System Design Document

For

MBSE Avionics System Capstone

Team members: Walter Hernandez, Shawn Merrick Miller, Luke Newcomb, William Pate

Version/Author	Date
0.1.0 / William Pate	9/28/23
0.1.0 / Walter Hernandez	9/28/23
0.1.1 / Luke Newcomb	9/29/23
0.2.0 / Shawn Merrick Miller	10/31/23
0.2.1 / William Pate	10/31/23
0.2.1 / Walter Hernandez	10/31/23
0.3.0 / Luke Newcomb	10/31/23
0.3.1 / Luke Newcomb	11/7/23
0.3.2 / William Pate	11/14/23
0.3.2 / Shawn Merrick Miller	11/14/23
0.3.4 / William Pate	11/20/23
0.4.0 / Walter Hernandez	11/21/23
0.4.0 / Luke Newcomb	11/21/23

TABLE OF CONTENTS

SYSTEM DESIGN DOCUMENT

1 INTRODUCTION

1.1 Purpose and Scope

This document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces for the Model-Based Systems Engineering (MBSE) Avionics Capstone (MASC) project.

1.2 Project Executive Summary

The following section provides a description of the MASC project from a management perspective and an overview of the framework within which the conceptual system design was prepared. The MASC project will include an evaluation of system modeling using the MagicGrid framework to capture the problem domain for a notional Crewed Cislunar Station (CCS), and the solution domain modeling for the avionics system. The purpose of this model is to evaluate the system behavior and determine the ability to identify failure modes to expand Failure Mode Effect Analysis capability.

