
System Design Document

for

MBSE Avionics System Capstone

Version 1.0.0

Prepared by
Walter Hernandez
Shawn Merrick Miller
Luke Newcomb
William Pate

Embry-Riddle Aeronautical University

April 14, 2024

TABLE OF CONTENTS

| | |
|--|-----------|
| 1 INTRODUCTION | 3 |
| 1.1 Purpose and Scope | 3 |
| 1.2 Project Executive Summary | 3 |
| 1.2.1 System Overview | 4 |
| 1.2.2 Design Constraints | 5 |
| 1.2.3 Future Contingencies | 5 |
| 1.3 Document Organization | 5 |
| 1.4 Project References | 5 |
| 1.5 Glossary | 5 |
| 2 SYSTEM ARCHITECTURE | 5 |
| 2.1 System Hardware Architecture | 5 |
| 2.2 System Software Architecture | 6 |
| 2.3 Internal Communications Architecture | 7 |
| 3 HUMAN-MACHINE INTERFACE | 8 |
| 4 DETAILED DESIGN | 9 |
| 4.1 Hardware Detailed Design | 9 |
| 4.2 System Detailed Design | 9 |
| 4.2.1 Problem Domain | 9 |
| 4.2.1.1 Black Box | 10 |
| 4.2.1.2 White Box | 15 |
| 4.2.2 Solution Domain | 15 |
| 4.3 Internal Communications Detailed Design | 15 |
| 5 VIEWPOINT DESCRIPTIONS | 15 |
| 5.1 Stakeholder Needs | 16 |
| 5.2 System Context | 17 |
| 5.3 Use Cases | 18 |
| 5.4 Measures Of Effectiveness | 20 |
| 5.4 Functional Analysis | 21 |
| 5.5 Conceptual Subsystems | 26 |
| 5.6 Failure Mode & Effects Analysis | 26 |
| 6 System Integrity Controls | 29 |

REVISION HISTORY

| Name | Date | Reason For Changes | Version |
|----------------------|-------------|---|----------------|
| William Pate | 9/28/23 | n/a | 0.1.0 |
| Walter Hernandez | 9/28/23 | n/a | 0.1.0 |
| Luke Newcomb | 9/29/23 | n/a | 0.1.1 |
| Shawn Merrick Miller | 10/31/23 | n/a | 0.2.0 |
| William Pate | 10/31/23 | n/a | 0.2.1 |
| Walter Hernandez | 10/31/23 | n/a | 0.2.1 |
| Luke Newcomb | 10/31/23 | n/a | 0.3.0 |
| Luke Newcomb | 11/7/23 | n/a | 0.3.1 |
| William Pate | 11/14/23 | n/a | 0.3.2 |
| Shawn Merrick Miller | 11/14/23 | n/a | 0.3.2 |
| William Pate | 11/20/23 | n/a | 0.3.4 |
| Walter Hernandez | 11/21/23 | n/a | 0.4.0 |
| Luke Newcomb | 11/21/23 | n/a | 0.4.0 |
| Shawn Merrick Miller | 2/3/24 | n/a | 0.5.1 |
| Luke Newcomb | 2/4/24 | n/a | 0.5.2 |
| Walter Hernandez | 2/4/24 | n/a | 0.5.3 |
| William Pate | 2/4/24 | n/a | 0.5.4 |
| Luke Newcomb | 3/2/24 | Reformatted title page and revision history table | 0.6.0 |
| Shawn Merrick Miller | 3/2/24 | Applied most of the feedback from the previous version of this document | 0.6.1 |
| Luke Newcomb | 3/2/24 | Added figures to Section 4's System Context section to show every step of the process of creation. | 0.6.2 |
| Luke Newcomb | 3/2/24 | Added project references and updated glossary with definitions of terms | 0.6.2 |
| Luke Newcomb | 3/2/24 | Rewrote all existing subsections in Section 4 to reflect the current version of our process | 0.6.3 |
| Walter Hernandez | 3/3/24 | Created a new sub section within Section 4 that explains Failure Mode and Effect Analysis. | 0.6.4 |
| Walter Hernandez | 3/3/24 | Added the new FMEA subsection within Section 5. This is in order to show a bit more clarity on how the team goes about generating FMEA items. | 0.6.5 |
| William Pate | 3/3/24 | Created Functional Analysis subsections within Section 4 and 5 | 0.6.6 |
| William Pate | 3/3/24 | Created Conceptual Subsystems subsections within Section 4 and 5 | 0.6.7 |
| Walter Hernandez | 04/14/24 | Finalized the FMEA description in section 5 with new fresh images and some new ones. | 0.6.8 |
| Luke Newcomb | 04/14/24 | Final document checks and format fixes | 0.6.9 |
| Shawn Merrick Miller | 04/14/24 | Final document checks | 1.0.0 |

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

The MASC project is contained in the model of the notional CCS described above. This model is a sophisticated set of files and diagrams which a user can perform design actions with regarding the CCS design. The MASC CCS Model is controlled using the MagicDraw™ software tool, and can be modified asynchronously by multiple users. The MagicDraw™ software is the mechanism by which the user can organize the CCS model, and populate the CCS design data sets. The CCS model is also constrained by the MagicGrid™ framework, and can accept input from external stakeholders. Lastly, the MASC project involves Failure Mode Effect Analysis on the CCS model, which can be performed using the MagicDraw™ FMEA Plugin. This is represented below in Figure 1, wherein the MASC Use Case Diagram is shown.

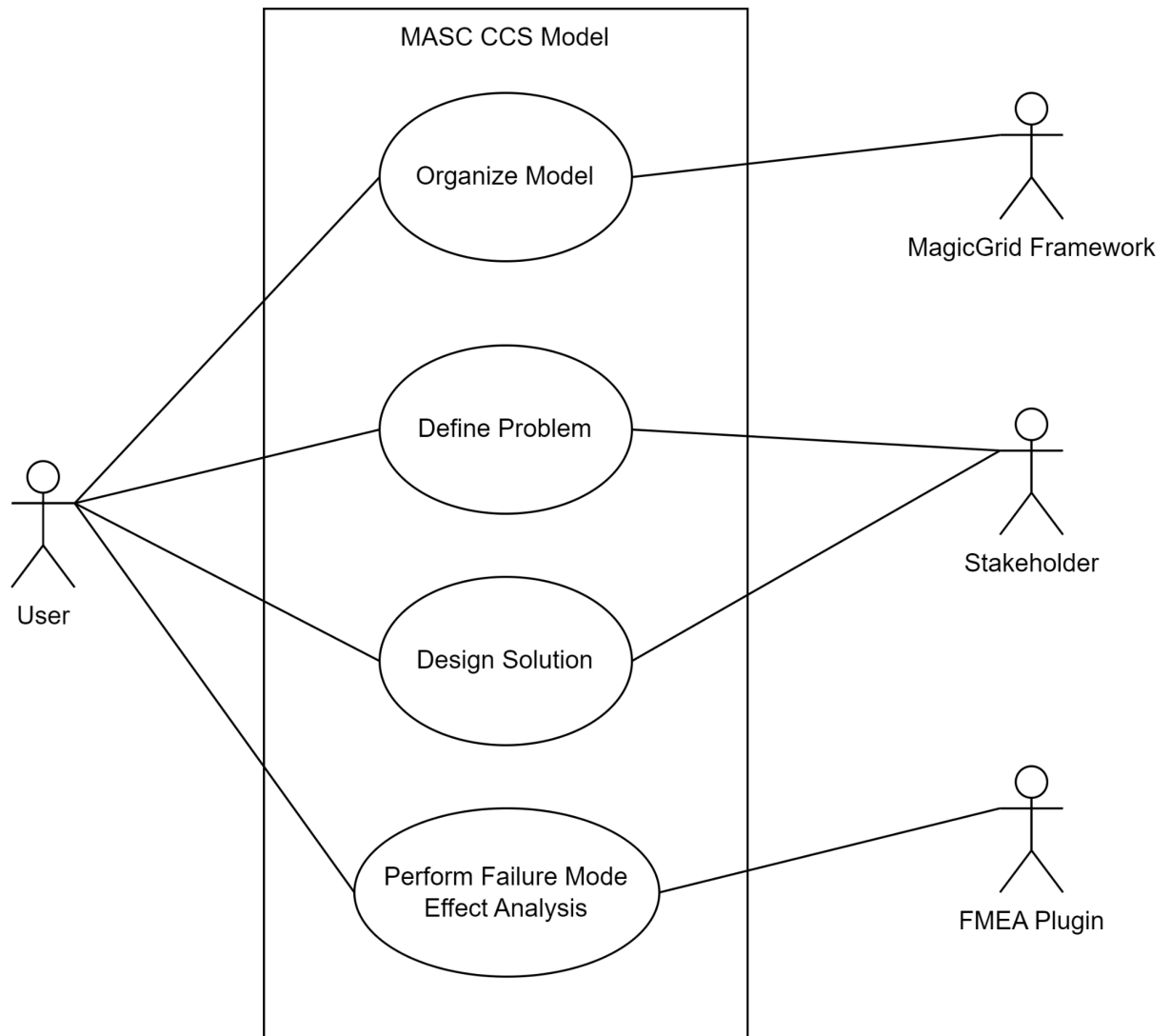


Figure 1: MASC Generic Avionics Use Case Model

The use case diagram in Figure 1 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

- [1] MagicGrid Book of Knowledge, 2nd Edition, from
<https://discover.3ds.com/magicgrid-book-of-knowledge>
- [2] Describing FMEA Items Guide, from
<https://docs.nomagic.com/display/CSRA190/Describing+FMEA+Items>

1.5 GLOSSARY

CCS - Crewed Cislunar Station. The system that is being modeled by using the process explored in this project. A space station that orbits the moon which can support a crew.

