.01804,#Solar panel area [m].
0.161,# Solar panel efficiency.
[0.0, 0.0, -1.0],# Normal Vector of the solar panel.
[0.0],# Transient efficiency.
[0.0]),# Transient times [s].
]
# 4. Batteries
# ================================================================================

battery = create_battery_pack(
2,# Num. of cells in series.
1,# Num. of cells in parallel.
1.0,# Charging efficiency.
1.05,# Discharge efficiency.
[1.0,0.8,0.5],# Degradation factors.
[50*torb,300*torb]) # Degradation periods.

# 5. Ground stations
# ================================================================================

NatalGS = GroundStation{Float64}(
"NatalGS",
-0.2644 * deg2rad,
-37.0403 * deg2rad,
0,
5 * deg2rad,
4800,
[(0.0, 24.0)]
```

```
)
SantaMariaGS = GroundStation{Float64}(
"SantaMariaGS",
-29.7124 * deg2rad,
-53.7174 * deg2rad,
113,
5 * deg2rad,
4800,
[(0.0, 24.0)]
)
ground_stations = [NatalGS, SantaMariaGS]
# 6. Mass memory
# ================================================================================

total_memory = 2000000000

# 7. Simulation configuration
# ================================================================================

conf = SimulationConfiguration(tf = 60*60*24.0*8,
orb = orb,
prop = :J4,
deltaT = 20.0,
standalone = false,
ground_stations = ground_stations,
equipment_list = equipment_list,
equipment_groups = equipment_group,
sag = sag,
battery = battery,
total_memory = total_memory)

run_simulation(conf)
```