1.2.1 System Overview

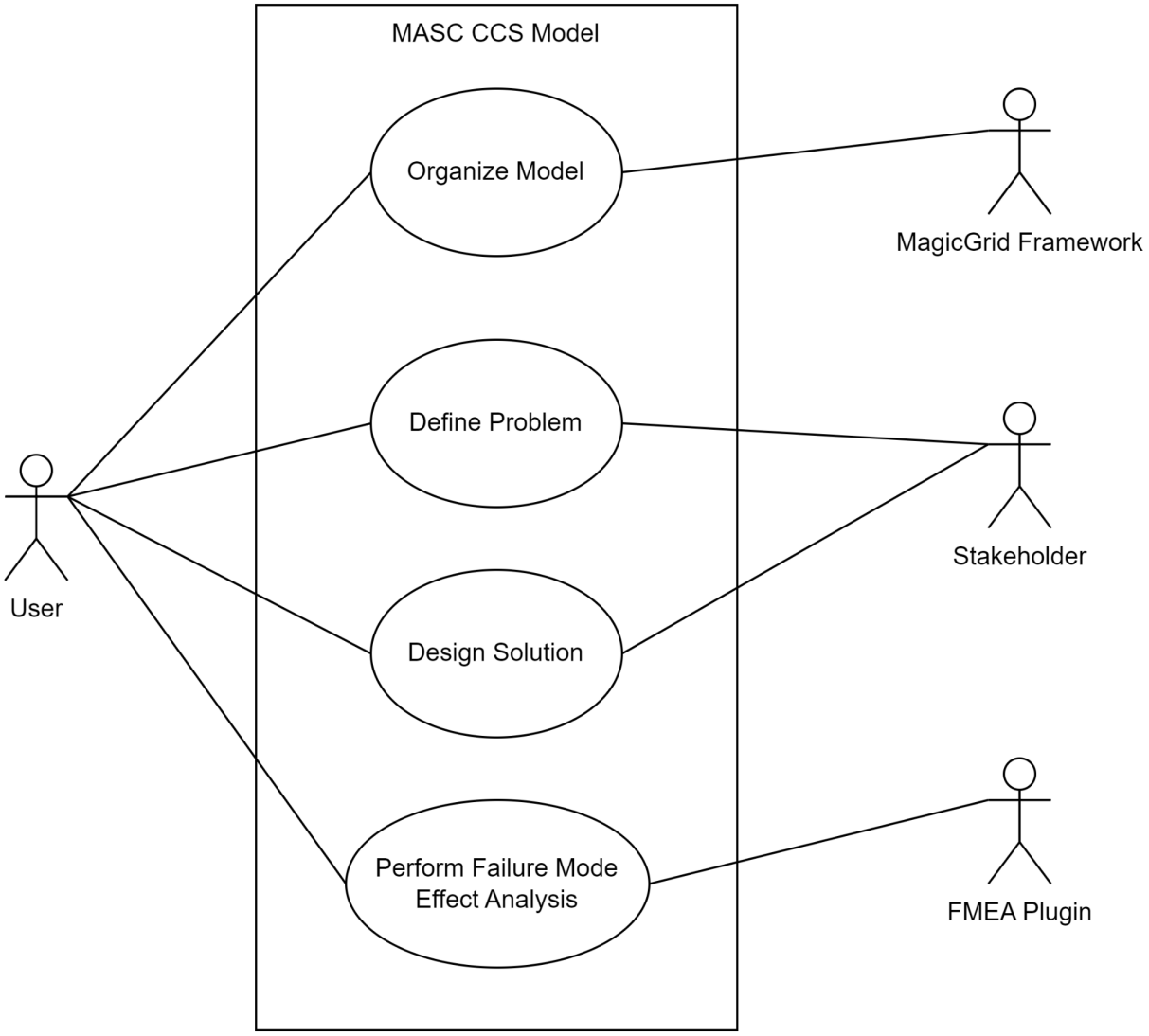

Figure 1: MASC Generic Avionics Use Case Model

This use case diagram describes the system as a completed model which is produced using the process that the team is exploring. All engineering user classes are combined here into a User actor which will perform two actions on the system: Perform FMEA Analysis, with the support of the FMEA Plugin, and the option to specialize the model into a real system model which can be used for a spacecraft design.

1.2.2 Design Constraints

The system design is constrained to the CCS mission discussed in the System Requirements Specification using the MagicGrid framework for model organization.

1.2.3 Future Contingencies

At the moment the constraints of the Magic System of Systems software and the constraints of the MagicGrid framework are unknown. These could both influence the direction and depth of the project by a great amount.

1.3 Document Organization

The following sections will provide information on what the product does, limitations, interactions, interfaces, hardware and software designs, and security.

1.4 Project References

MagicGrid Book of Knowledge, 2nd Edition, from
https://discover.3ds.com/magicgrid-book-of-knowledge

1.5 Glossary

CSS - Crewed Cislunar Station
MBSE - Model-Based Systems Engineering
FMEA - Failure Mode and Effects Analysis
SOI - System of Interest
RAM - Random Access Memory

2 SYSTEM ARCHITECTURE

2.1 System Hardware Architecture

System as described does not contain any external hardware components at this time.

2.2 System Software Architecture

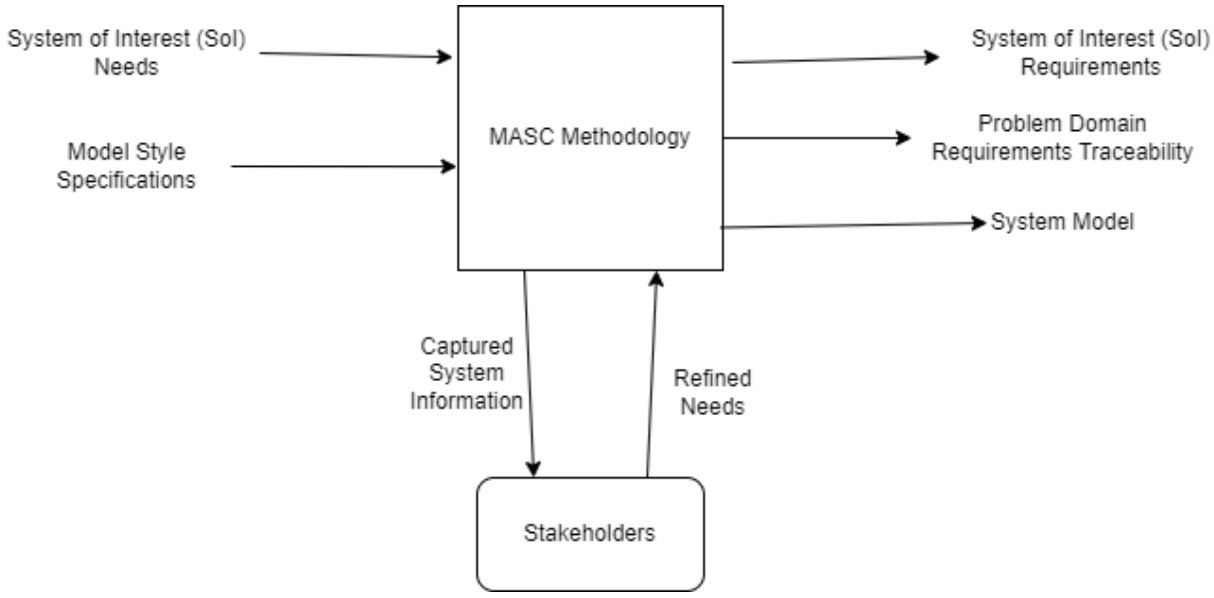

Figure 2: MASC System Architecture

The system architecture describes the way in which the MASC methodology is iterated and updated through the process of modeling a System of Interest (SoI). The process receives an initial set of 'Needs', as well as a set of standards for which the models will be built. During the design process, the methodology allows for the 'captured system information' at a current iteration to be sent to the stakeholder for review. From this, a set of refined needs will be returned to the process to be utilized during the next iteration. Once the process is finished, a set of requirements, the system model, and the ability to trace said requirements are the final given result.

