Model-Based CubeSat Flight-Software Architecture using a Docs-as-Code approach

Sharan Asundi <sasundi@odu.edu>, Sean Marquez <sean@space.coop>, Kevin Chiu <kchiu002@odu.edu>, Gus Williams <awill098@odu.edu>, Jimesh D. Bhagatji <jbhag001@odu.edu>, Patrick Clark <pclar001@odu.edu>

Introduction

Model-Based Systems Engineering (MBSE) is an approach to systems engineering whereby models serve as the *authoritative source of truth* for conducting systems engineering activities, such as the design, specification, analysis, verification, and validation of a system [1].

The NASA handbook on systems engineering can be applied to CubeSat Mission design, in effort to facilitate a top-down design methodology from mission concept to specification of subsystem components, including flight software architecture [2].

The SeaLion Mission, a joint CubeSat mission between the Old Dominion University (ODU) and Coast Guard Academy (CGA), adopted both a model-based and docs-as-code approach to developing a flight software architecture as a means to minimize configuration management overhead. While also providing full traceability of requirements and component specifications of the flight software, such as stakeholder needs, user stories, and data structures pertaining to the CubeSat mission, which are captured are within a model that yields deduplication of content both human-readable and machine-queryable.

This article presents the modeling language, tools, and technical approach used to fascilitate the configuration management, design, specification, and implementation of the SeaLion mission architecture for the flight software.

Background

Information in a traditional systems engineering approach today is mostly captured informally, not authored based on a methodology, configuration managed in silo tools, adhocly and infrequently integrated, not easily traceable to its provenance, not properly configuration managed, not properly changed managed, and not effectively shared with stakeholders [3].

In contrast, a model-based systems engineering approach supports capturing information in a highly structured modeling language, authored based on a methodology, configuration managed in a common tool, highly integrated, traceable to its provenance, and sharing with stakeholders.

Goals

The goal of the SeaLion CubeSat flight software architecture was to capture the data structures and expected behaviors of the flight software, such that it can be unambiguiously understood enough to be implemented, as well as provide full traceability and rationale for architectural elements with minimal configuration management overhead [4].

The MBSE approach was of interest to the SeaLion CubeSat flight software team, as to yield the benefits of reducing ambiguity that usually comes with using informal language for specifying aspects of a system, as well as minimizing duplication of content that tends to accumulate in a document-based systems engineering approach.

Furthermore, it was also of interest to adopt a docs-as-code approach, as to yield the benefits of utilizing the same tools used to manage code, using version control tools (e.g., Git), for the configuration management of flight software architecture documentation, captured in a model-based approach [5].

Selection of a modeling language, modeling tool, and technical approach needs to be taken into consideration, in order to properly adopt a MBSE approach.

Other considerations include overhead incurred with training the team, as well as technical overhead with setting up and maintaining the modeling tools.

Modeling Language, Tool, & Methodology

The Mach30 modeling language (m30ml) is a YAML-based modeling language for defining a software architecture. Since YAML is a lightweight, highly-structured, human-readable, machine queryable, and line-oriented markup language, it was ideal for document generation use cases, as well as use with version control tools like Git.

Mach30 also had minimal technical overhead as it was compatible with modern doc tools such as asciidoctor & bibtex.

Mach30 also provided modeling elements familiar in agile software development, such as stakeholder needs, user stories, and data structures, with relationship elements for defining traceability between modeling elements.

For the aforementioned reasons, the SeaLion CubeSat flight software team downselected m30ml for specifying the SeaLion CubeSat flight software architecture [6].

Stakeholder Needs

The SeaLion project's methodology documentation uses m30ml based on YAML architecture modeling tools. The first step to build the architecture is to define the stakeholder needs. The two stakeholders for Sealion are ODU and CGA with their respective needs categorized on priority from primary to secondary to tertiary. These stakeholder needs are listed in the following Table 1.

Table 1. Stakeholder Needs

ID / Name	Statement
1.1: Primary Mission Objective A1	The SeaLion mission shall establish UHF communication link with Virginia ground station
1.2: Primary Mission Objective A2	The SeaLion mission shall establish S-Band communication link with MC3 ground station
1.3: Primary Mission Objective A3	The SeaLion mission shall successfully transmit “mission data” defined above to ground stations on the Earth.
1.4: Primary Mission Objective A4	The SeaLion mission shall adhere to CubeSat standards.[1]
1.5: Primary Mission Objective A5	The SeaLion mission shall validate the operation of the Impedance Probe (IP) as a primary payload in-orbit.
2.1: Secondary Mission Objective B1	The SeaLion mission shall provide a means to validate a Multi-spectral Sensor (Ms-S) in-orbit
2.2: Secondary Mission Objective B2	The SeaLion mission shall provide a means to validate a deployable composite structure (DeCS) in-orbit
3.1: Tertiary Mission Objective C1	The SeaLion mission shall qualify on-orbit the deployment and functioning of the newly developed UHF antenna system and its deployment.
3.2: Tertiary Mission Objective C2	The SeaLion mission shall qualify a CubeSat bus architecture for very-low Earth orbit (VLEO)
3.3: Tertiary Mission Objective C3	The SeaLion shall verify DeCS in-orbit behavior performance.

