10th ANKARA INTERNATIONAL AEROSPACE CONFERENCE **AIAC-2019-187**
18-20 September 2019 - METU, Ankara TURKEY

APPLICATION OF MODEL-BASED SYSTEMS ENGINEERING WITH SYSML IN A SMALL SATELLITE PROJECT

Emre Canbulut[1] and Volkan Aydıngül[2]

Middle East Technical University
Ankara, Turkey

Burak Yağlıoğlu[3]

TUBITAK UZAY
Space Technologies Research Institute
Ankara, Turkey

ABSTRACT

For small satellite systems, systems engineering studies and precise and accurate simulations of the satellite are of great importance since all subsystems and components are physically connected and operate in a small volume. What is more is the fact that redundancy is not available most of the time in such spacecraft. In this study, the benefits of model-based systems engineering on such a system are investigated. Moreover, a certain methodology for modeling is proposed, which involves modeling the behavioral aspects of the system firstly and then the required structure to enable this behavior. After that, a simulation is built within the model and a graphical user interface is created. The benefits of this simulation are examined and it is found out that such a built-in simulation and user interface may even lead to an advanced, satellite-specific mission analysis and simulation tool for small satellites in exchange for a labour that is not significant if a SysML model of the system already exists.

[1] Senior Student, Department of Aerospace Engineering, Email: emre.canbulut@metu.edu.tr
[2] Senior Student, Department of Aerospace Engineering, Email: volkan.aydingul@metu.edu.tr
[3] Chief Researcher, Attitude and Orbit Control Group, Email: burak.yaglioglu@tubitak.gov.tr

INTRODUCTION

The authors of this paper are participants in APSCO Student Small Satellite (SSS) project and are members of the Turkish team who are developing SSS-2B small satellite which is based on a 3U CubeSat platform [Yağlıoğlu et al., 2019]. SSS-2B carries four payloads which are; intersatellite communication unit, optical camera, radiation dosimeter and propulsion system. The payloads are chosen regarding mission objectives. In this project, the authors research the application and benefits of model-based systems engineering (MBSE) methods for the CubeSat that is being developed. In Figure 1, SSS-2B is shown in the planned constellation.

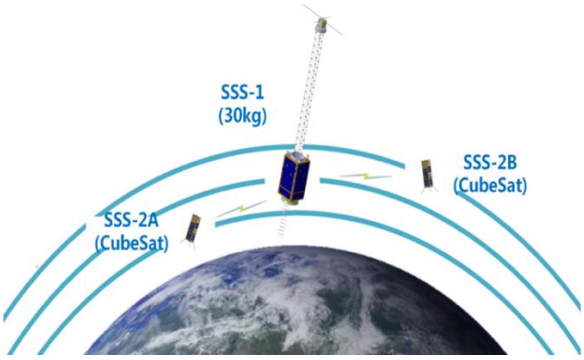

Figure 1. APSCO SSS satellites' constellation

A CubeSat or U-class spacecraft is a kind of miniaturized satellite which is made from a number of 10 cm x 10 cm x 11.35 cm cubic units [NASA CubeSat Launch Initiative, 2017]. 3U stands for a CubeSat built using three of these cubic units. The CubeSat standard allows affordable and relatively easier access to space since they are less costly and there are more opportunities for launching a CubeSat due to its small size. In the development of CubeSats, systems engineering processes are of great importance since in CubeSats a variety of subsystems and payloads operate in a small volume and most of the time redundancy is not available. Furthermore, CubeSats have their mass, volume, data and power consumption budgets as small as possible due to their design philosophy. Due to limited resources, precise systems engineering study is necessary for efficient utilization of these resources.

Model based systems engineering (MBSE) is a relatively novel approach to systems engineering problem. In MBSE, the main artifact of systems engineering processes is the system model, which is a consistent, integrated and comprehensive model of the physical system which is being designed and developed and its subsystems. MBSE offers its return of investment (ROI) especially when a change in the configuration is needed [Delligatti, 2013]. In contrast to traditional document-based systems engineering in which search of all locations of an artifact or a block along documents, diagrams, presentations, tables and matrices is required to make a change in the system configuration. However, in MBSE, changes automatically propagate throughout the whole model. With that being said, MBSE is an object oriented approach to the systems engineering problem and it can be categorized as a subbranch of Object Oriented Systems Engineering (OOSE) [Fermi and Lamberti (eds), 2013].

Model based systems engineering paradigm has numerous and easily deducable advantages over traditional document based systems engineering particularly when the system of interest has higher complexity, such as spacecraft. However, in this research, one distinct benefit of MBSE is taken into account and investigated which is the opportunity to integrate

the system model with programming languages or simulation toolkits such as MATLAB and Simulink in order to run the static system model as a dynamic simulation.

METHOD

Modeling Language

In order to apply MBSE, a modeling language is needed. A modeling language specifies the elements that can be used in a system model and the set of possible relationships between those elements. In case of a graphical modeling language, the language defines the notations and symbology that should be used to present model elements in diagrams. For this research, Systems Modeling Language (SysML) is chosen.

SysML is a graphical modeling language derived from Unified Modeling Language (UML). It is initiated by International Council on Systems Engineering (INCOSE) and published as an open standard by Object Management Group (OMG) [Weilkiens, 2007]. Its origin, UML, is used for modeling, design and analysis during the software development and production process. It is mostly used by software engineers in the industry. SysML is an extended profile of UML and it was created to be applicable for physical, technical and even sociological systems using the profiling mechanism of UML. In a sense, SysML extends the usability of UML beyond software systems by adding a new set of rules, new model elements, relationships and diagram types as well as altering the original ones in UML. In order to deploy SysML, a tool is needed to be the software basis of SysML, much like integrated development environments and compilers are for programming languages.

SysML offers nine types of diagrams to build a system model. These diagrams are categorized as "Behavioral Diagrams" and "Structural Diagrams" in terms of the way they represent the model. Behavioral diagrams represent the functionality of the system by giving an insight to the system's behavior in various situations, including tasks of the system and unexpected failure situations. These diagrams involve the functional relations between the components of the system such as which component activates which function and how it is invoked. On the other hand, structural diagrams represent the structural and logical build-up of the system as well as the physical relations, containment and connections such as which component is located on another part, or which component has the ports connecting it to another. Therefore, behavioral diagrams provide a dynamic view of the system while structural diagrams give a static view. Behavioral diagrams include; use case diagrams, activity diagrams, sequence diagrams and state-machine diagrams. Structural diagrams include; block definition diagrams, internal block diagrams, package diagrams and parametric diagrams. Lastly, there is requirement diagrams which are not categorized as either.

With its growing popularity, SysML becomes the industry standard for MBSE applications. In the table given in Table 1, percantages of SysML usage by industries are given based on a survey conducted by OMG [Luther, 2016]. It is no surprise that the active usage of SysML increases with increasing product complexity.

Industry	% of Active SysML Usage
Space	23
Aircraft	20
Defense	20
Automotive	7

Table 1. Usage percentages of SysML in some industries.

Software and Tools

For modeling SSS-2B, many SysML tools were considered. As can be seen in Table 2, every different tool comes with various features in terms of their cost and usability. Firstly, Modelio was used. Modelio provides an open-source extensible modeling environment. However, its interaction with the user was poor and it did not provide a simulation framework and opportunity to integrate MATLAB to the model. Then, IBM Rhapsody was considered. Although, it has suitable tools for simulation and integration with MATLAB, due to the financial issues, it was not the most appropriate option for the SSS-2B. Finally, Cameo Systems Modeler is chosen. Cameo Systems Modeler is a modeling tool with native support for SysML standard [Magic Inc., n.d.]. It should be emphasized that modeling tools such as Cameo Systems Modeler are different from plain diagramming tools. Modeling tools deploy a modeling language and provide automated consistency checks and validations of the model according to the set of rules defined by the modeling language deployed in the project. Therefore, by validating the system model through Cameo Systems Modeler, it is ensured that the model does not have any inconsistencies or errors in terms of model integrity and logic.

Software	Cost	Robustness	Simulation	Integrated modeling
Modelio	Free	Low	Low	Low
Cameo Systems Modeler	Medium	High	High	Medium
IBM Rhapsody	Very high	Medium	High	High
Papyrus	Free	Low	Low	Low

Table 2. Comparison of the SysML tools.

In an effort to simulate the behavioral installment of the system model, Cameo Simulation Toolkit is used. Cameo Simulation Toolkit comes off-the-shelf with Cameo Systems Modeler. It allows running certain types of diagrams as visual animations, which can be considered merely as behavioral simulations. However, the toolkit also supports integration with MATLAB's MathEngine and when combined with the calculations carried out inside MathEngine, the behavioral and visual simulations also attain the features of a numerical simulation with real-time calculations and decisions. Moreover, values and data types of the variables and parameters defined as model elements in the system model are used directly by MATLAB's engine without the need to declaration or value assignment in the MATLAB script. When these features are combined with the allowance in SysML standard to use opaque actions and control flow guards which can contain codes or functions in various programming or scripting languages including MATLAB, here rises an opportunity to natively build numerically enabled simulations within the system model.