2.3 Internal Communications Architecture

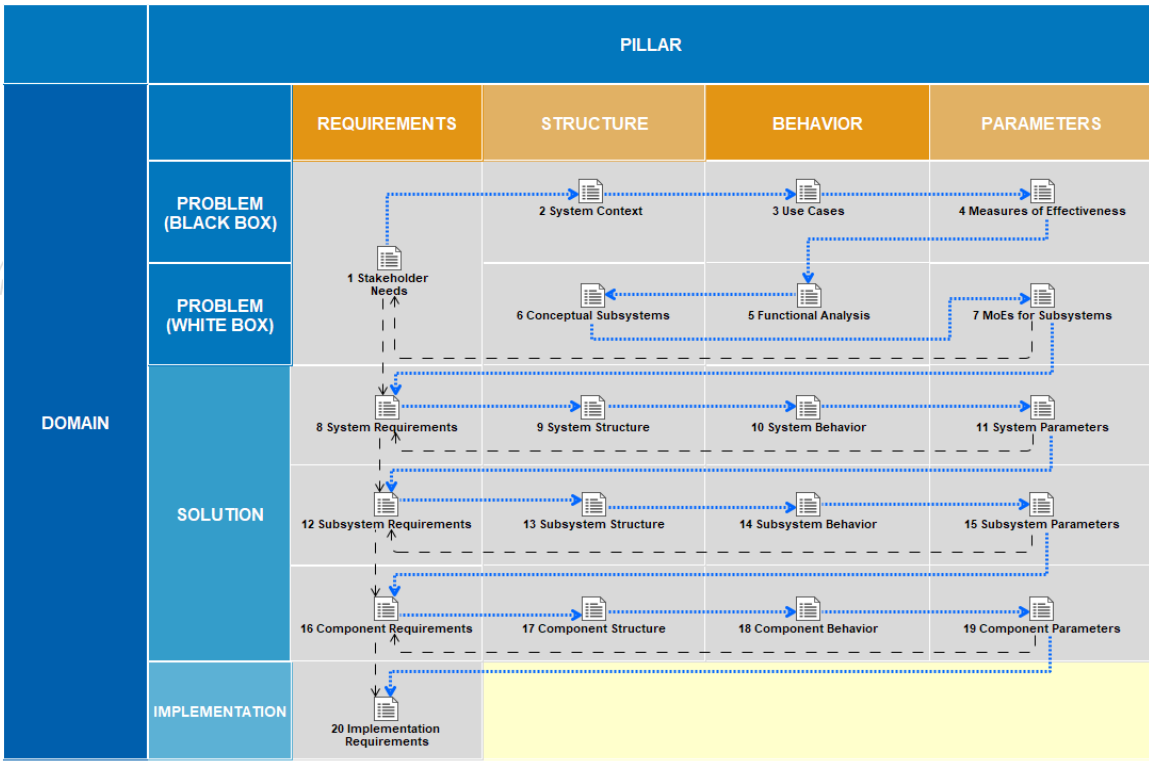

Figure 3: Internal Model Flow

Within the MagicGrid Framework, the internal model flow shows the direction in which views of the system influence each other. Each view is utilized the views following it, with some models having a direct influence (shown by the black dotted lines). The normal sequence of models is shown by the blue dotted lines. The domain of our system is separated into sections of: problem (black & white box), solution, and implementation. The views are also separated into functional categories of requirements, structure, behavior, and parameters.

3 HUMAN-MACHINE INTERFACE

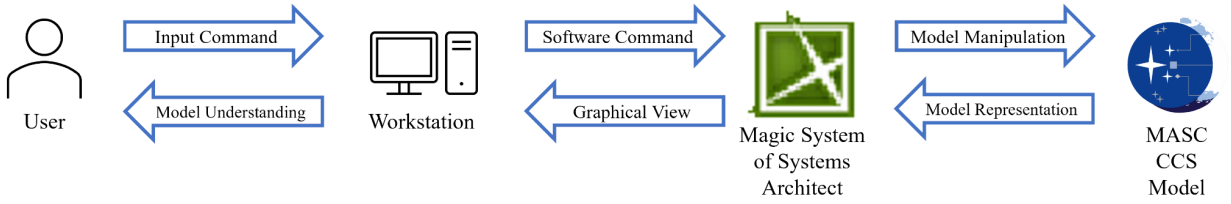

Figure 4: MASC Human-Machine Interface Architecture

The inputs and outputs relative to the user/operator from the system are implemented via the workstation, Magic System of Systems Architect Software, and the MASC CCS Model. Figure 4 represents this exchange as a typical interaction using a software interface. The user is only able to manipulate the model with a workstation which is compatible with the software, and the software is the only means by which that model can be used. No unique design or constraints need to be considered.