MBSE - Model-Based Systems Engineering. The current alternative to document-based systems engineering.

FMEA - Failure Mode and Effects Analysis. The process of finding and characterizing various ways that the system could fail in order to drive the creation of new safety and reliability requirements for the system.

SOI - System of Interest. The system that is being modeled by MBSE.

BDD - Block Definition Diagram. A diagram which defines how a block interacts with other blocks in an external manner.

IBD - Internal Block Diagram. A diagram which describes processes or components which are internal to a block.

RAM - Random Access Memory

2 SYSTEM ARCHITECTURE

In the system architecture section, the MASC system and the methodology for modeling the CCS will be discussed. Furthermore this section will discuss the MASC architecture involving the MagicGrid™ framework.

2.1 SYSTEM HARDWARE ARCHITECTURE

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

2.2 SYSTEM SOFTWARE ARCHITECTURE

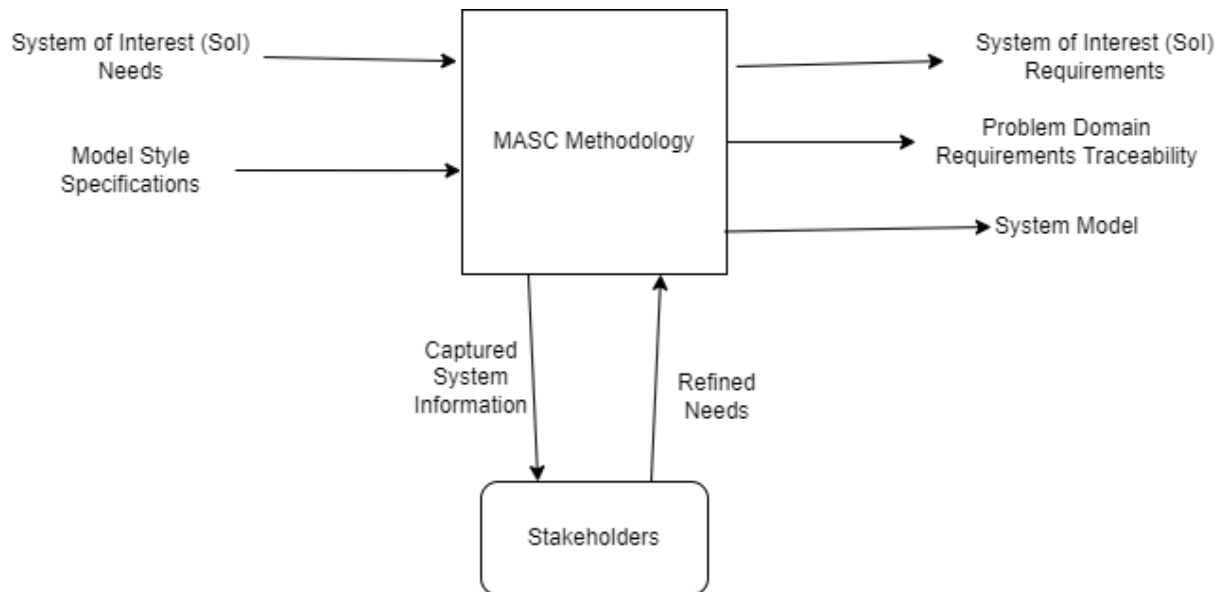


Figure 2: MASC System Architecture

The system architecture presented in Figure 2, 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

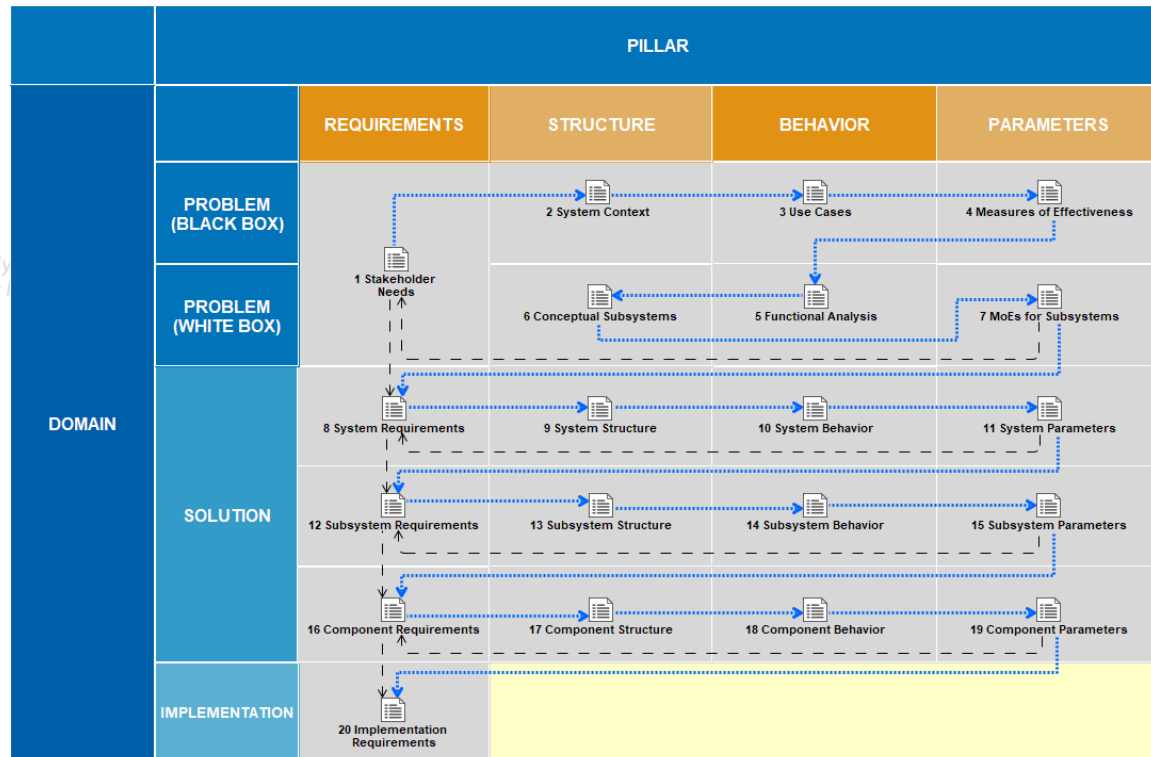


Figure 3: Internal Model Flow

Within the MagicGrid Framework, the internal model flow, presented in Figure 3, 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

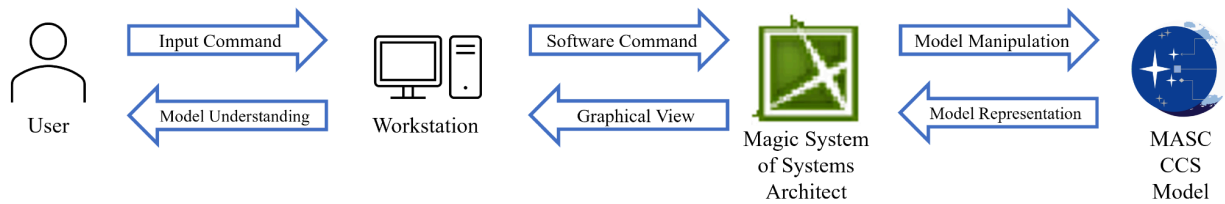


Figure 4: MASC Human-Machine Interface Architecture

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.