Methodology and Philosophy

When practicing MBSE, systems engineers have a modelling language, a modelling tool and a system to model. However, there are infinitely many ways to model the same system. Hence, systems engineers should adopt a certain methodology to build the model. There are various standardized methodologies designed and provided for use by several companies or institutes [Holt and Perry, 2008]. In this research, a methodology in which firstly the behavior, functionality and responses of the satellite and its subsystems are modelled using behavioral diagrams and elements, and then the structure required to enable those behavior, functionality and responses are modelled such that the modelled set of behavior is backed up by logical and physical structure of the system. After that; the requirements, fundamental equations, parameters and analyses are integrated to the model in order to begin simulating the characteristics and workflow of the satellite. The selected subset of SysML diagrams for use in this work are given in Figure 2 below.

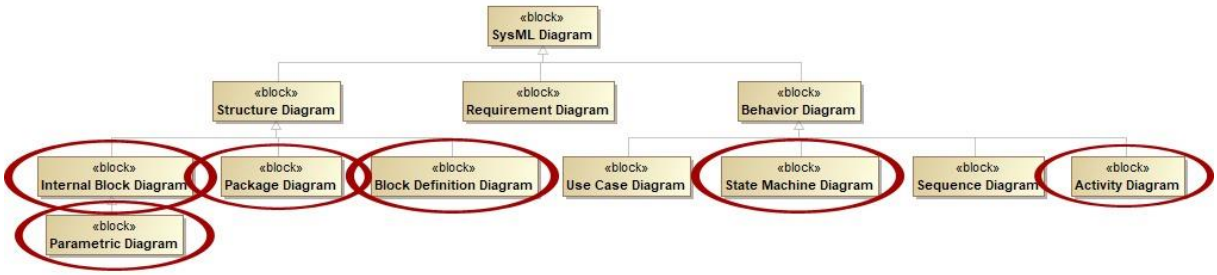

Figure 2. Selected subset of SysML diagrams

Usually, a system is modeled in SysML at early development stages. However, in the mentioned APSCO SSS Project, the system model was started building after the early design phases. Therefore, requirements diagrams are left out for the moment. However, necessary constraints derived from the requirements are implemented in block definition diagrams and internal block diagrams for simulation and testing purposes of SSS-2B. At this point, it should be noted that Cameo Systems Modeler supports natural language processing (NLP) for requirement statements written in the formal language defined by IEEE-SA Standards Style Manual or other well-known standards and interprets them automatically to extract constraints from those statements [No Magic Inc., 2017]. This is a useful feature when searching documents and transferring requirements into the model, particularly if a study involves creating the model of a system that has already passed preliminary or critical design phases.

Regarding the modeling philosophy, the system model was created in a way that is explorable and understandable for all stakeholders and developers from different integrated product teams. The details of SSS-2B SysML Model was kept at a level such that the model provides every detail about the satellite's structure and functionality that is given in a critical design report document. Moreover, along with component properties like mass, volume or power consumption, certain universal physical constants and parameters are also present in the model allowing quantitative simulation since the next stage of this research after modeling is simulating the satellite.

Speaking of simulation, there may be a variety of ways to dynamically simulate a SysML model. In this paper, the approach to simulation is to integrate the simulation inside the system

model. Furthermore, as long as possible it is tried to develop the simulation from the ground up inside the modeling environment along with the developing system model.

RESULTS & DISCUSSION

Modeling

Block Definition Diagrams: It is surely beyond doubt that block definition diagram is the most frequently used diagram type in the model constitution process. Not only does it enable modelers to reveal various types of model elements, but it also reflects the relations between model elements. The sole purpose behind this is to give information about the logical and physical structure of the system which needs to be modeled without exposing any discrimination between the software and the hardware. The discrimination between the hardware and the software is not a subject of the constitution process because how the segments of the system are associated with each other is more important than the actual physical connections between segments. To illustrate, in SSS-2B the control software is a component of the Attitude Determination and Control Subsystem, but it does not disseminate the idea that this component is a physical part of this subsystem, rather it is a logical component of ADCS. In Figure 3, the example block definition diagram that displays the general structure of the SSS-2B model can be shown.

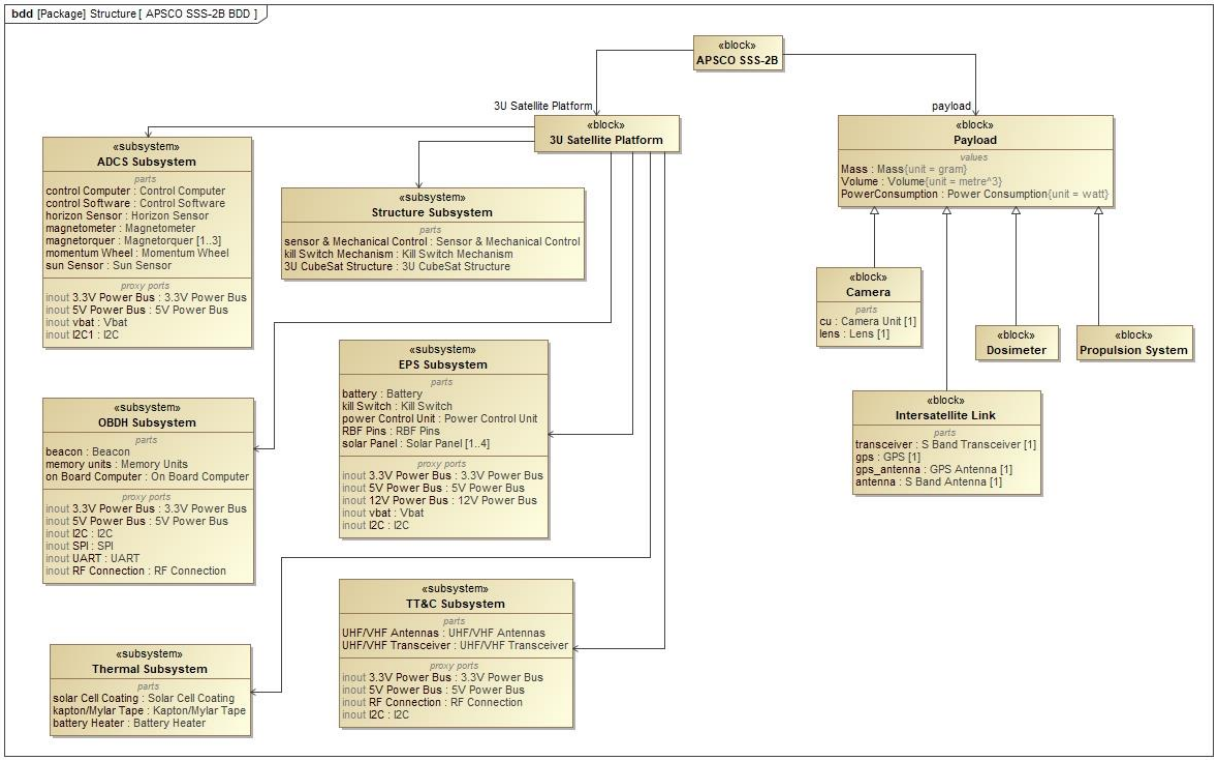

Figure 3. Block definition diagram of SSS-2B spacecraft segment.

A block is the most primivite and key element of SysML and block definition diagrams. It can contain ports that allows to transfer objects between blocks, value types that allows to define data structures and constraints that restrain the analytical domain of the system. Above all, the most practical attribute which block definition diagrams have is associations. Associations are the fundamental tools that can establish relationships between blocks and they reflect the logical and constructural interior of the system. Moreover, generalization is a special kind of association which organizes hierarchical scheme through the system model and

introduces inheritance to the system model. In SSS-2B, parent payload block and its four child block payloads are a good example of the generalization because all payloads have some properties in common such as mass and power consumption. Finally, this inheritance and hierarchy workflow allows information to propagate over the whole SSS-2B system.

Internal Block Diagrams: A block which is an essential unit of the block definition diagrams has parts which are the essential unit of the internal block diagrams. Internal block diagrams convey information about how the enclosed components of a block connected with each other in a logical and physical way. Generally, internal block diagrams become useful when the needs for seeking more detail emerges. An internal block diagram of a block may contain ports, interfaces, connectors and parts.

Ports and interfaces are densely interconnected pieces of the system model that builds a way of describing the flow of matter such as data, energy, service etc. While ports are representing the points that transitions into or out of a block happen, interfaces are representing the methods which enable ports to interact with each other in terms of services provided and demanded by components. All ports and interfaces are linked with each other via connectors. Connectors can be power bus, data bus or neither of both. It can also represent not substantial transmittal path at all.