4 DETAILED DESIGN

This section provides the information needed for a system development team to actually build and integrate the hardware components, code and integrate the software modules, and interconnect the hardware and software segments into a functional product. Additionally, this section addresses the detailed procedures for combining separate COTS packages into a single system. Every detailed requirement should map back to the FRD, and the mapping should be presented in an update to the RTM and include the RTM as an appendix to this design document.

4.1 Hardware Detailed Design

The hardware constraints for the system are that the model must be contained within the Magic System of Systems Architect 2022x software. The software will be running on Windows 10 laptops which contain greater than or equal to 8 gigabytes of RAM, and have at least 50 gigabytes of free disk space available. As no hardware is being used in this project, the computer required to run the software is the only hardware involved.

4.2 System Detailed Design

The product consists largely of following the MagicGrid framework workflow to generate parts of our model in sequence. This section documents those steps, and the team's deviations from the framework. The team started by creating a new project using the MagicGrid Quickstart v2 project template.

4.2.1 Problem Domain

The role of the problem domain is to capture the stakeholder needs and understand the stakeholder needs in order to fully understand the problem at hand. All of the models contained in the problem domain section of MagicGrid deal with generating system requirements which will later drive the solution domain. These models include the black box and white box views. Fully defining the problem domain is equivalent to a system Preliminary Design Review (PDR).

4.2.1.1 Black Box

The role of the black box row of the MagicGrid framework is to understand the context in which the system operates and what actions the system needs to perform at a high-level. This is used to generate more stakeholder needs as more questions are asked about the problem domain, and drives requirements dealing with system behavior but not architecture.
The black box domain consists of the stakeholder needs, system context, use cases, and measures of effectiveness.

4.2.1.1.1 System Context

The system context describes an external view of the system and its interactions with external entities including their relationships. More than one system context can be defined for a system of interest.
This was first accomplished by reviewing the most complex system context that was identified: Crewed Station Operations. The team began by reviewing the mission of interest for the model and identifying key external entities that are necessary to the mission such as Ground Control, the Astronauts, Visiting Vehicles, etc. Inside of Magic System of Systems Architect the system is modeled using a Block element which is given the type of System.
The external systems were modeled as Part Properties with no name, and a type that corresponds to the system. External system types are modeled as Blocks that are under the general problem domain in the containment tree. Those blocks are used as the types for the Part Properties in the system context.
The team then identified the resources that flow between each external system and the system of interest. These are modeled in the modeling software by creating Connectors to and from each external system and the system of interest, and then creating a new Item Flow for each connector. The Item Flows contain resources that they provide by containing a Signal element of the type of resource. For example, the Visiting Vehicle external system has an Item Flow from itself to the Station. A Signal named “Crew” would be created and set as the data for the Item Flow.

4.2.1.1.2 Use Cases

The Use Case model describes the external view of the system as a user of the system would see it, and shows the core functionality of the system which can be interacted with.
Each use case model is tied to a particular system context because the external entities on the system context model are used as actors on the use case model. Therefore the “Crewed Station Operations” context was chosen to build our use case model off of. This was done by creating a new Use Case Diagram in the containment tree under “3 Use Cases” and dragging in the System Context block into the use case to provide the system boundary for the use case model. Then each of the Actors under “Users and Other Systems” in the system context model were dragged onto the Use Case model as external actors. A new Use Case was created by clicking the Use Case option in the Diagram Pallette. It was given a name, and then an Association was drawn between each Actor it interacts with and the Use Case.

Use Cases have associated scenarios which describe the actions that take place when that functionality is activated in the system. The tool provides an easy way to create this, and each Use Case was right-clicked and the “Create Diagram” option was selected, and the “SysML Activity Diagram” sub-option was selected. The first thing to do in a new Activity Diagram is create the swimlanes for the different actions that can take place. The “Vertical Swimlanes” item under the Diagram Palette was selected and the checkboxes checked for the different systems that will be taking part in the use case. Then actions were placed inside of the corresponding swimlane by clicking the “Action” option in the Diagram Palette and clicking the appropriate swimlane. These can then be named, and connected to other Actions using a Control Flow or Object Flow relationship. Control Flow relationships show that one Action leads to another, while Object Flows show the flow of data from one Action to another.