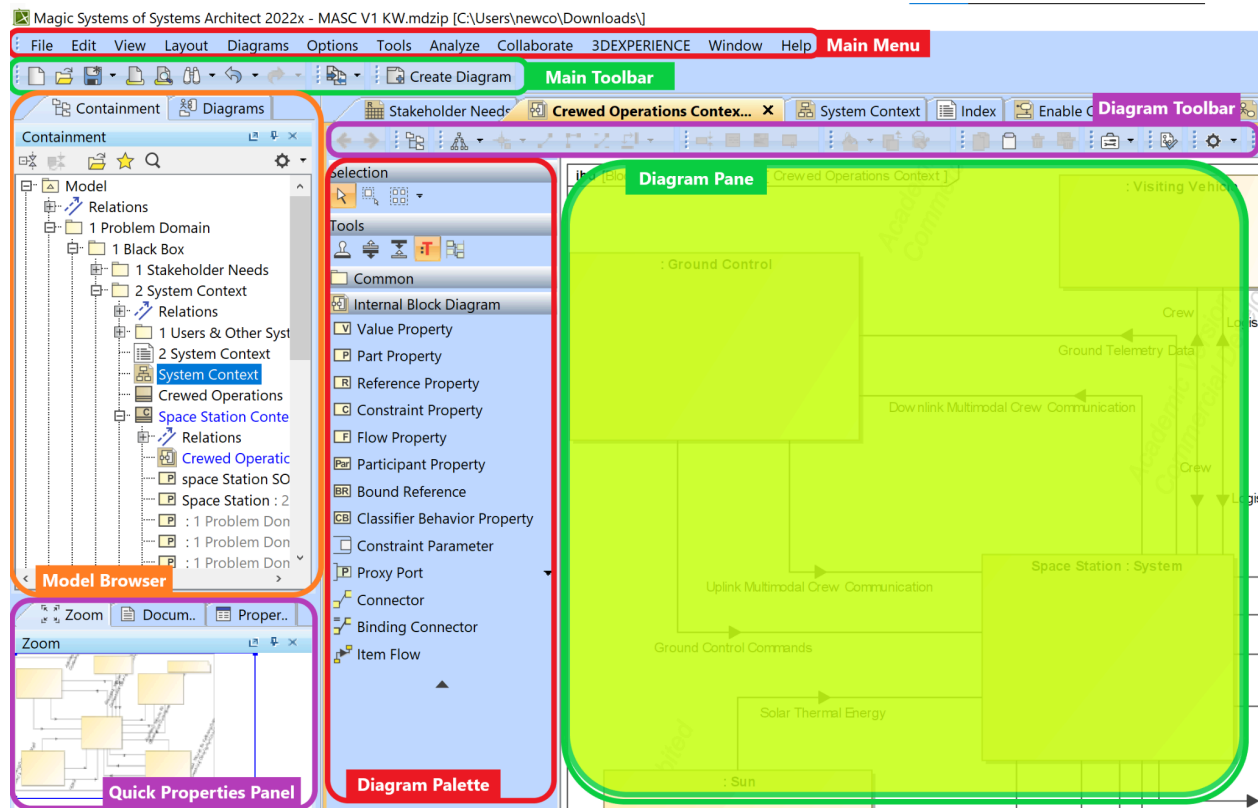


Figure 5: Magic System of Systems Architect User Interface

The components of the user interface for the Magic System of Systems Architect application are shown in Figure 5. The different parts of the user interface, such as the containment tree and diagram pane, allow the user to view and interact with the model in a number of ways.

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 v2 Blank project template as shown in Figure 6 below.

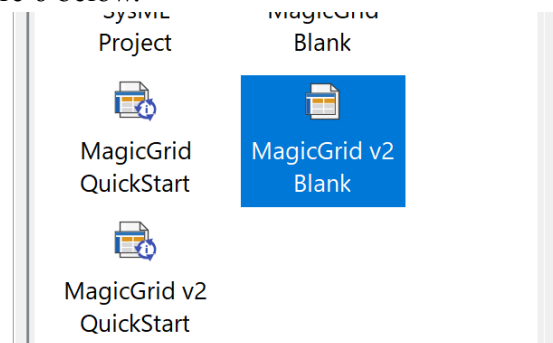


Figure 6: MagicGrid v2 Blank Model Menu Item

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 would 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). The problem domain is represented with MagicGrid as shown in Figure 7.

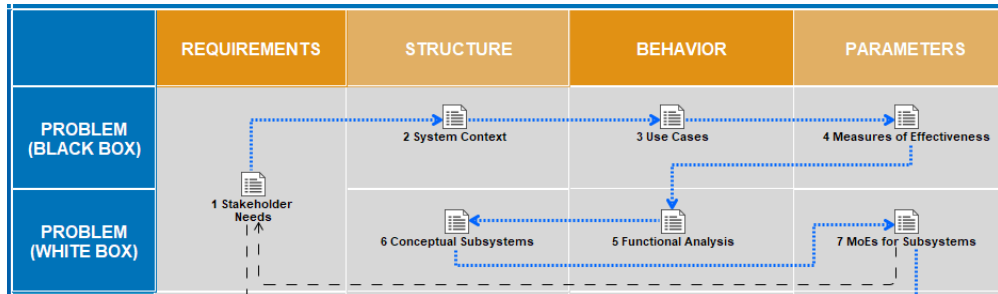


Figure 7: MagicGrid Problem Domain

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.

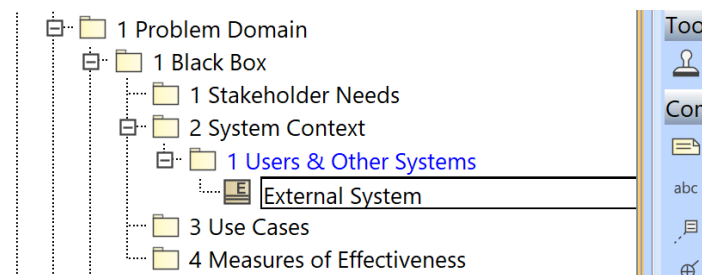


Figure 8: Creating an External System

Figure 8 shows how and where an external system is created as an External item, indicated by the block icon with an E next to the name. These are placed under the Users & Other Systems package. A new diagram is created by right-clicking the System Context package and selecting Create Diagram and creating one of type SysML Block Definition Diagram, as shown in Figure 9.

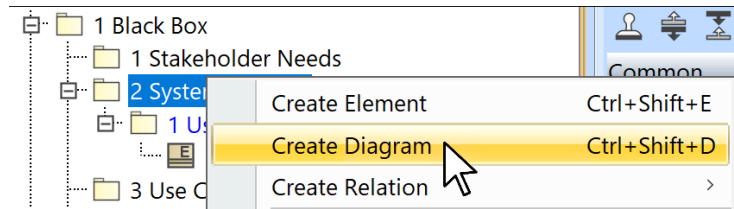


Figure 9: Creating a System Context Diagram

The external system blocks are dragged along with the system itself from the containment tree into the diagram and connected using directed aggregation relationships. This is done by selecting the directed aggregation option from the diagram palette, clicking and holding on the system and releasing the click on the external system, as shown in Figure 10 and Figure 11.

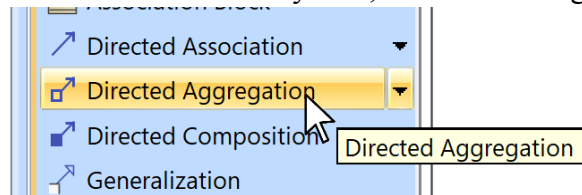


Figure 10: Directed Aggregation Option From The Diagram Palette

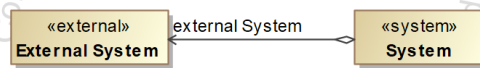


Figure 11: Resultant Directed Aggregation Relationship

The team then identified the resources that flow between each external system and the system of interest. An IBD is also created to show the resources that flow from external systems to the SoI. The blocks for the external systems are used as the types for the Part Properties in the system context. A new Block which will represent the system context itself and an IBD on that Block is created under the same package as the previous BDD. These steps are shown in Figure 12, Figure 13, and Figure 14.

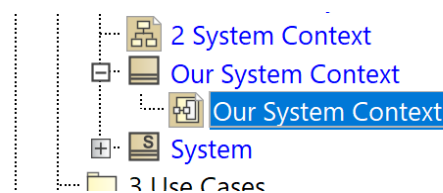


Figure 12: Block and IBD Creation

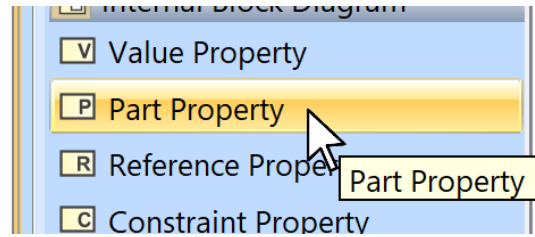


Figure 13: Part Property Diagram Palette Item

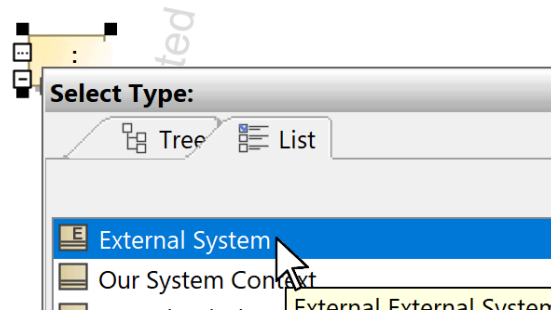


Figure 14: Part Property Typing Using The External System Block

Resource flow is 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. The Connector is found in the Diagram Palette. Item Flows can be created when the created Signal is dragged from the Containment Tree onto the Connector. Creating a new Item Flow will open a creation window where it is necessary to check that the direction of the Item Flow is correct, either From External System to System or From System to External System, and then the “Finish” button is clicked. This process is shown in Figure 15, Figure 16, Figure 17, Figure 18.

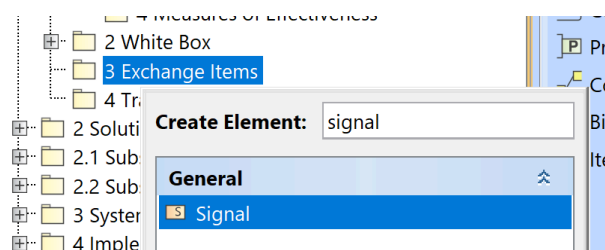


Figure 15: Creating A Signal Under Exchange Items



Figure 16: Using A Connector

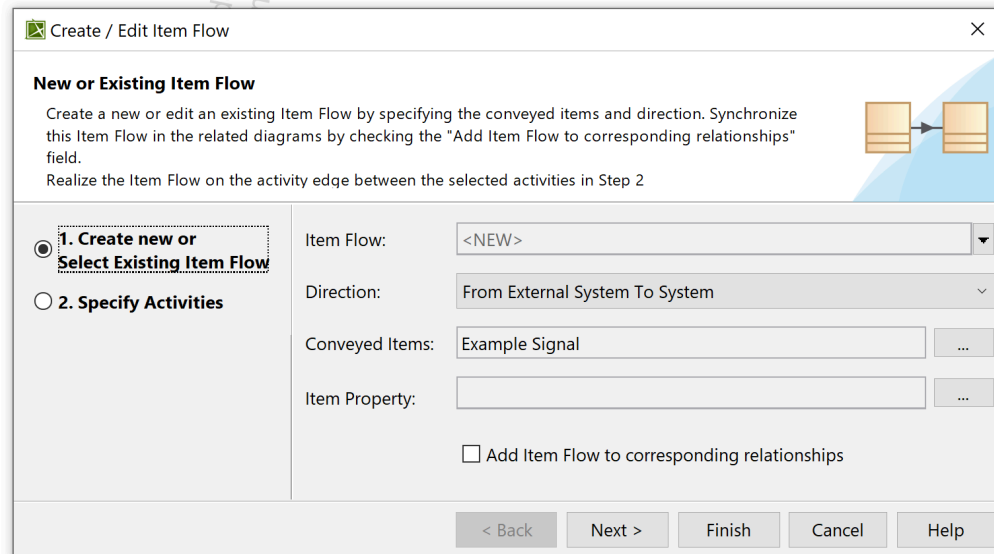


Figure 17: Item Flow Creation Window

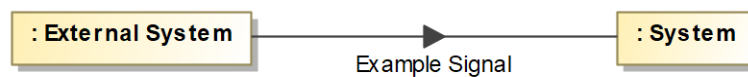


Figure 18: Finished Item Flow

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. Many of the procedures for generating diagrams and interacting with model elements are consistent with the techniques presented in section 4.2.1.1.1, however the specific traits which pertain to the Use Case model are mentioned below. 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 Palette. 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.