In SSS-2B, representation of the data and power bus connection architecture is a critical part in the design and the development phase of the project. Here, internal block diagrams can be used for this purpose in the detailed design. In Figure 4, an example internal block diagrams of the Attitude Determination and Control Subsystem and Electrical Power Subsystem in SSS-2B and the parts which are involved in these diagrams are wired with each other as it can be observed looking at the diagrams.

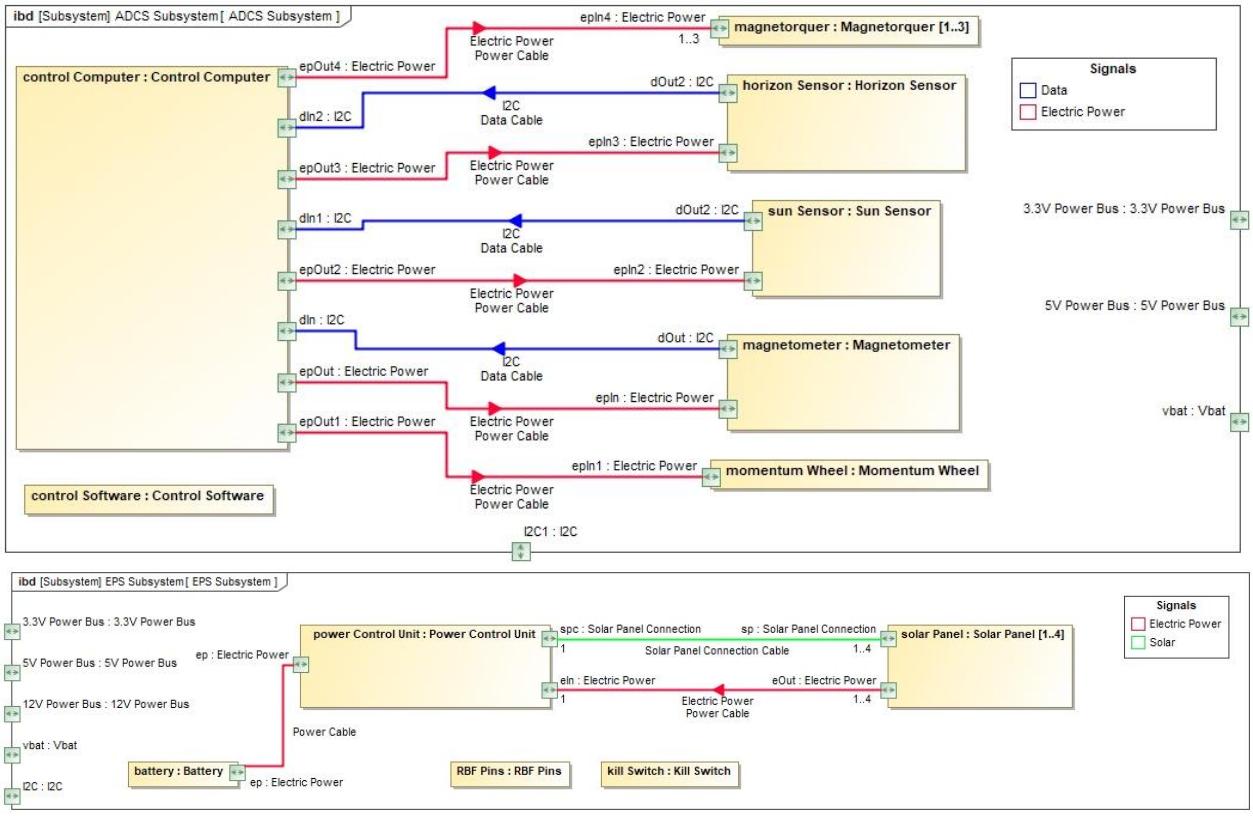

Figure 4. Internal block diagrams of ADCS and EPS.

Parts compose a massive section of the internal block diagrams and have the ability of containing and conveying the information about the block which is subjected to the creation of internal block diagram process. All technical details and information lie under how these lower level parts are connected to each other in an internal block diagram and this feature helps stakeholders gain a deeper understanding about the system and its components.

Package diagrams: Package diagrams act as an exploration guide and categorization system for both stakeholders and modelers. The fundamental element of the package diagrams is a package. A package is a SysML data structure to store different types of model elements. A package can be an aggregation of elements of any diagram, an aggregation of different diagrams or both of them in a way that these are varied. Package diagrams helps stakeholders to have an understanding about how the model is structured. In the same manner, it also helps modelers to build the model in a more organized way. There are three types of categorization that is suggested by OMG [Friedenthal, Moore and Steiner, 2008]. In Figure 5, these three types of model organization are given.

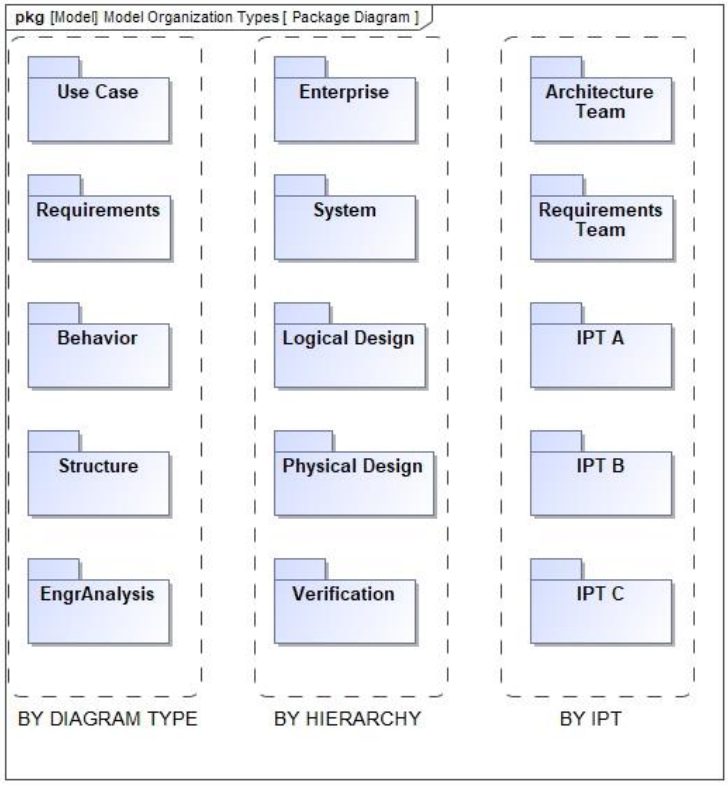

Figure 5. Model organization methods proposed by OMG.

In SSS-2B model, the main package diagram is structured by both diagram type and hierarchy concurrently. In this way, the package diagram of the SSS-2B system model is able to display the behavioral and structural views of the model together. In Figure 6, the organization package diagram of the SSS-2B model can be examined.

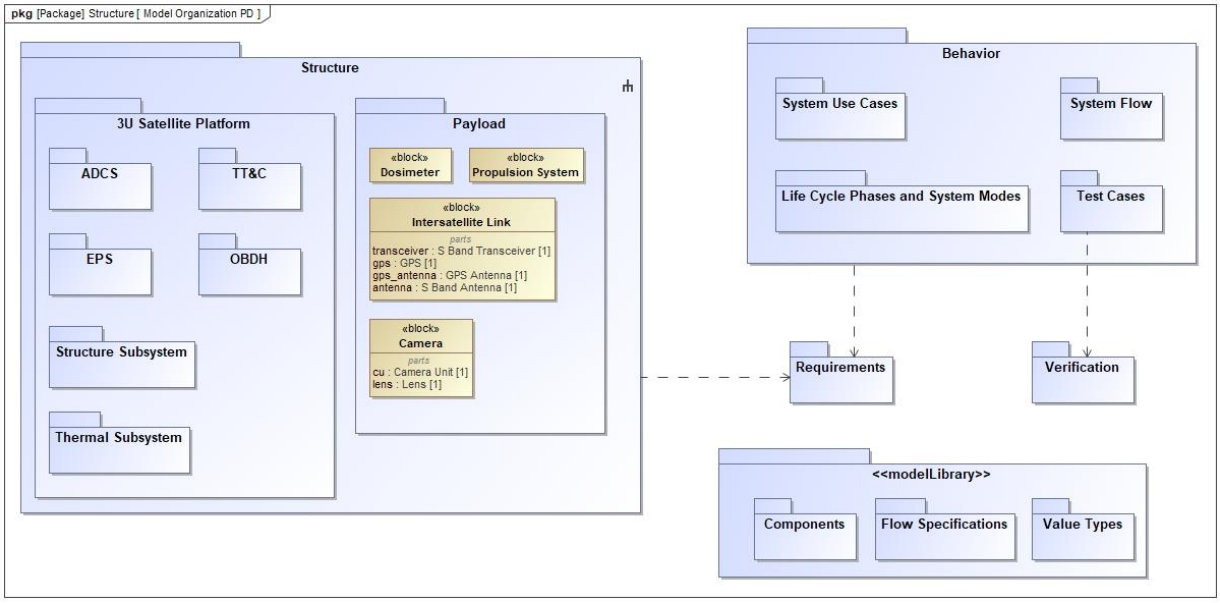

Figure 6. Model organization package diagram of SSS-2B.