4.2.1.1.3 Measures Of Effectiveness

The purpose of the measures of effectiveness part of the model is to quantize nonfunctional stakeholder needs and system requirements in order to be able to verify if they have been met.
This is accomplished in the model by creating a new Measures of Effectiveness BDD under the “4 Measures of Effectiveness” folder in the containment tree. Then, an “MoEs Holder” Block is created alongside it in the containment tree and dragged into the BDD. Then the system Block is also dragged in, and the two are linked using a Generalization relationship from the system to the MoEs Holder. Each measure of effectiveness is represented in the model using a Value Property which is named for example “/powerProduction” to represent the amount of power that the system is producing. Then it is marked as a measure of effectiveness by right-clicking it, selecting the “Stereotype” option, and checking the “moe” stereotype. Then a type (units) can be set on the value by right-clicking, selecting “Specification” and clicking next to the “Type” heading.

4.2.1.2 White Box

The role of the white box row of the MagicGrid framework is to understand the internal components and behavior of components in the system.

4.2.2 Solution Domain

The solution domain builds off of the problem domain and the list of system requirements that were generated as the problem domain was modeled. The solution contains the design of the system without describing any implementation details.

4.3 Internal Communications Detailed Design

Within the MagicGrid Framework, components are interlinked between different views. This allows a user to ‘click-through’ to different models. From this, created models are directly used in the creation of later models (see section 2.3 for an overview of internal communication).

As of the time of writing, this is all of the observed interlinking between separate models:

- Captured stakeholder needs are refined into requirements
- Requirements are used to drive the flow of data in the system context
- External systems in the system context are used to develop actors in the use case model
- Use cases in the use case model are used as the basis for an activity diagram

5 VIEWPOINT DESCRIPTIONS

This section will provide a short overview of each model within the problem/solution domain as described in the MagicGrid Framework.

5.1 Stakeholder Needs

#	△ Name	Text	Documentation
1	SN-1 User Needs		
2	SN-1.1 Cislunar Crewed Mission	The SoI should facilitate human crewed missions to cislunar space including capabilities that enable	Add sources in this field
3	SN-1.2 Exploration Science Mission	The SoI should provide capabilities to meet scientific requirements for lunar discovery and exploration.	Add sources in this field
4	SN-1.3 Forward Compatablility	The SoI should enable, demonstrate, and prove technologies that are enabling for deep space missions.	Add sources in this field
5	SN-1.4 Manual Flight Control	The SoI should allow for manual control of flight dynamics.	Add sources in this field
6	SN-1.5 Automatic Flight Control	The SoI should be able to maintain its orbit.	Add sources in this field
7	SN-1.6 Independent Power	The SoI should produce, store, and regulate its own power.	Add sources in this field
8	SN-1.7 Crew Safety	The SoI should keep the crew alive and safe.	Add sources in this field
9	SN-1.8 Crew Mission Extensability	The SoI should accomodate extended crew mission durations.	Add sources in this field
10	SN-1.9 Extra-Vehicular Activity	The SoI should allow crew to perform extra-vehicular activity.	Add sources in this field
11	SN-1.10 Visiting Vehicle Docking	The SoI should allow for Visiting Vehicles to dock.	Add sources in this field
12	SN-1.11 Vehicular Logistical Tranfer	The SoI should accept the transferring of crew and cargo.	Add sources in this field
13	SN-1.12 Lunar Surface Communication	The SoI should provide communication to the Lunar surface.	Add sources in this field
14	SN-2 Design Constraints		
15	SN-2.1 User Interoprability	The SoI should have to ability to support multiple self, commerical, and international partner objectives.	Add sources in this field
16	SN-2.2 Crew Size	The SoI should accomodate up to 4 crew members.	Add sources in this field
17	SN-2.3 Mission Duration	The SoI should enable 30 to 90 days of crew missions.	Add sources in this field
18	SN-2.4 Orbtal Access	The SoI should be easy to access from Earth with current launch vehicles.	Add sources in this field
19	SN-2.5 Earth Communication	The SoI should have continuous communication with Earth	Add sources in this field

Figure 5: StakeHolder Needs

Figure 3 describes the stakeholder needs captured by MagicGrid. It's important to note that these are not requirements but what the stakeholder wants the system to have. These needs are later shaped into requirements as the team progresses with other models and diagrams.