4.2.1.1.4 FUNCTIONAL ANALYSIS

The Functional Analysis builds off of the Use Cases in order to describe the logical sequence of actions between different actors in the system. Starting with a specific Use Case, you can right-click it and select “Create Diagram” to create a new functional analysis diagram. Once created, you can select the “Vertical Swimlanes” option from the toolbar and there will be separate ‘lanes’ for each actor. Next, “Actions” can be created or brought in from other diagrams. These actions should be linked together to demonstrate the ‘flow’ or transfer of data between your actors during the initial Use Case. The functional analysis should have both a start and finish state, and can optionally include ‘ports’ where the information is sent or received. As well as horizontal and vertical “Forks” and “Joins” to better illustrate the flow and direction of data in the system. Some use cases don’t involve every actor, by right-clicking the lanes there is an option to add or remove actors from the diagram. The Functional Analysis can be ‘broken down’ into further detail by right-clicking on an action and creating a new Functional Analysis diagram within the parent.

4.2.1.1.5 CONCEPTUAL SUBSYSTEMS

The Conceptual Subsystem diagram allows the system to be detailed into a layer further than shown in the System Context. From this, further requirements and diagrams can be derived from the subsystem, depending on the required specificity of the model. The main system block acts as the central connection from which all the subsystems are derived. The system shows each of its actors as shown in the System Context. Each subsystem can be created using a “Block” item and should have its relationship with the main block defined through a relation.

4.2.1.1.6 FAILURE MODE AND EFFECTS ANALYSIS

The purpose of Failure Mode and Effects Analysis (FMEA) is to provide a structured approach to identify and address potential failures within a system and its components. Usually initiated with stakeholder needs, FMEA is most effective when a Conceptual Subsystem is in place. This helps in understanding the relationships between subsystems and the potential for a failure to

impact multiple components or the entire system. FMEA focuses on analyzing how the system will respond to failures and the consequent effects, rather than devising solutions for these failures. The MagicDraw tool does not currently support any FMEA creation in its base version; a specific plugin, *Cameo Safety and Reliability Analyzer*, is required to integrate this methodology. This plugin introduces several new elements to facilitate the creation of FMEA-related items, including the FMEA Table. The FMEA Table is central to this process, serving as the primary diagram for documenting system failures. Within this table, failures are detailed as FMEA Items, each characterized by various attributes and fields. These include the Local Effect of Failure and the Cause of Failure, among others. These attributes are crucial for mapping the relationships between system components, their potential failures, and the severity of these failures.

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

Figure 19: StakeHolder Needs

Figure 19 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

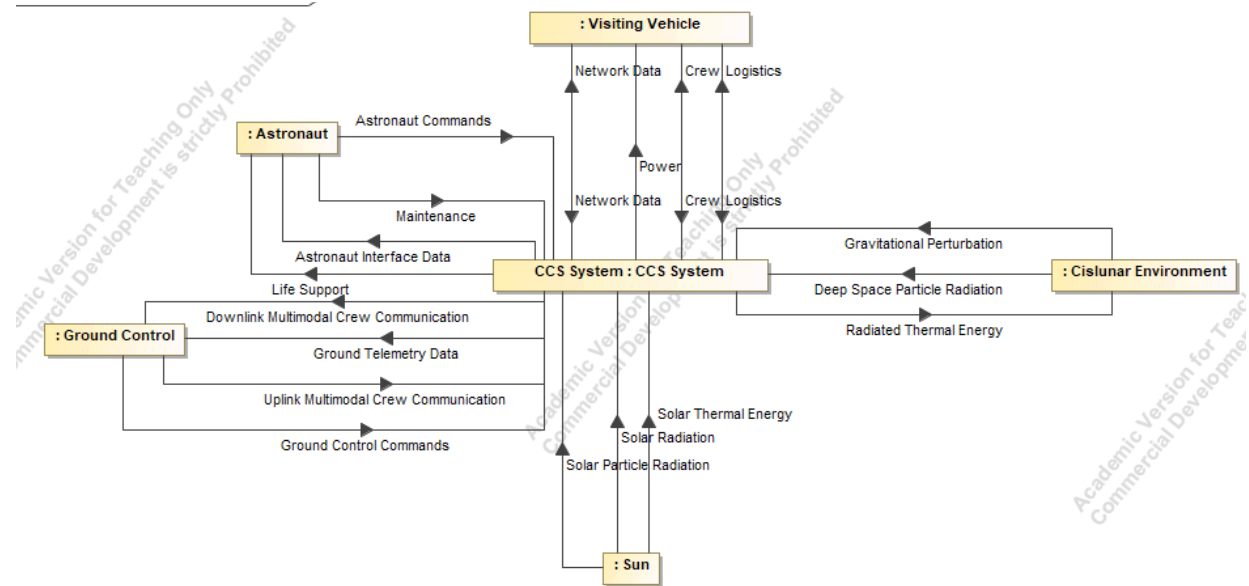


Figure 20: System Context

The System Context shown in Figure 20 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

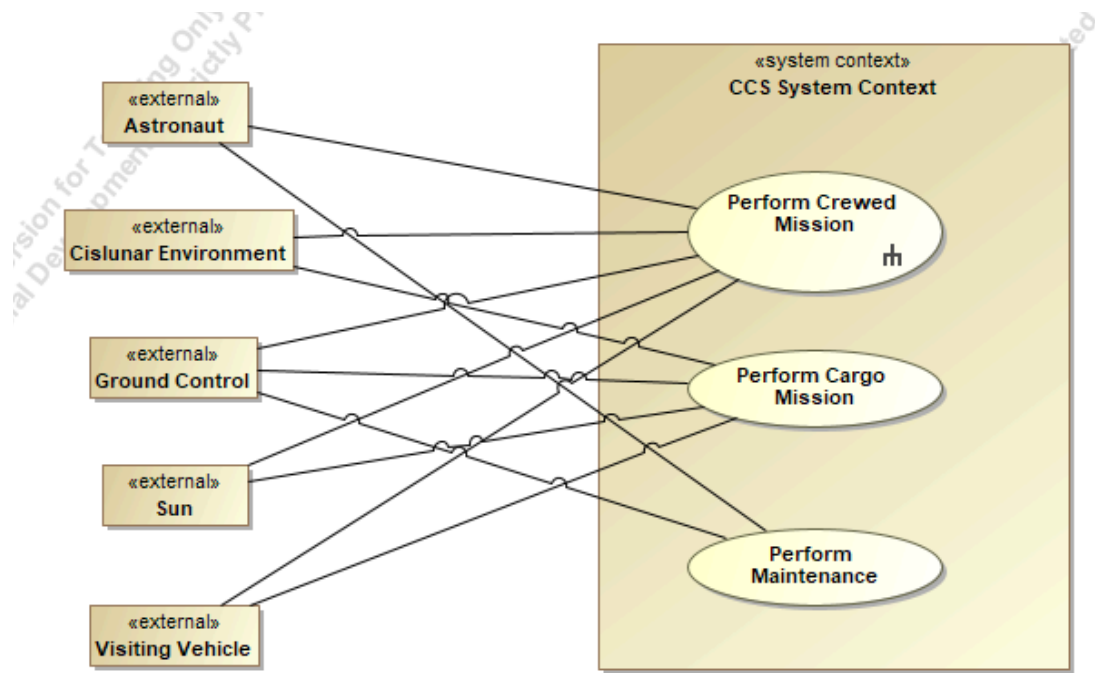


Figure 21: Use Cases

Figure 21 describes the Use Case model of the SoI. The Use Case diagram captures and refines 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