Generally, all model package diagrams contain a package which is named <<modelLibrary>>, where the brackets indicate that this namespace is a stereotype. Model library package serves as a depository to store mostly repeated model elements. In the long run, it is aimed that the final version of the model library and well-detailed system model will serve as a general framework for CubeSat model developments, which may be included in future works of this project.

Parametric Diagrams: Parametric diagram is a sui generis type of diagram in SysML. It represents the computational domain of the system in a way that everything in the model is defined in a large space of constraints. It helps modeler to validate the system. In other words, modelers are able to check whether a given set of requirements are satisfied during some stage of the life-cycle with the help of the parametric diagrams. Normally, in the early design and development stages, all constraints restraining the SSS-2B model were embedded within the model frame. However, it is not adequate to address all the constraints to the blocks because that method does not declare specific computational flows between constraints and physical phenomena. Therefore, a need for using parametric diagrams emerged. Parametric diagram does not only convey the information of the constraints, but it also commentates how each parameter of each constraint is connected with each other and the direction mathematical information flows. In Figure 7, the example parametric diagram that shows the attitude maintenance which is involved in SSS-2B can be shown.

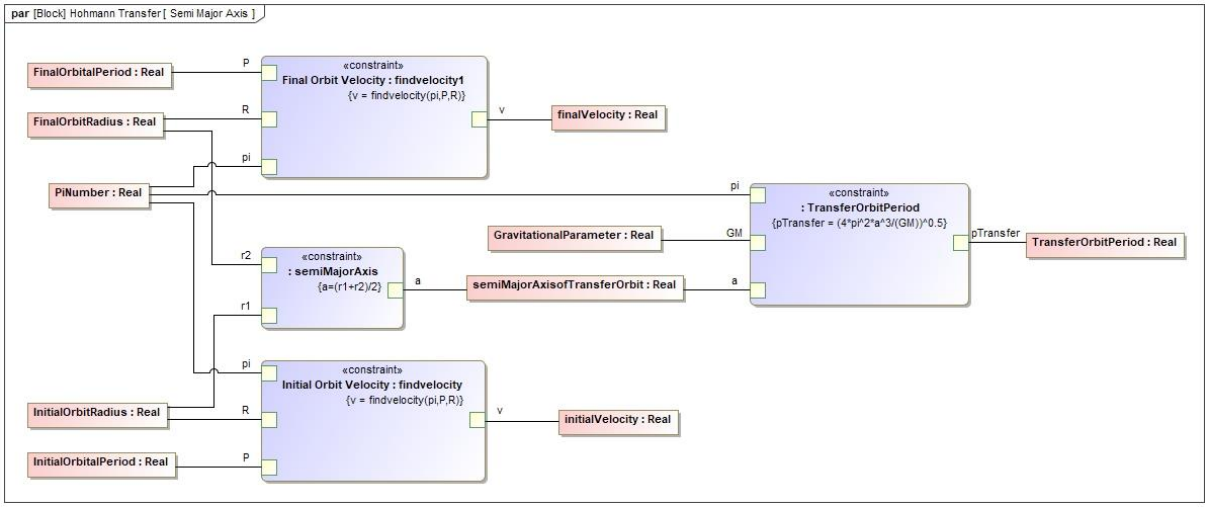

Figure 7. Example parametric diagram representing altitude maintenance.

The parametric diagram which is placed in the above figure contains several constraints. However, all constraints are not the built-in SysML constraints; some of them are created via MATLAB and added to the model with drag and drop method. Cameo Systems Modeler allows users to add MATLAB functions as constraints to the model directly. Also, doing this does not require the execution of the MATLAB at the same time. Cameo Systems Modeler recognizes the parameters in the function automatically and includes the necessary ports to the constrain block. In Figure 8, "findvelocity" function written in MATLAB and used as a constraint in the above diagram is given.

```
function v=findvelocity(pi,P,R)
    v=2*pi*R/P;
end
```

Figure 8. Sample MATLAB code used in constrain block.

Use Case Diagrams: The resulting use case diagrams provided a compact and comprehensive view of the services that the system provides and the interaction of operators and stakeholders with those services. The use case diagram in Figure 9 is prone to be confused with "scenarios", however, use case diagrams do not represent usage scenarios. The diagrams are a black box, high level view of the services provided by the system and how those services are related to each other and to the actors external to the system boundary. Instead of taking use case diagrams into account as scenarios, they can be thought including many scenarios that may occur along the services of the system and stakeholders.

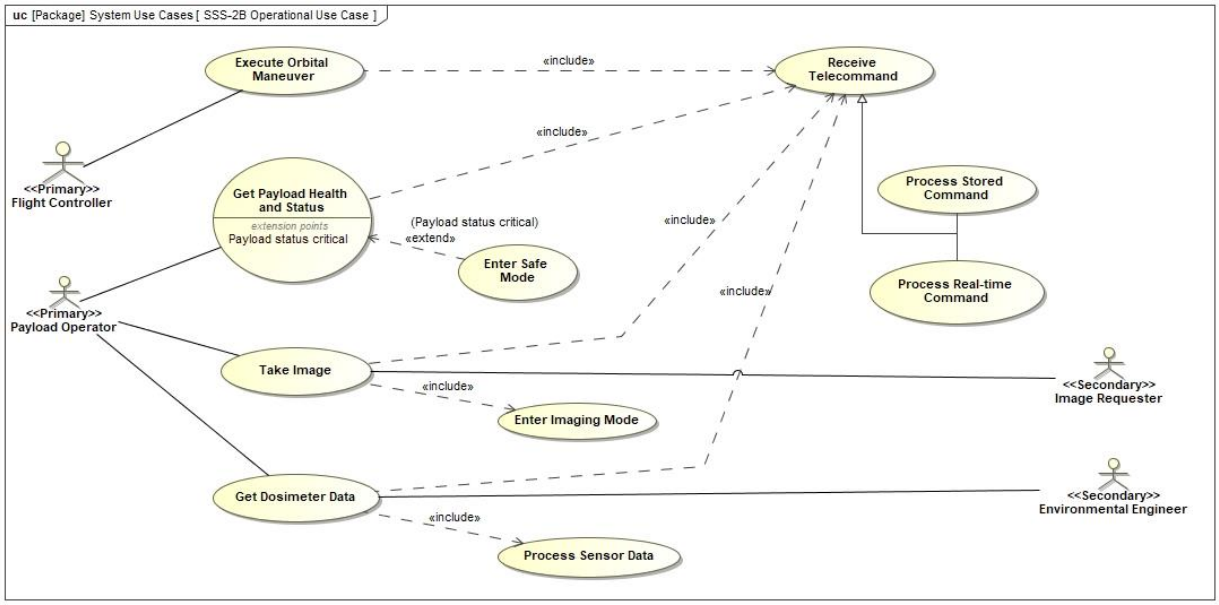

Figure 9. SSS-2B operational use case diagram.

Activity Diagrams: The resulting activity diagrams, such as the one given in Figure 11, provide an insight to the inner workflow of certain events, activities and use cases of the system. Activity diagrams provide more of a white box view of the system compared to use cases. Activity diagrams provide the flow of events in a certain order; however, they do not include the information of which components are active while the events are happening on a time basis as well as the information about the messages passed between components and inner checks and verifications of components or subsystems.

At this point sequence diagrams become handy. However, by using signal sending and receiving events in the activity diagram frame, the messages passed between components and inner checks of those components can be shown. In addition to this, by using swimlanes in the activity diagram frame, the components actively performing can be shown. The activity diagram in Figure 10 has partitions showing the components or subsystems of SSS-2B in which the actions are happening. Therefore, there is no special need for deploying sequence diagrams and leaving sequence diagrams out of the subset of diagrams chosen to work on becomes a good idea considering the complexity of the model.

Along with their main functions in a system model, activity and use case diagrams stand for an effective means of communication between integrated product teams thanks to being understandable without in-depth SysML knowledge.