5.2 System Context

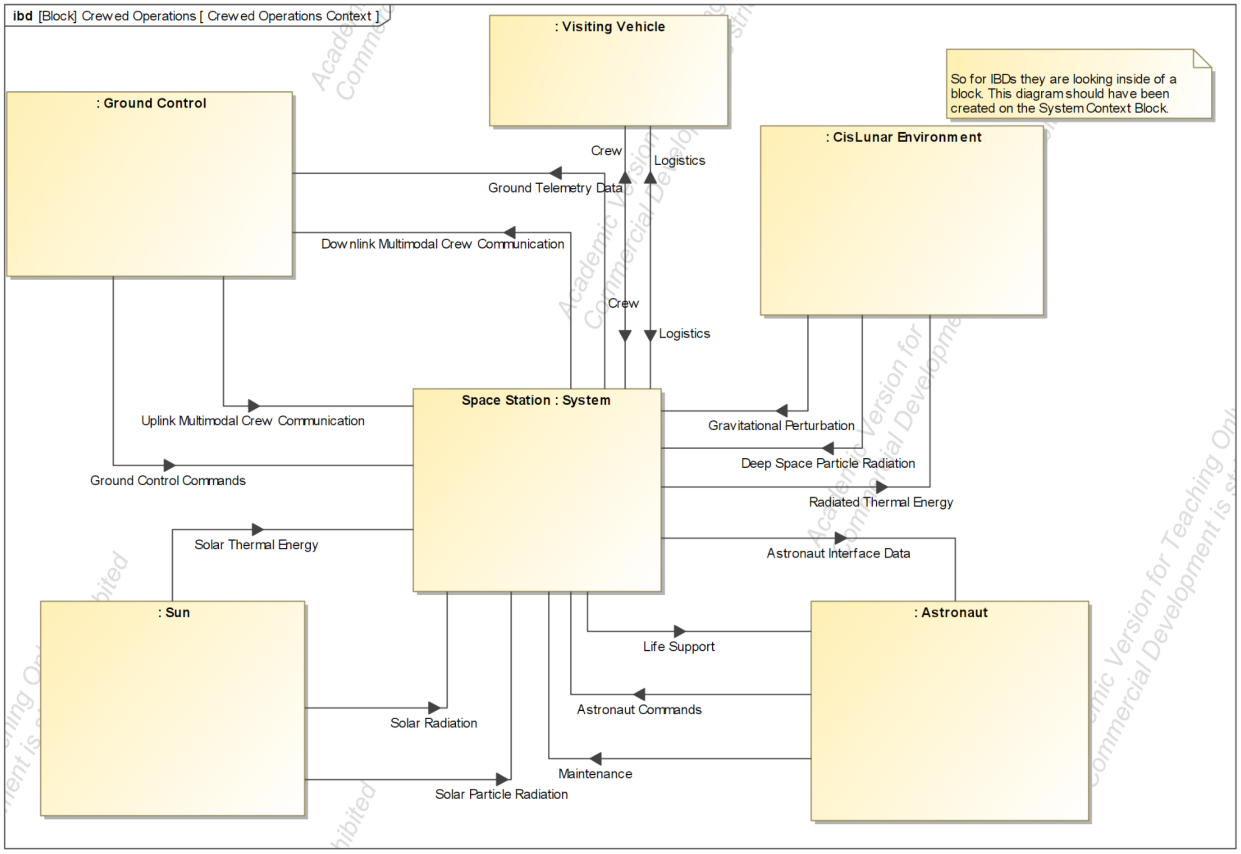

Figure 6: System Context

The System Context shown in Figure 4 attempts to capture the external view of the system. This model introduces all external systems that do not belong into the system of interest. The diagram also gives a view on what kind of data these external systems exchange with the SoI.