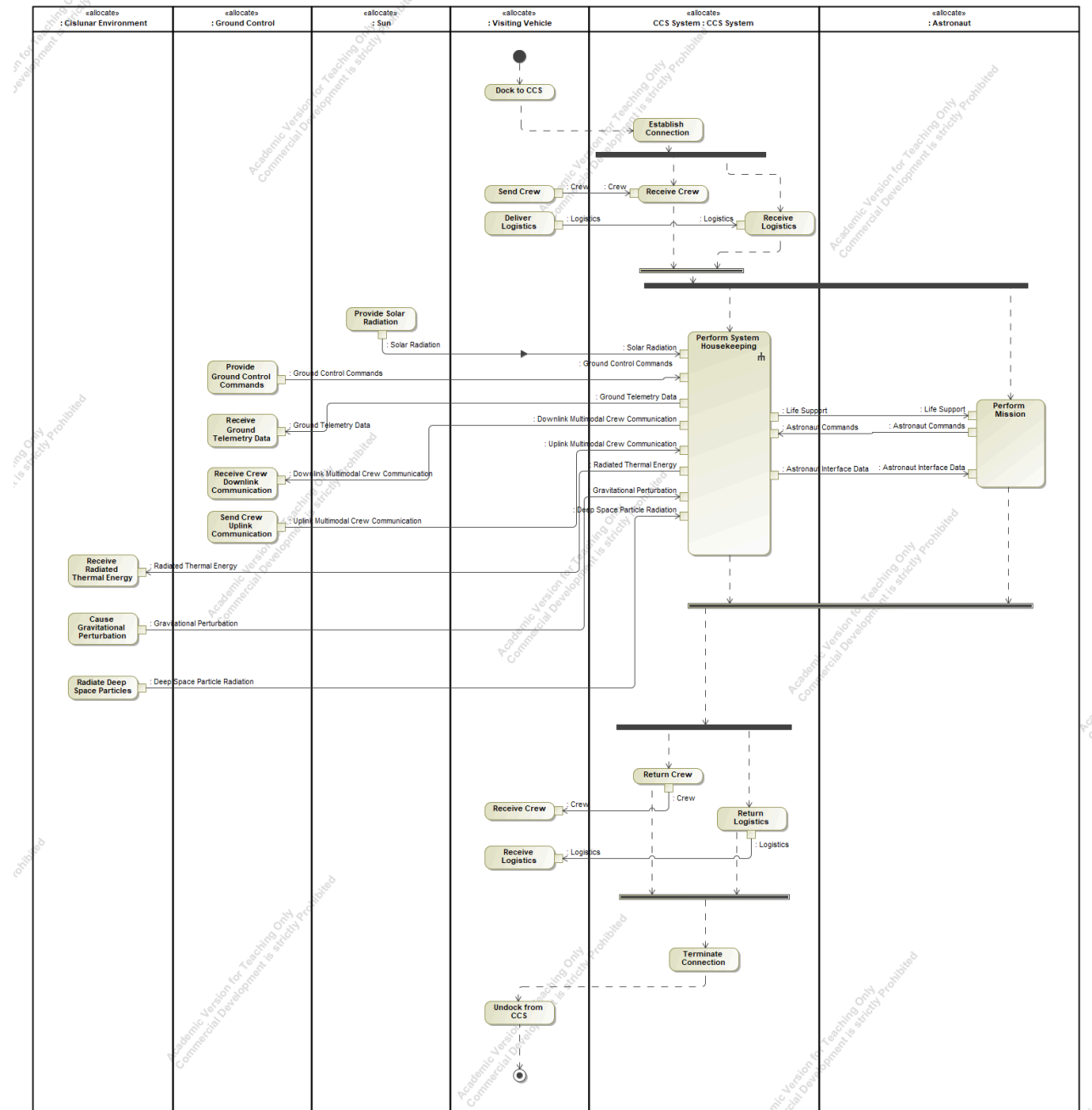


Figure 22: Use Case Scenarios

Figure 22 shows a Use Case Scenario of the Perform Crewed Mission Use Case. Use Case Scenarios are needed to give a story to the use cases mentioned in Figure 21. They give a more detailed view on how the system communicates with an actor/s.

5.4 MEASURES OF EFFECTIVENESS

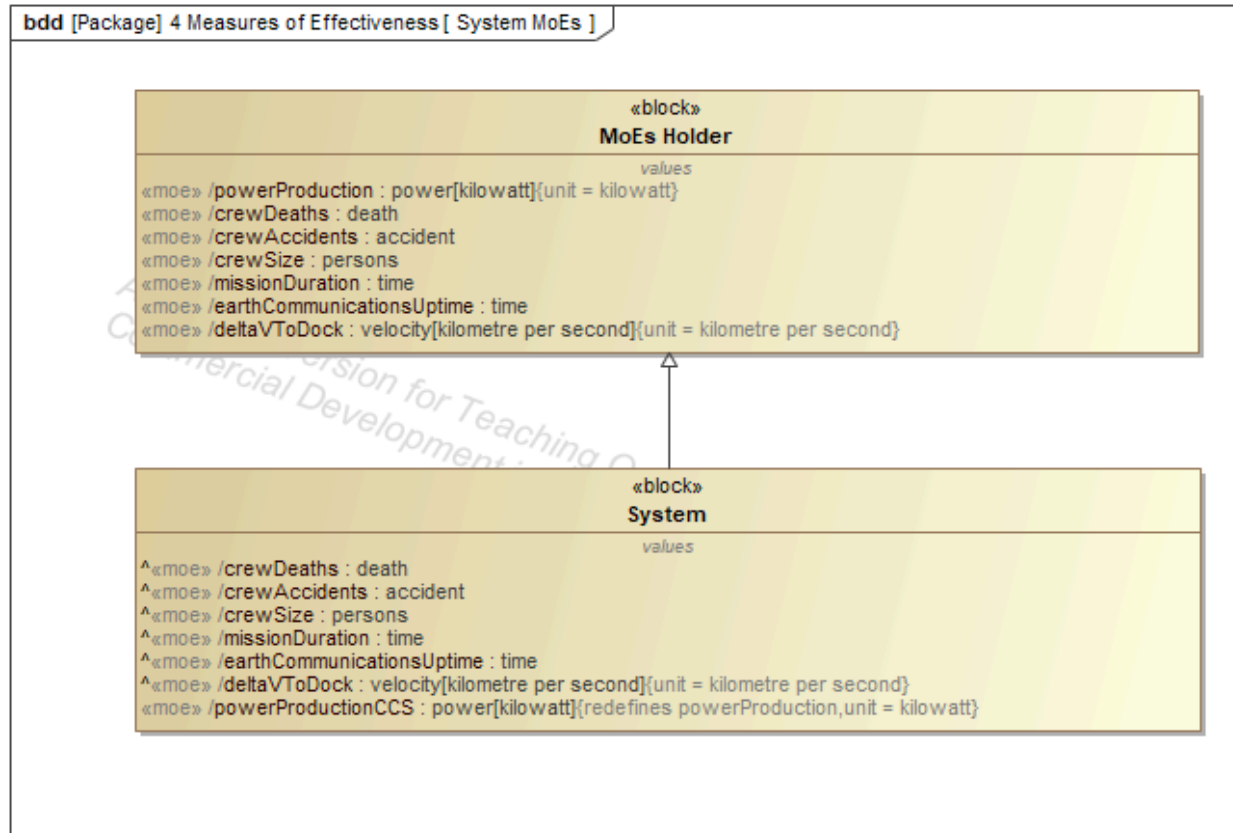


Figure 23: System Measures of Effectiveness

Figure 23 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.