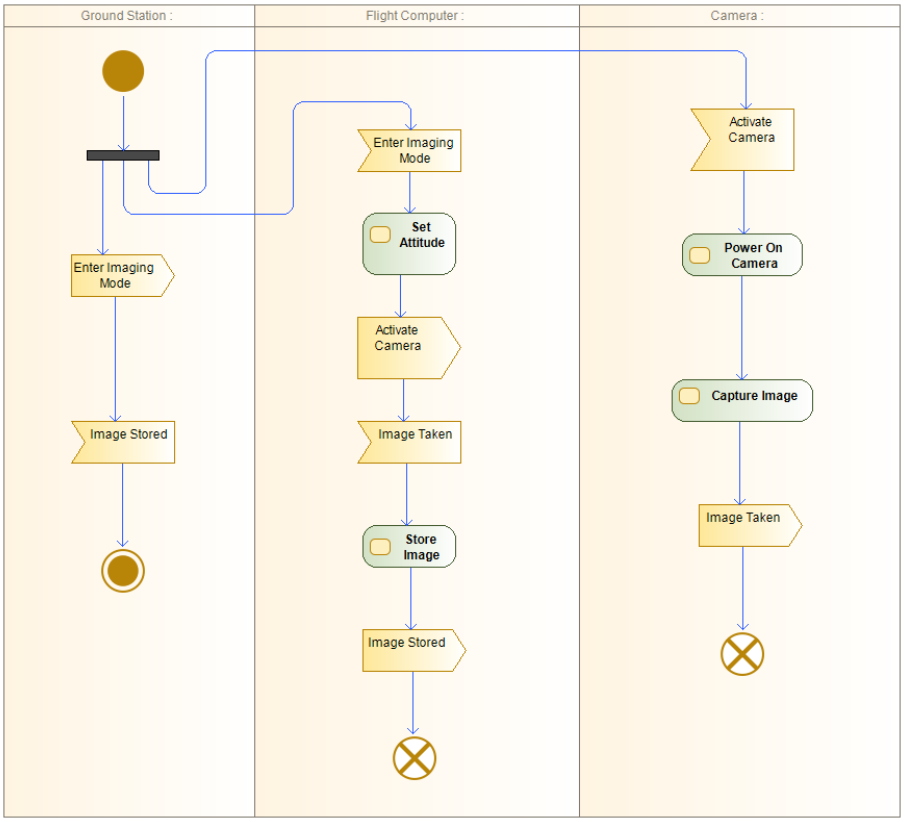

Figure 10. Activity diagram of imaging process of SSS-2B.

State-Machine Diagrams: State machine diagrams show the operational modes of the system. These diagrams include transitions between different states, the classifier behavior of those states which are the main jobs handled when entering the state, during the state and when finalizing the state. Entry and exit behaviors are considered "atomic" meaning that those actions cannot be interrupted and will occur certainly when entering and leaving a state, or an operational mode. For instance, in one of the state machine diagrams of the resulting system model given in Figure 11, it can be interpreted that the SSS-2B satellite is not able to enter nominal phase without going through nominal flow activity each time the nominal phase state-machine is activated.

At this point when speaking of state-machines, a clarification of the notions state, mode and phase is necessary. Moreover, a clear definition for the notion configuration is needed. In this research, configuration term is considered to provide a view of the system based on the combination and mechanical form of active components at a given time, which can be interpreted from the standard [ISO/IEC/IEEE 24765:2017]. As for other three notions, mode is defined as a "selectable option" that may be requested from an actor, a state is a performance based condition of the system and a phase is a segment of the product life-cycle [Wasson Strategics LLC, 2014]. In this paper, the definition of a mode is extended such that the mode option can also be requested by the system itself, for instance power safe mode is initiated by the system automatically when there is a critical battery condition. Power safe mode and attitude recovery mode along with end-of-life (EOL) phase are placed outside the borders of normal state state-machine since these modes are selected or EOL phase begins at certain situations where the satellite can not perform properly.

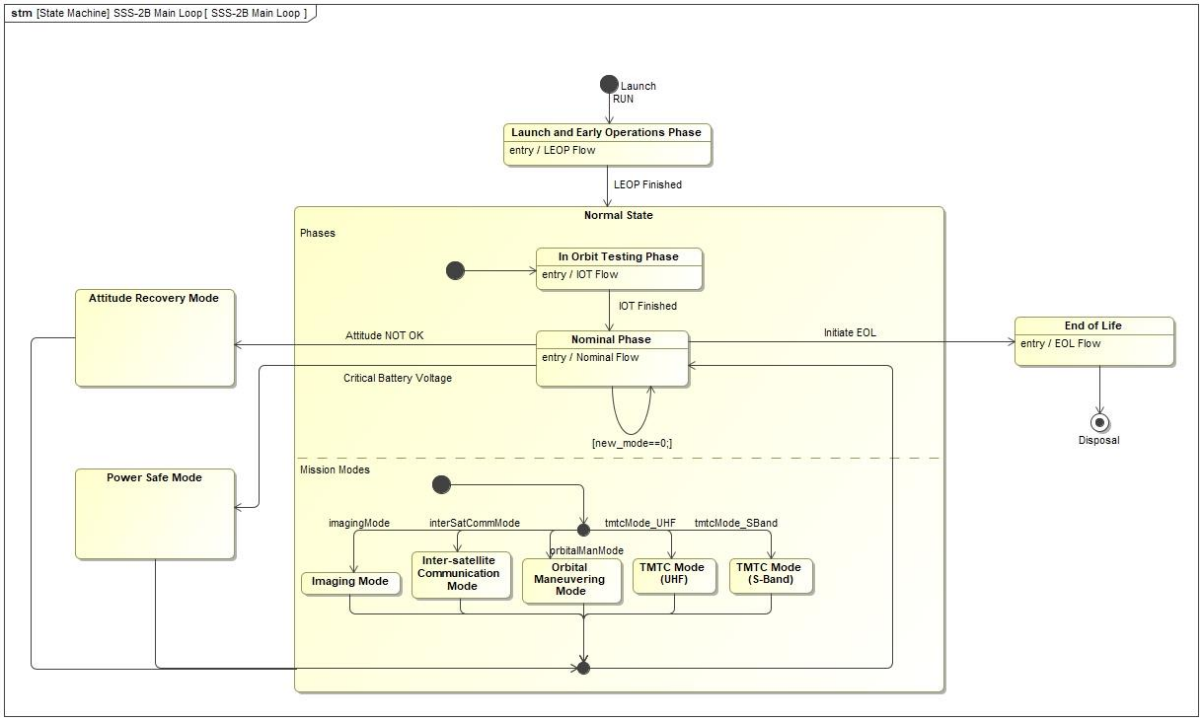

Figure 11. SSS-2B mission state machine diagram.

Automated Model Validation: Automated validation is another key feature of SysML and the software used. Automated validation enables modelers to observe the illegitimate orientations and it provides early warning mechanism in the design process. In this way, design procedures yield flawless systems. Automated validation system checks consistency of the system all the time as the compilers do for programming languages. While the compilers seek for an error in syntax in programming languages, the automatic validation systems seek for an error in usage of model elements in the proper context in SysML.

Simulation and Testing

Bringing the Static System Model to Life: State Machine diagrams show more of a black box view of the system whereas activity diagrams can be considered a white box view of the internal workflow of the system at a lower level, which are traced from the state machines by assigning the related activity as entry, do or exit behavior of certain states. Use cases, on the other hand, may be used to provide the system's services to stakeholders at the highest level of abstraction. Therefore, use case diagrams are not included in the simulation package of SSS-2B for the sake of model simplicity. In the simulation package, the main satellite operation loop and household functions of the system are shown in the "SSS-2B Main State-Machine Diagram" given in Figure 12.

The behaviors of these state elements in the diagram are defined by middle step activity diagrams traced from the states, like the one given in Figure 13. This middle step is required to input necessary objects like mathematical constants to the activity which is on the other end of the connecting diagram.