5.3 Use Cases

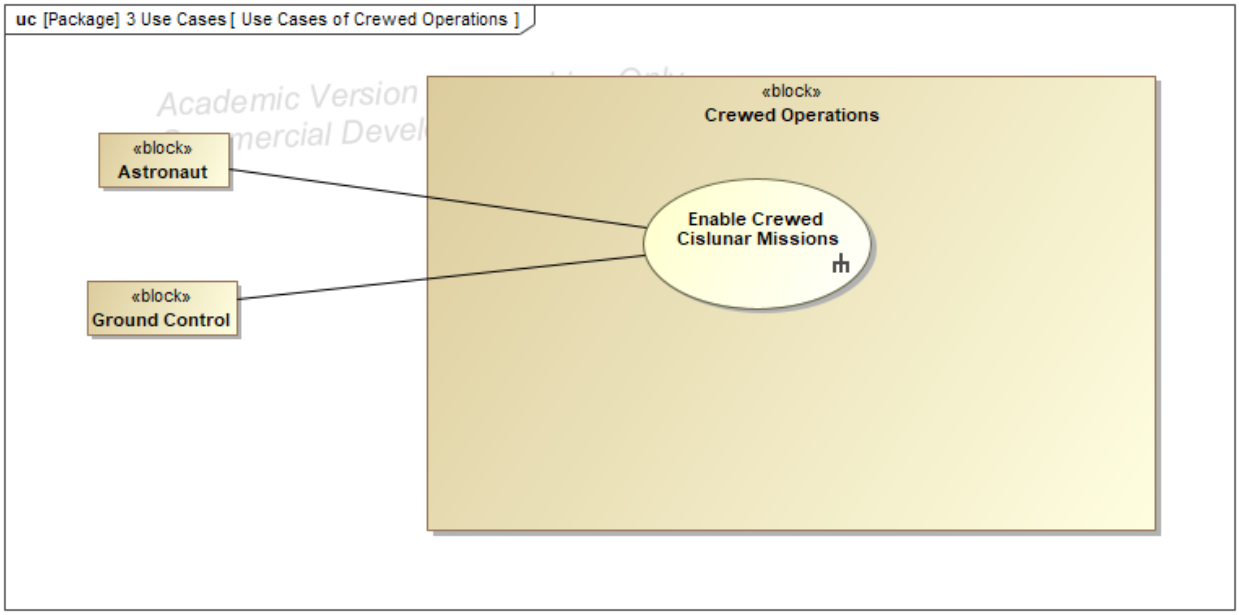

Figure 7: Use Cases

Figure 5 describes the Use Case model of the SoI. The Use Case diagram captures and refine the functional stakeholder needs. The use cases are more precise in telling what the stakeholder expects out of the system.

5.3.1 Use Case Scenarios

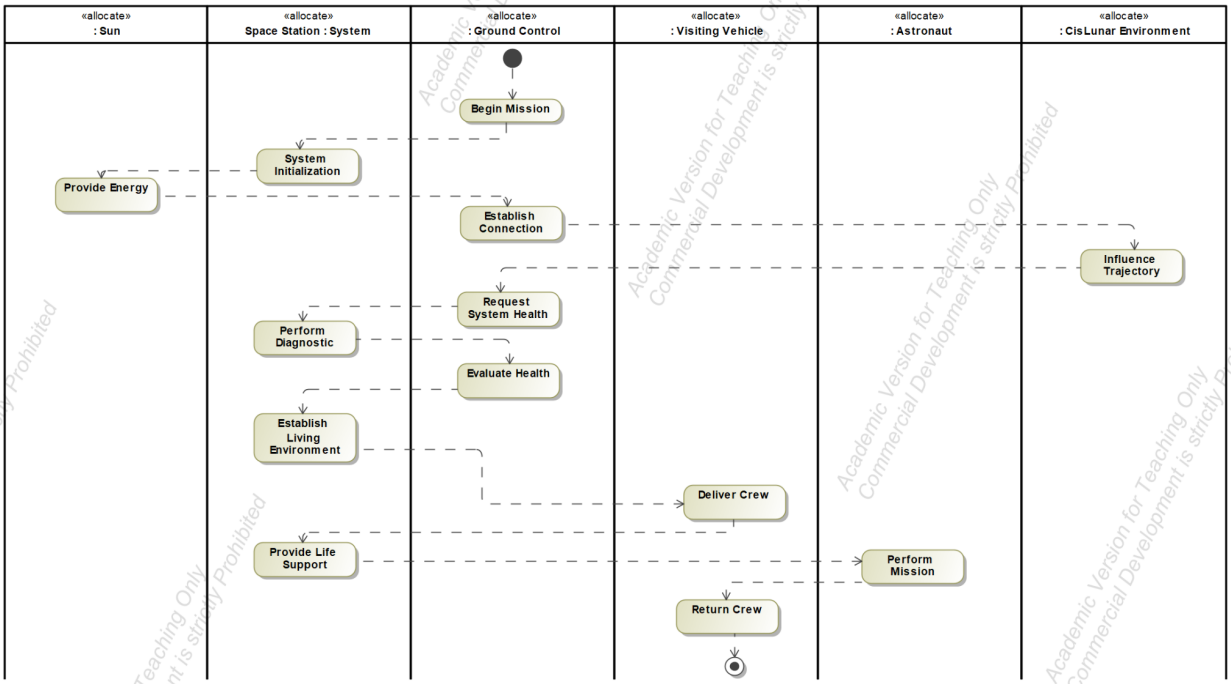

Figure 8: Use Case Scenarios

Use Case Scenarios are needed to give a story to the use cases mentioned in Figure 5. They give a more detailed view on how the system communicates with an actor/s.