User Stories

The SeaLion Mission Architecture’s stakeholder needs are then used to identify a series of user stories which then lead to design decisions captured in data structure and activity definitions. These are created from the perspective of the ground station operator to define the tasks that need to be completed to satisfy the user stories. These user stories are listed in the following Table 2.

Table 2. User Stories

ID / Name	Statement	Derived From
1: Ping Satellite	As a **Ground Station Operator** I want to **Ping satellite** so that I can **Establish communication link with satellite**.	Primary Mission Objective A1

ID / Name	Statement	Derived From
2: View Satellite Beacon Data	As a **Ground Station Operator** I want to **view satellite beacon data (alternating between health & mission data), received via UHF** so that I can **verify that satellite is operating nominally**.	Primary Mission Objective A1 Primary Mission Objective A3 Primary Mission Objective A5 Secondary Mission Objective B1 Secondary Mission Objective B2 Tertiary Mission Objective C1 Tertiary Mission Objective C2 Tertiary Mission Objective C3
3: Send Request to Set Interrupt Timer	As a **Ground Station Operator** I want to **send a request to set count value at which interrupt timers (i.e., beacon, GPS ping, or orbit propagator) are triggered** so that I can **finetune parameters for attitude or orbit analysis or to conserve power**.	
4: Request Telemetry or EventLog Data	As a **Ground Station Operator** I want to **Request satellite telemetry or eventlog data** so that I can **verify/validate health status or mission data**.	
4.1: Request Satellite Health Data	As a **Ground Station Operator** I want to **request satellite health data packet** so that I can **verify/validate AODS sensors & GPS data are within nominal parameters**.	Request Telemetry or EventLog Data
4.1.1: Request Satellite Health Data via S-Band Radio	As a **Ground Station Operator** I want to **request satellite health data packet via S-band radio** so that I can **verify/validate AODS sensors & GPS data are within nominal parameters**.	Request Telemetry or EventLog Data Primary Mission Objective A2

ID / Name	Statement	Derived From
4.2: Request Satellite Mission Data	As a **Ground Station Operator** I want to **request satellite mission data** so that I can **validate in-orbit AODS and/or payload performance**.	Request Telemetry or EventLog Data Primary Mission Objective A1 Primary Mission Objective A3 Primary Mission Objective A5 Secondary Mission Objective B1 Secondary Mission Objective B2 Tertiary Mission Objective C1 Tertiary Mission Objective C2 Tertiary Mission Objective C3
5: Send Request to Set Mission Mode Duration	As a **Ground Station Operator** I want to **send a request to set mission mode duration** so that I can **manage time spent per mission mode**.	

Example Data Structure

An example data structure can be derived from user stories as presented in Table 3.

Table 3. Satellite Health Data Packet Specification

Field	Type	Item Type	Description
call_sign	string		Identifying call sign for the Sealion mission.
battery_health	float		Percent value indicating the remaining charge of the batteries.
temperature_battery	float		The temperature of the battery. Units in Kelvin.
mode	integer		Integer value indicating current mission mode. 0 = Safe, 1 = mission mode 1, 2 = mission mode 2, 3 = mission mode 3.
tle_data	TLE		TLE data from orbit propogator at time of beacon.

Derived From:

- View Satellite Beacon Data
- Request Satellite Health Data

Conclusion & Future Works

The methodology presented in this article is an effort to reduce the painpoints associated with traditional systems engineering for the CubeSat developers. The ever growing number of CubeSat projects in existence demands a lightweight model-based approach that can be adopted for implementing flight software while minimizing configuration management overhead. Docs-as-code can be used as an approach to flight software for tight coupling between both software and documentation as well as providing a highly structured template to base future developments on for the CubeSat community [4]. Future actions include validating this approach as the flight software is created based on the model-based docs-as-code approach.

References

[1] S. Friedenthal and C. Oster, *Architecting spacecraft with SysML*. .

[2] S. A. Asundi and N. G. Fitz-Coy, "CubeSat mission design based on a systems engineering approach," in *2013 IEEE Aerospace Conference*, Mar. 2013, p. nil, doi: 10.1109/aero.2013.6496900.

[3] "CAESAR Model-Based Approach to Harness Design," *Proceedings of IEEE Aerospace Conference*, Mar. 2020.

[4] "SeaLion Mission Architecture," [Online]. Available: https://github.com/ODU-CGA-CubeSat/sealion-mission-architecture.

[5] E. Holscher, "Docs as Code," [Online]. Available: https://www.writethedocs.org/guide/docs-as-code/.

[6] J. Simmons, "Mach30 Modeling Language," [Online]. Available: https://github.com/Mach30/m30ml.

[1] CubeSat Design Specification Rev. 13 https://www.cubesat.org/s/cds_rev13_final2.pdf