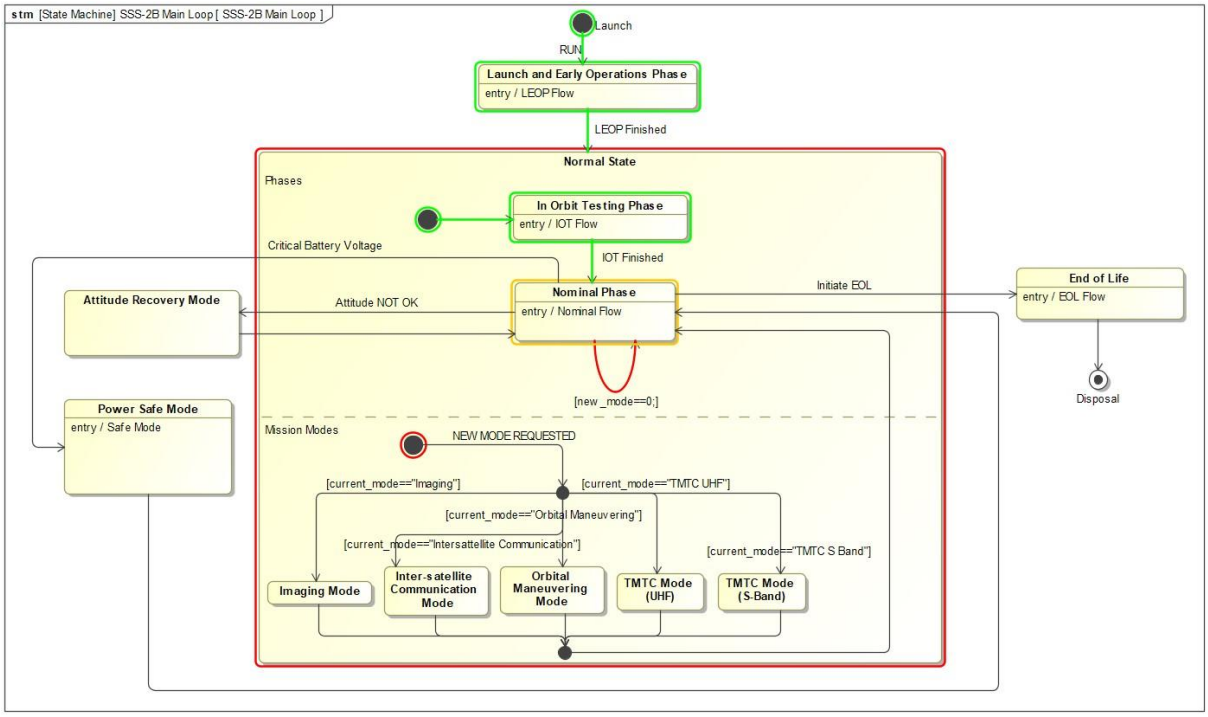

Figure 12. Screenshot of Main Loop State-Machine Diagram of SSS-2B during simulation.

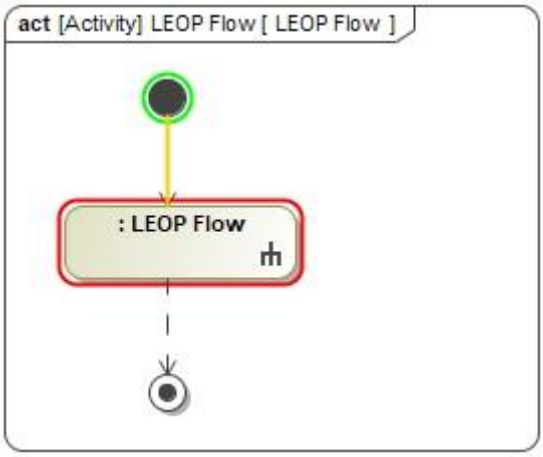

Figure 13. Middle step activity diagram connecting the background activity diagram to the main state-machine diagram.

When the simulation is run, the simulation toolkit starts with the main state-machine diagram and does the necessary jumps to lower level activity diagrams such as "Nominal Flow" activity given in Figure 14. Then, automatically turns back when an inner procedure is finished. All signals, parameters and alterations of values are carried through the model during these jumps. Therefore, the behavioral aspects of the satellite model become interconnected as well.

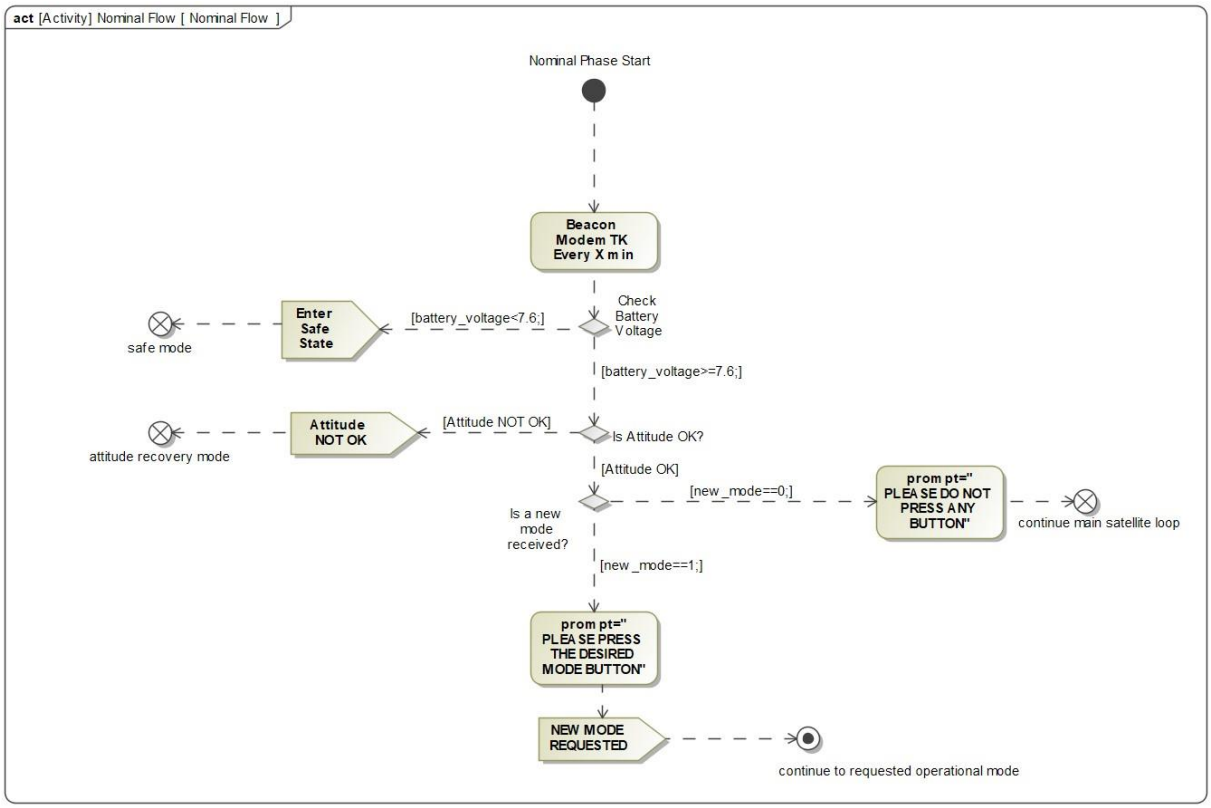

Figure 14. Activity diagram of nominal household loop of SSS-2B.

In the activity diagram frame, usage of opaque actions and guards are allowed. Opaque actions are actions composed of executable scripts written in one of the supported programming languages and they are executable statements which either alter some variable values or display some outputs in the built-in console. Opaque guards are scripts that result in boolean data type that they yield whether a decision turnout is true or false. These kind of guards are executed in MATLAB Engine and the simulation continues in a certain path according to the resulting boolean value. However, the modeler may choose not to use a scripting language but to use yes or no prompts graphically. Examples of both of these scenarios are used in the launch and early operations (LEOP) phase activity diagram given in Figure 15.

There is also a GUI developed right inside the model, in the simulation package, that accompanies the main satellite loop simulation. Because most SysML tools also support UML which is the origin of SysML and used for software development, there are tools to create a user interface mock-up in most modeling tools. Combining this ability to create UI mock-ups in the modeling tool with simulation capabilities and programming language integrations results in a potential to produce an utilizable and serviceable interface with a mission and/or system analysis software right inside the model. For instance, in a scenario where there is an imaging task and an orbital maneuver planned right after the imaging task, such a tool would not only calculate the imaging opportunity but would also be able to calculate the electrical power status of the satellite and show whether the orbital maneuver is possible or not in a unified hub. There is no doubt that a satellite-specific analysis software developed along with the MBSE utilization of the satellite would be able to simulate much more detail and assist more efficient mission profiling and life-cycle maintenance in exchange for a low labor cost in view of these benefits.