5.4 Measures Of Effectiveness

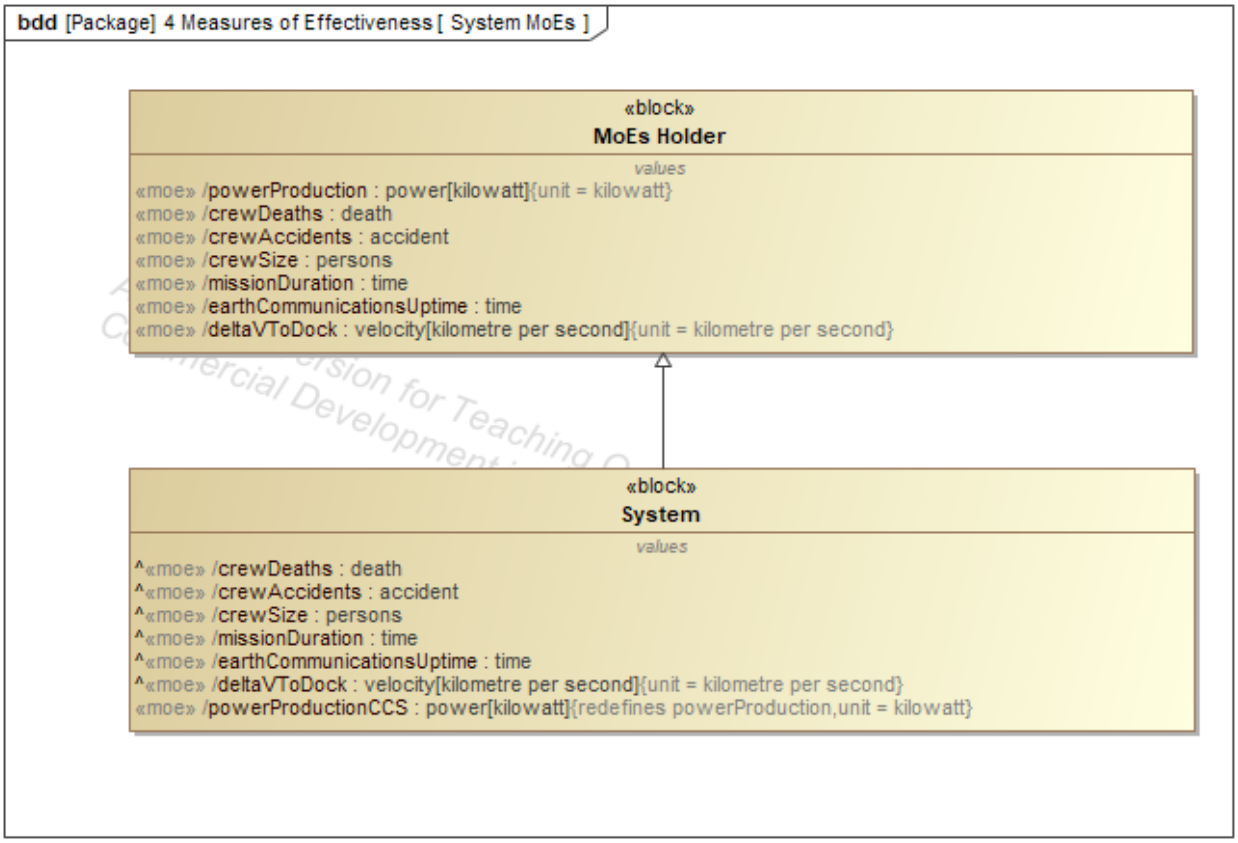

Figure 9: System Measures of Effectiveness

Figure 9 shows the measures of effectiveness for the entire system. Measures of effectiveness allow us to capture ways to verify nonfunctional requirements by deriving tangible numerical values that can determine if our system is successfully fulfilling its requirements.

6 SYSTEM INTEGRITY CONTROLS

The data within the MagicDraw tool is categorized as confidential as the licenses needed to use the software were paid for. The licenses are also educational-only, and may not be used for commercial purposes. The MagicDraw tool has a watermark that specifically shows that the software version used is for learning only. The licenses are also tied/linked by hardware to the MAC address of the host's computer. Making it close to impossible to spoof or use in any other device.

The team uses Git and GitHub for version control. GitHub allows to have a more secure version control and make sure the team can share their work without unauthorized outside source tempering with the source files.

In terms of communication, the team also has various data and files being transferred to outside sources other than the team(advisor to check the MagicGrid framework). The team uses their organization's email to communicate using the communication software Teams.