5.4 FUNCTIONAL ANALYSIS

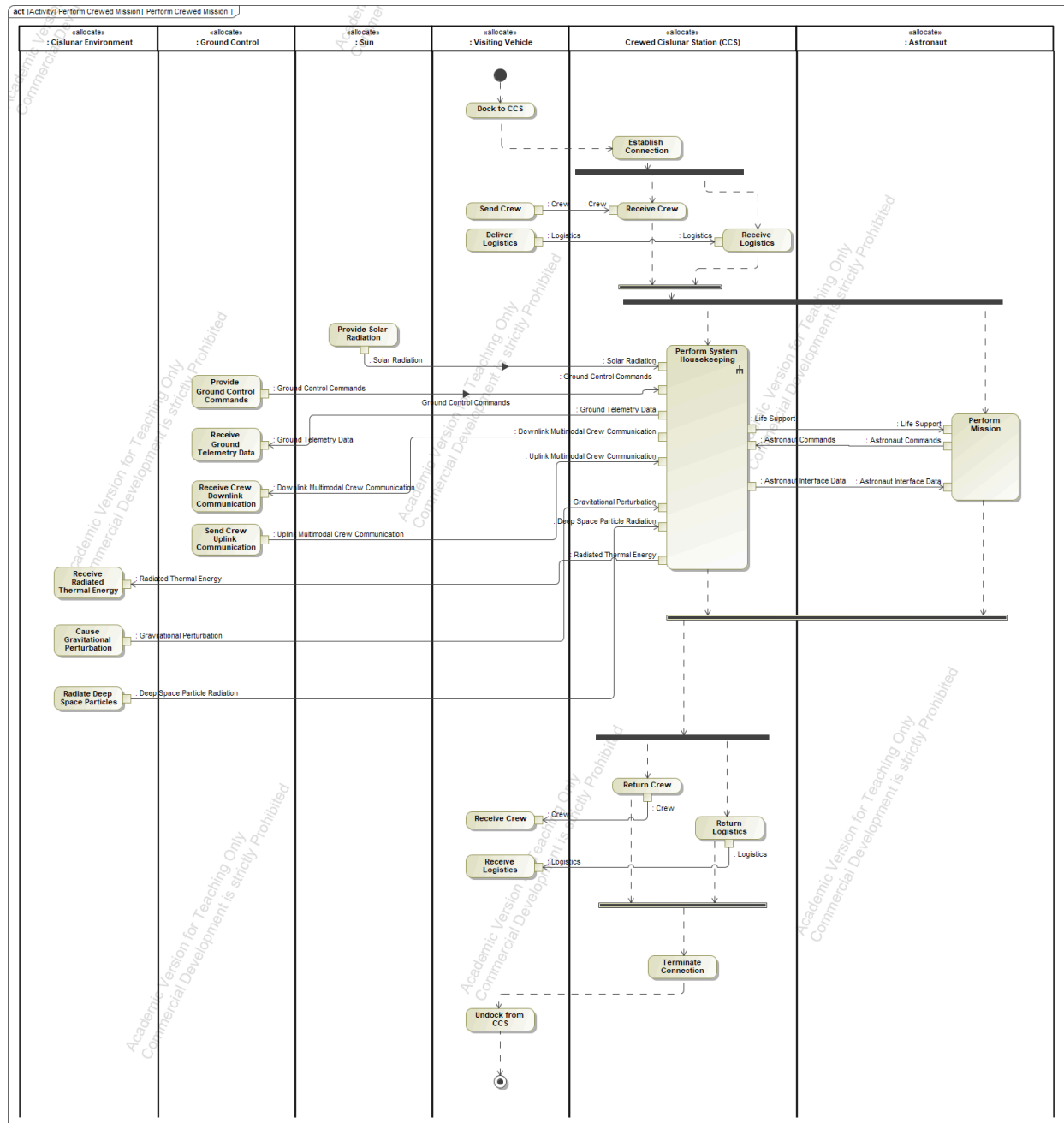


Figure 24: “Perform Crewed Mission” Functional Analysis

Figure 24 shows the Functional Analysis diagram for the Perform Crewed Mission Use Case. This describes the flow of actions and data between actors. Initially, the visiting vehicle docks to the CCS, which begins the rest of the Use Case. Then the system handles the interaction, accounting for input from the Sun, Ground Control, and the Cislunar Environment. The majority of system functions in this Use Case are collectively described as “Performed System Housekeeping”. This action is further detailed in the next level Functional Analysis diagram, as shown below.

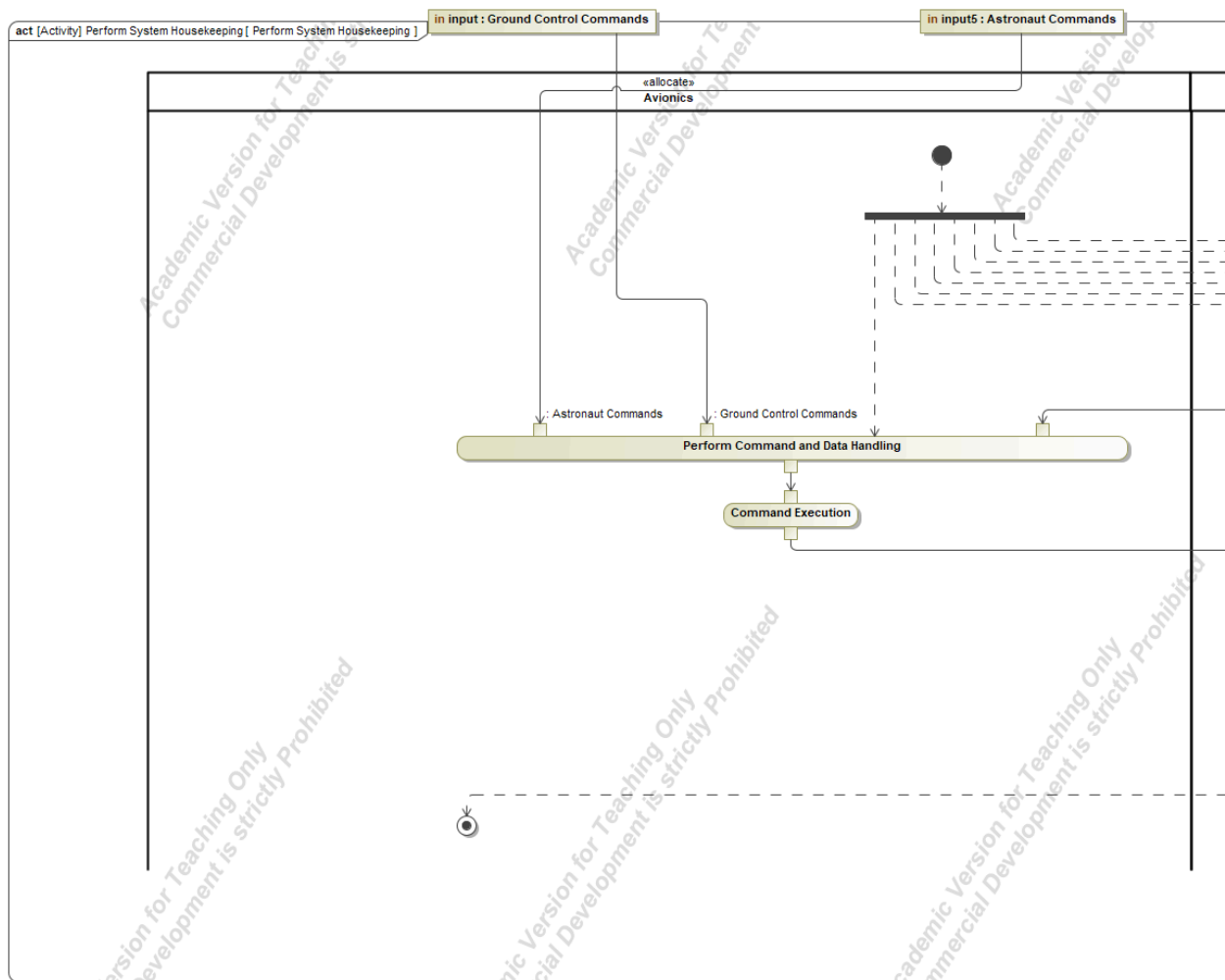


Figure 25-a: “Perform System Housekeeping” Functional Analysis (Avionics)

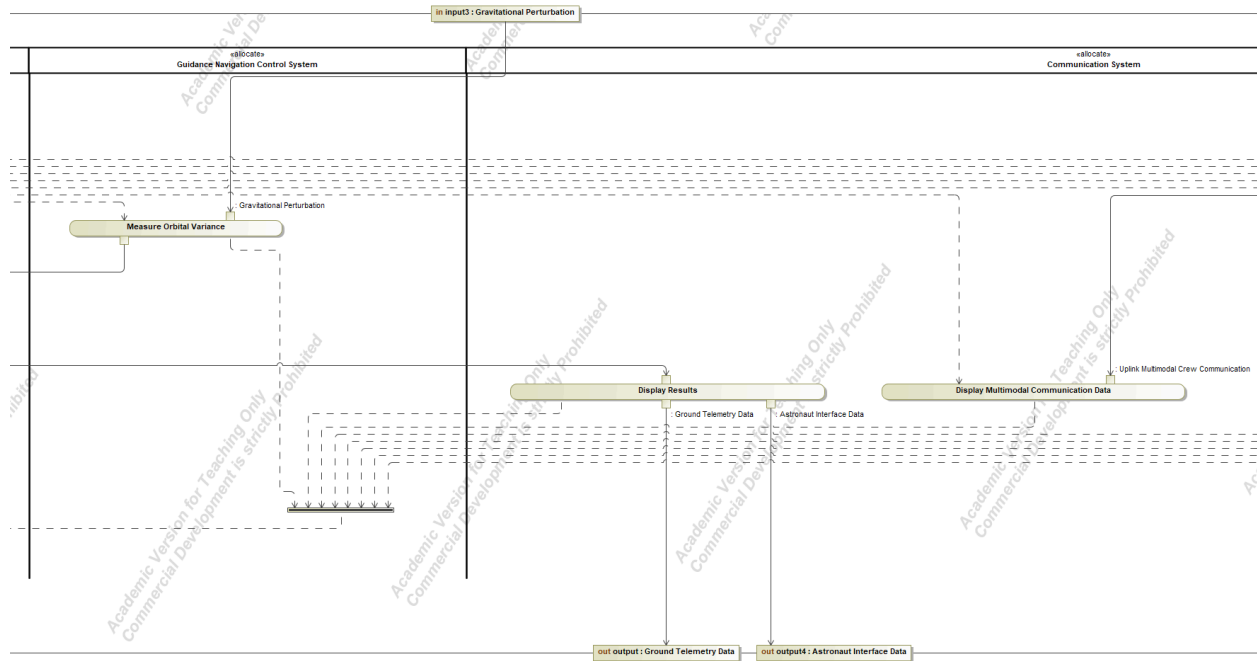


Figure 25-b: “Perform System Housekeeping” Functional Analysis (Guidance Navigation & Communication)