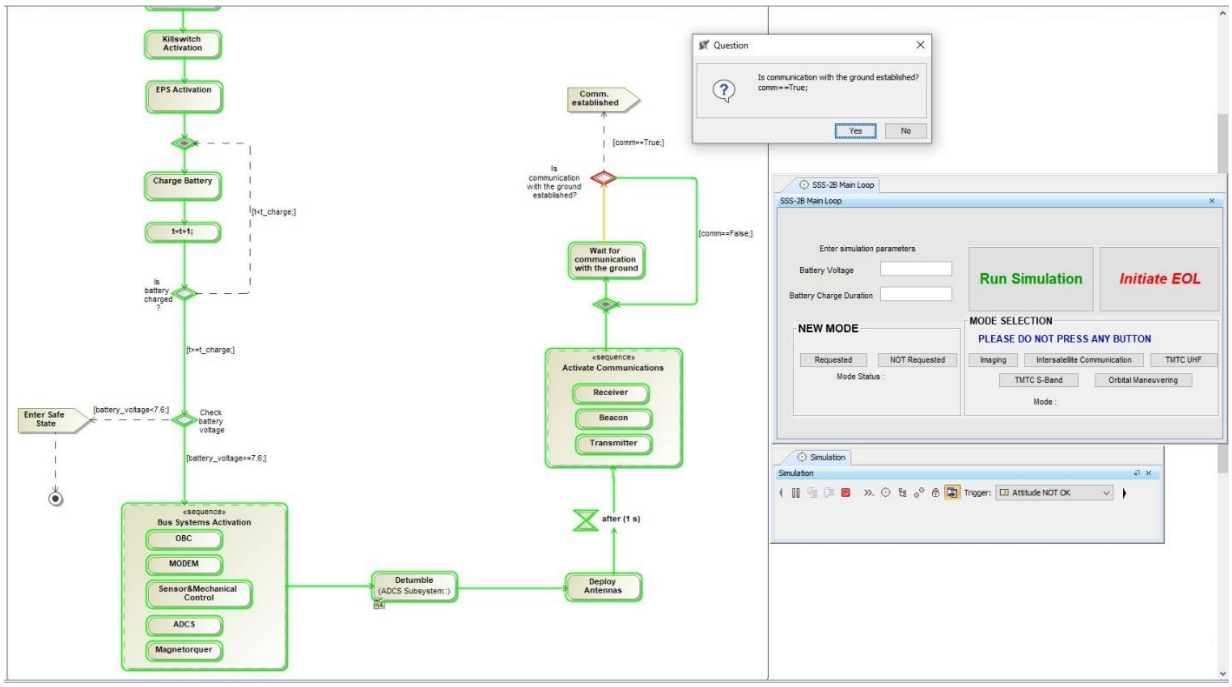

Figure 15. LEOP Flow activity diagram during simulation.

Using this GUI given in Figure 16, the user is able to input simulation parameters and send necessary signals in real-time such as initiating EOL phase. These inputs are done with easy to use interface elements such as textboxes and buttons. In addition to inputs, the user is able to read data from the simulation such as new mode request status or values of time-variables of interest.

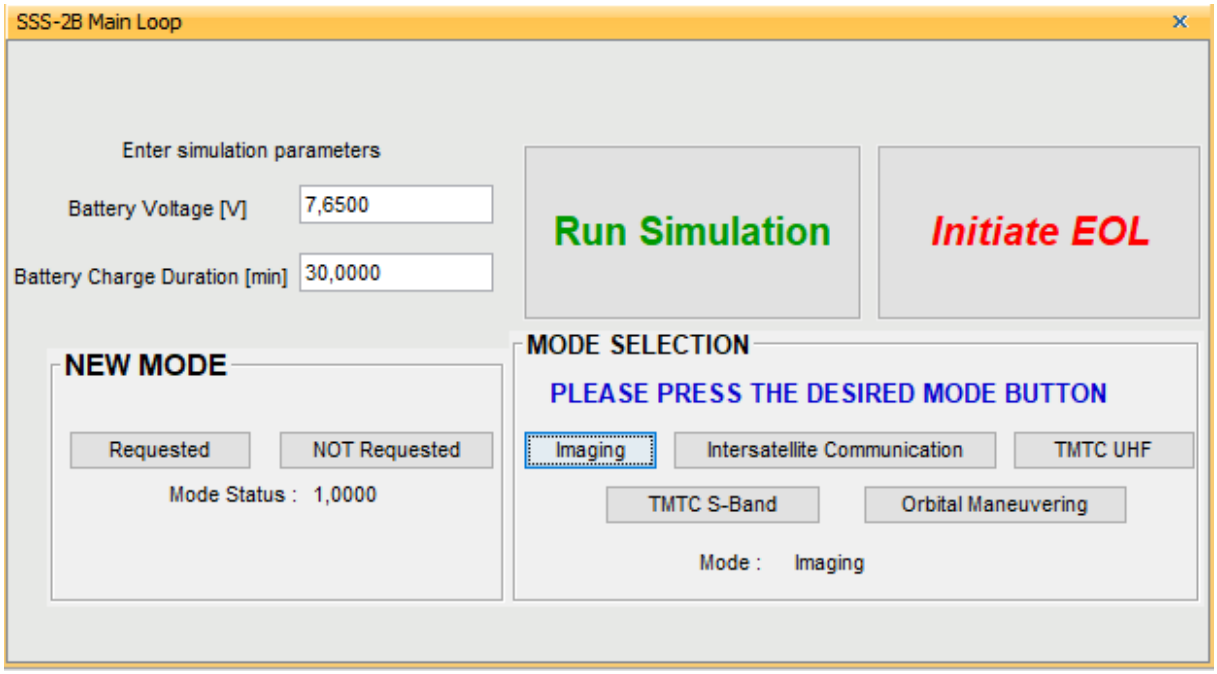

Figure 16. Graphical user interface accompanying SSS-2B model simulation.

Validating the Budget and Usage of Resources: In the all design and development stages of the SSS-2B, budget and resource analysis occupy a huge place. To be able to deal with this situation, the rollup analyses and requirements verifications are used. Rollup analysis of a block aims to detect the gross value of a property by taking into account the all values which belong to the subdivisions of this block. In Figure 17, the block that is subjected to the rollup analysis and its requirement and constraint block can be marked.

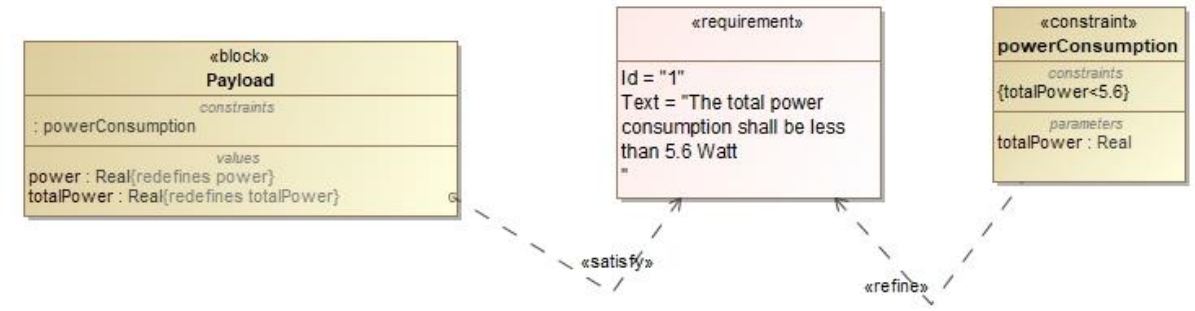

Figure 17. Requirement and generated constraint inside block definition diagram

The rollup analyses can be produced with the cooperation of the parametric constraints and the blocks. Besides, the live feedback can be provided via parametric diagrams. In Figure 18, the parametric diagram that shows the total power constraint and rollup analysis results can be shown.

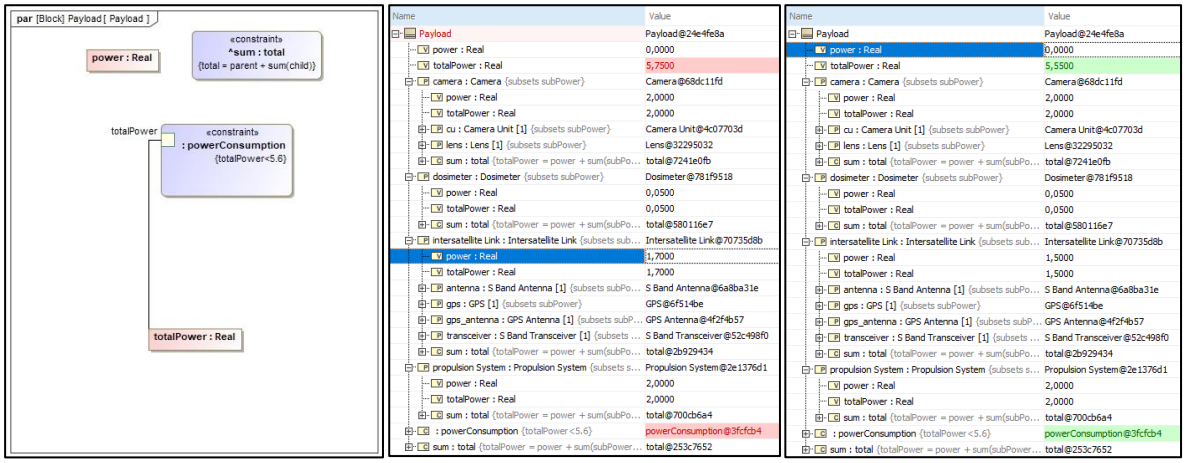

Figure 18. Screenshots of results from the session of running the parametric diagram with Cameo Simulation Toolkit.

In SSS-2B case, a power consumption rollup analysis is implemented. In this analysis, the power consumption of every element in the payload block is taken into account and with the help of existing constraints, the payload power consumption is simulated and verified.

Testing SSS-2B Satellite: SysML is an extension profile of the UML, which is a modelling language to develop software projects. Thanks to this feature, most of the SysML tools allow modelers to design and construct user interface mock-up within the model simulations. Adoption of the user interface mock-ups becomes very useful when the demand for test procedure emerges. Test procedures requires the collaboration of the many tools in the SysML and the software. In SSS-2B, test procedures are controlled by the parametric diagrams and the flow of the test is navigated by the user interface mock-up. In Figure 19, the parametric diagram which identifies the test constraints can be examined.

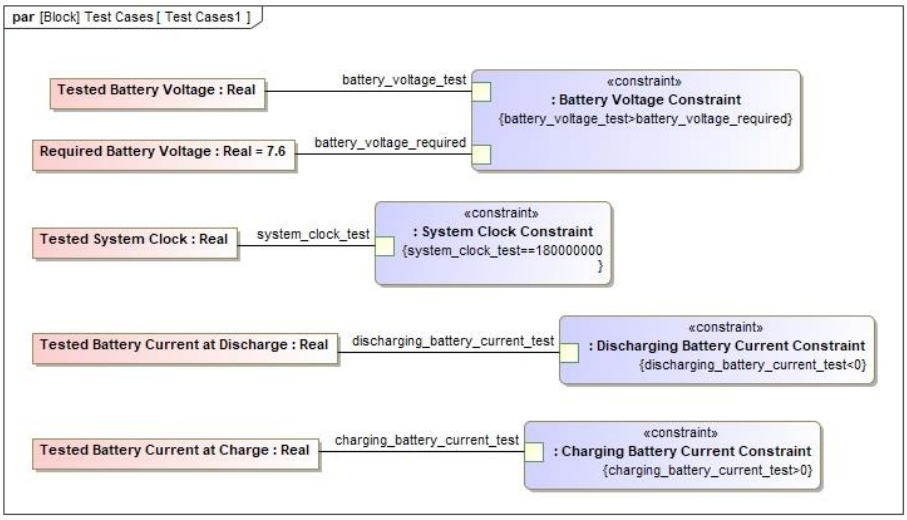

Figure 19. Test case parametric diagram

Contrary to traditional test procedures which requires real-time logging and comparison with respect to the flow of the test data, this collaborative method which is included under model-based systems engineering provides robustness to human errors.

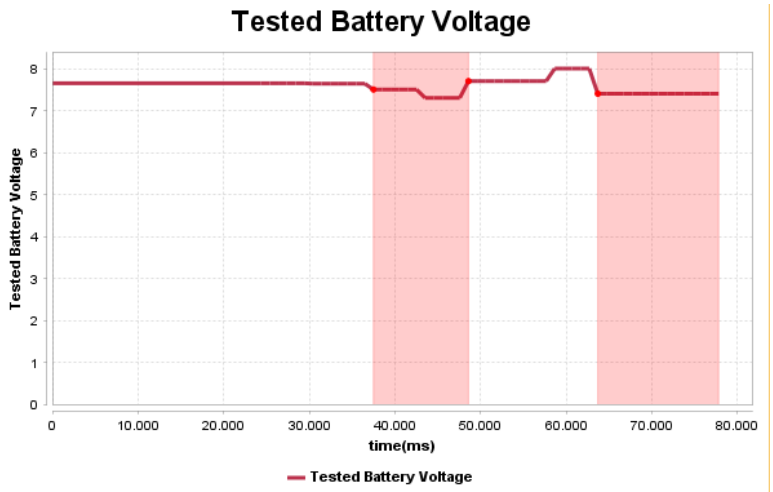

Figure 20. Battery voltage time-series chart during a test mock-up.

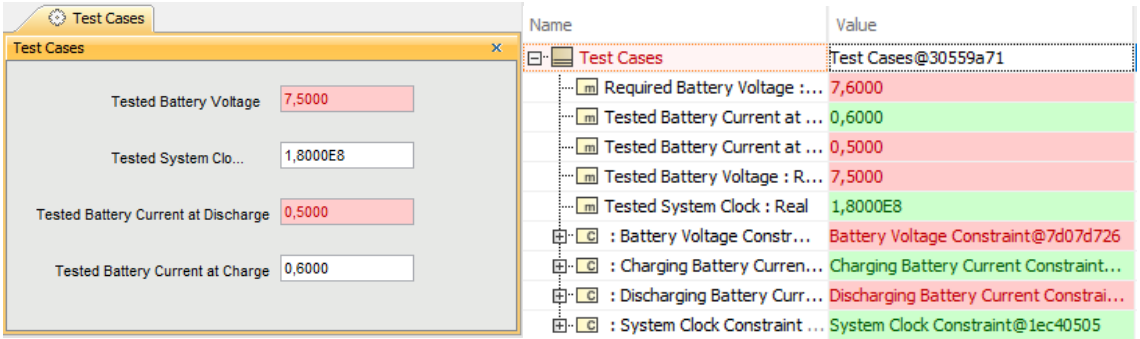

Figure 21. Input/Output GUI with examples of successful and unsuccessful validations.

As can be seen in Figures 20 and 21, SSS-2B involves an interactive test assembly, which involves a graphical user interface mock-up, time series chart and real-time data table. Therefore, a test operator can input a test variable to the system and observe how actually system responses to this input by the virtue of the color codes standing in the data tables and

the time series charts. Furthermore, the data acquired during the test process can be used in system analysis matrixes and requirements verification procedures later on.

CONCLUSION

During the modeling process, it was found out that SysML does not impose strict rules on the modelers about the way they use the provided model elements. Instead, SysML offers the types of diagrams and a range of model elements like a sandbox from which the modeler may take the necessary ones and bring various elements together in a way that makes sense for the modelers' goals. Therefore, every team that is going to practice MBSE or study simulations using SysML should adopt or develop a methodology and create sets of rules among the teams regarding the necessities of modeling and simulating the system of interest.

Throughout the study in this paper, a certain recipe for building a practically simulable system model is proposed. This practice asserts the usefulness of creating a main state-machine diagram to be the front face of the model and enlarging it upon details of certain events happening inside the system by the help of activity diagrams which are interconnected this face of the model. In SSS-2B model, use case diagrams are not seen essential in simulation and thus left out of the simulations for keeping the model complexity at the minimum as possible. On top of it, it is advised to build a GUI along with the simulation construct inside the model. A GUI will provide an applicative instrument for input and output by giving an orderly hub to manage the simulation and tests.

Lastly, the possibility to interface or embed a mission and/or system analyses software inside the system model synchronously with the design and development processes of a satellite was shown in this work. A satellite-specific mission profile tool encapsulated within the model would give operators more insight to working of the satellite which would in turn verify the use cases as well. In addition to that, a test model which is directly connected to requirements with a dependency link in SysML would provide test engineers with a better interface for typing and recording test data compared to traditional text file logs or seperate software.

FUTURE WORK

The future work of this study is developing a full-scale simulation inside the system model along with a graphical user interface in order to take place of general mission analysis tools and extend our capability to simulate a small satellite.

ACKNOWLEDGEMENTS

We would like to thank to No Magic Inc. for providing us with an academic license and GUMUSH Aerospace and Defense for providing us with CubeSat system inputs. Also, we would like to thank to Özgür Eser Akdemir for his introduction to MBSE and our friends Beril Günay, Merve Saraç and Şulenur Altunay for their support in this study.

References

Delligatti, L. (2013). *SysML Distilled.* Crawfordsville, IN: Addison-Wesley.

Fermi, E., & Lamberti, A. (2011). *Advances in Systems Engineering Research.* Hauppauge, N.Y.: Nova Publisher.

Friedenthal, S., Moore, A., & Steiner, R. (2008, 04 15). OMG Systems Modeling Language (OMG SysML™) Tutorial. INCOSE.

Holt, J., & Perry , S. (2008). *SysML for Systems Engineering.* London: The Institution of Engineering and Technology.

Luther, S. T. (2016). *SYSML BASED CUBESAT MODEL DESIGN AND INTEGRATION WITH THE HORIZON SIMULATION FRAMEWORK.* San Luis Obispo.

Magic Inc. (n.d.). *Cameo System Modeler*. Retrieved from No Magic: https://www.nomagic.com/products/cameo-systems-modeler

NASA CubeSat Launch Initiative. (2017, 10). CubeSat 101: Basic Concepts and Processes for First-Time CubeSat Developers. San Luis Obispo, California, USA.

No Magic Inc. (2017). *CAMEO CONCEPT MODELER PLUGIN.*

Systems and Software Engineering - Vocabulary. (2017). *ISO/IEC/IEEE 24765:2017*.

Wasson Strategics, LLC. (2014, 10 29). System Phases, Modes, and States Solutions to Controversial Issues.

Weilkiens, T. (2007). *Systems Engineering with SysML/UML.* Heidelberg: Denise E. M. Penrose.

Yağlıoğlu, B., Ataş, Ö., Kahraman, D., Köse, S., Tekinalp, O., Süer, M. (2019), "*A MULTI-NATIONAL MULTI-INSTITUTIONAL EDUCATION FRAMEWORK: APSCO SSS-2B CubeSat Project* ", IEEE Xplore/RAST 2019, Istanbul, Türkiye, 11-14 June 2019.