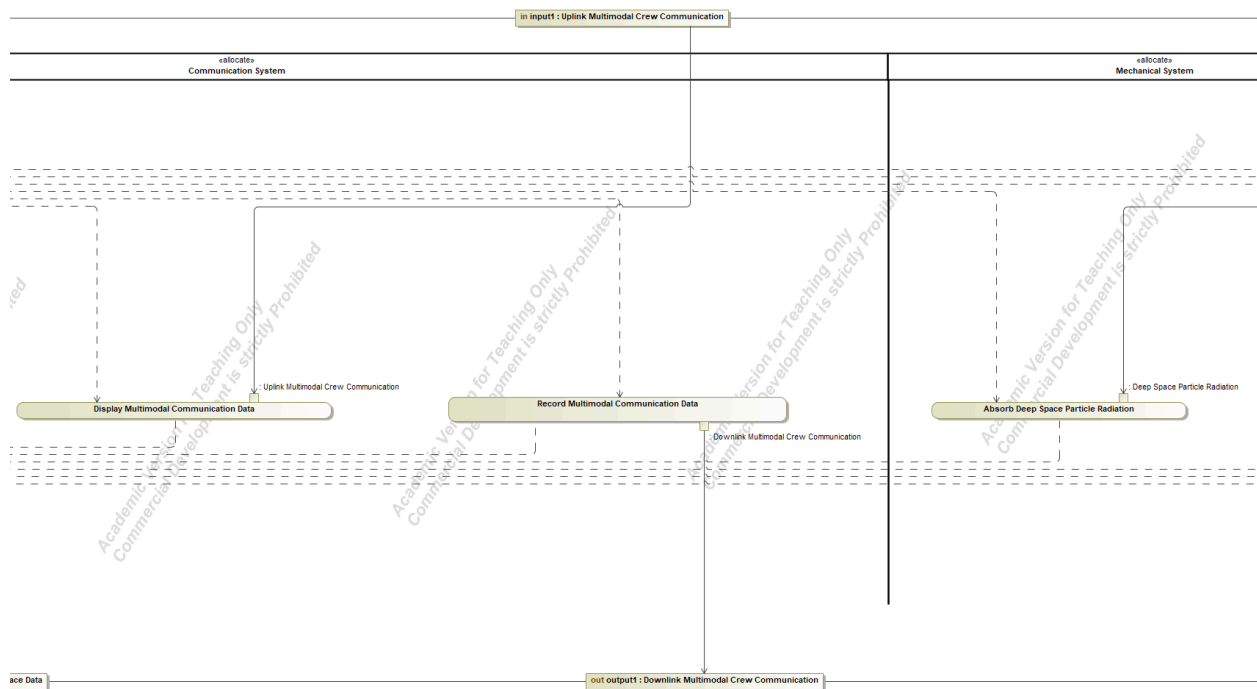


Figure 25-c: “Perform System Housekeeping” Functional Analysis (Communication & Mechanical)

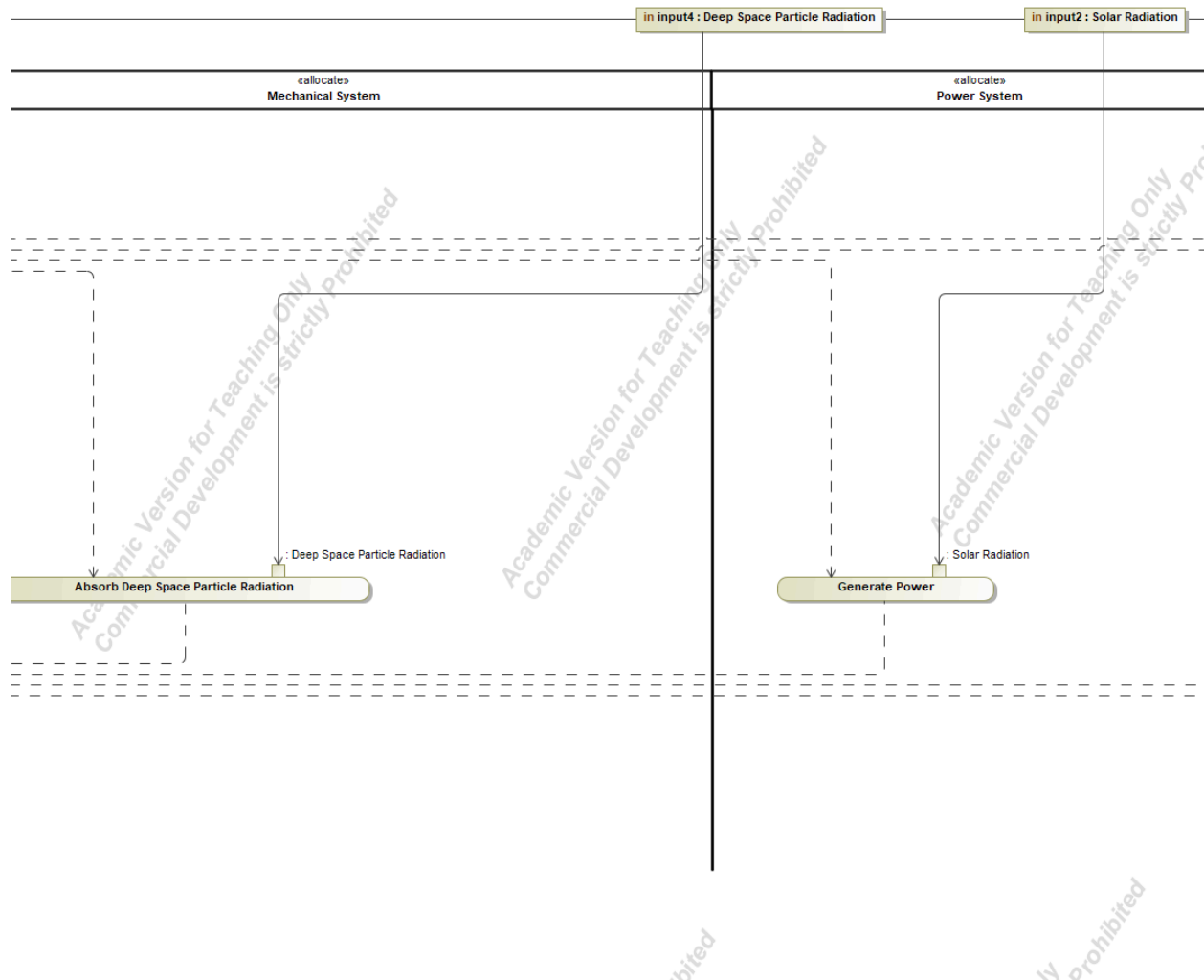


Figure 25-d: “Perform System Housekeeping” Functional Analysis (Mechanical & Power)

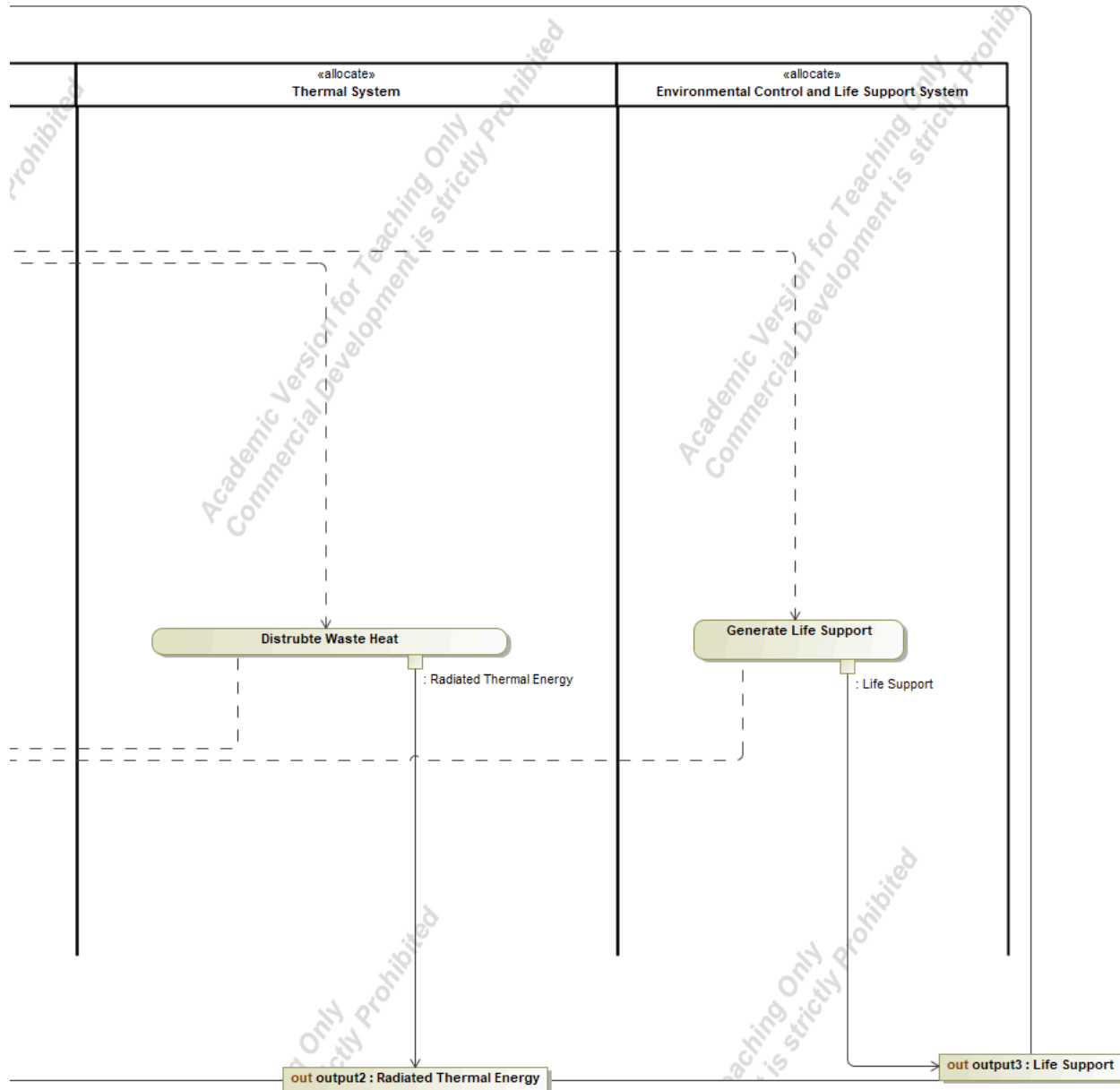


Figure 25-e: “Perform System Housekeeping” Functional Analysis (Thermal & Environmental Control / Life Support)

Figure 25 shows the next level of detail for the Perform Crewed Operations Use Case. This model is the breakdown of the Perform System Housekeeping action. The actors in this diagram relate now to the subsystems of the model, rather than the external actors.

5.5 CONCEPTUAL SUBSYSTEMS

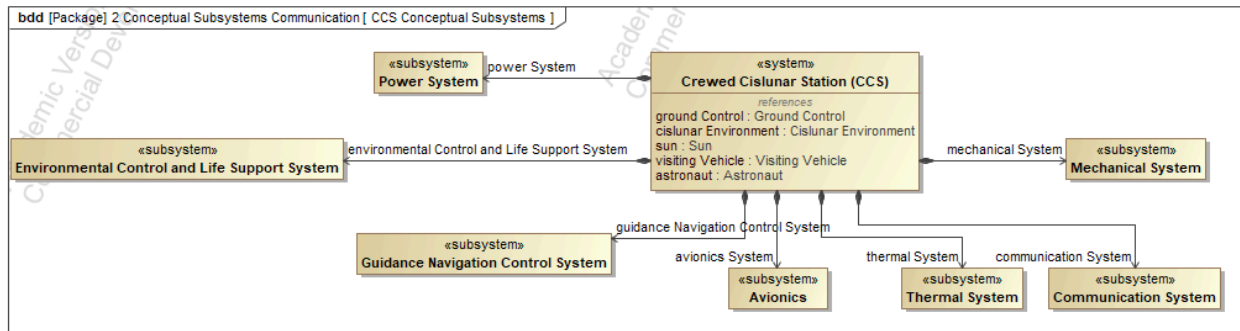


Figure 26: Conceptual Subsystems

Figure 25 shows the Conceptual Subsystems of the model. The Crewed Cislunar Station block contains the actors as shown in the System Context: Ground Control, Cislunar Environment, Sun, Visiting Vehicle, Astronaut. The main system is then connected to each of the subsystems present within the main system: Power System, Environmental Control and Life Support System, Guidance Navigation Control System, Avionics, Thermal System, Communication system, Mechanical System.

5.6 FAILURE MODE & EFFECTS ANALYSIS

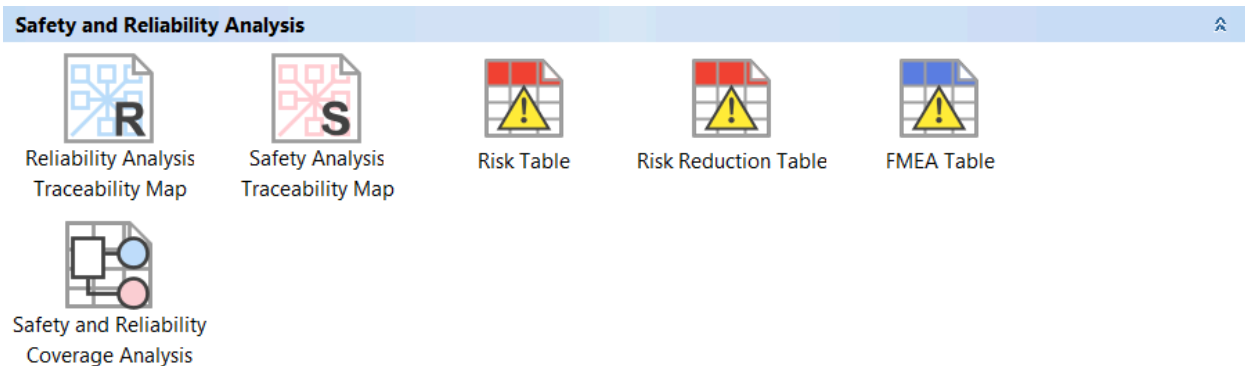


Figure 27: New Diagrams upon Installation of FMEA plugin

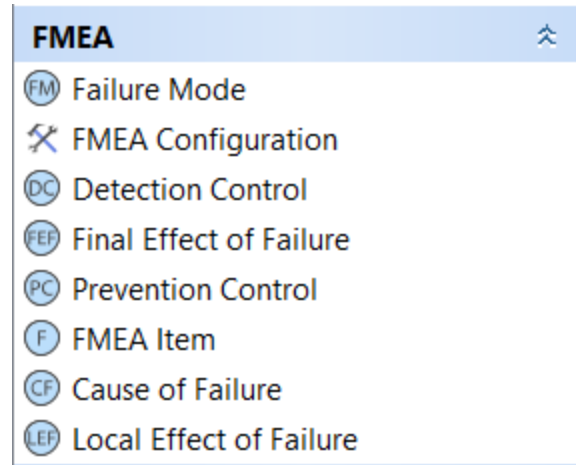


Figure 28: New Elements upon Installation of FMEA plugin

Figure 27 & 28 shows the new elements and diagrams we are able to create upon installing the required plugin for FMEA.

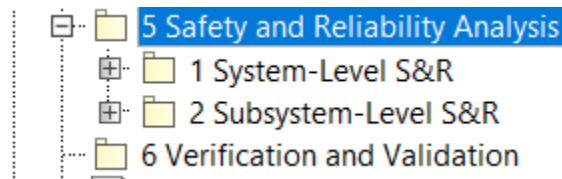


Figure 29: New Package Structure with FMEA

Figure 29 shows the new package structure, in a high-level, that the team will use to store and safeguard the FMEA resources. This structure may not be definitive as we may need to change a few things based on how many resources need to be created to represent FMEA within the system in both the system and subsystem level.

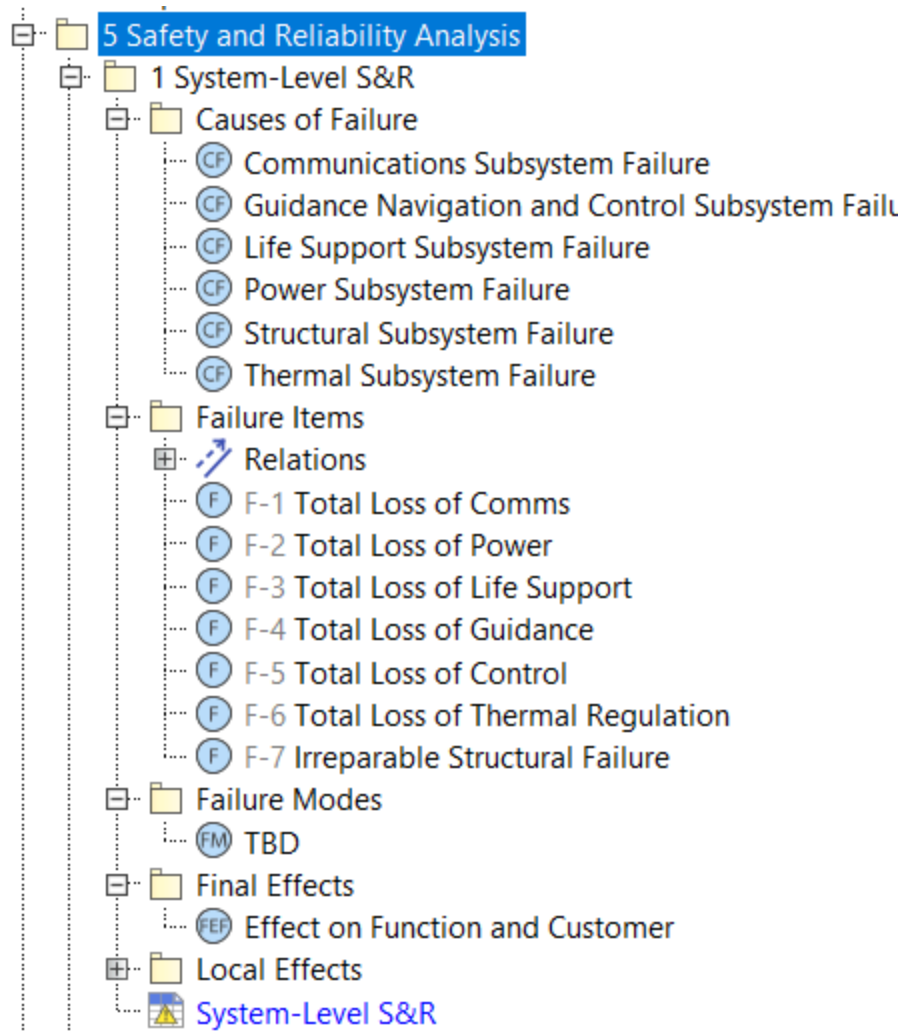


Figure 30: More detailed level of Package Structure

Figure 30 is more detailed on how the package structure is organized for FMEAs. As seen, we will have a package for every category that describes an FMEA Item. The only thing that has not been fully defined are the Failure Modes of the system.

| Criteria | | | | | | |
|-------------------------|-----|--------------------------------------|----------------|---------------|--|--|
| Element Type: FMEA Item | | Scope (optional): 1 System-Level S&R | | Filter: Y* | | |
| # | Id | Name | Classification | Item | Local Effect Of Failure | Final Effect Of Failure |
| 1 | F-1 | Total Loss of Comms | electrical | CCS System... | Crew is unable to contact ground control, and vice versa | Crew is unable to contact ground control |
| 2 | F-2 | Total Loss of Power | electrical | CCS System... | System is unable to provide power to any subsystems | Crew is unable to continue the mission |
| 3 | F-3 | Total Loss of Life Support | electrical | CCS System... | System is unable to sustain crew | |
| 4 | F-4 | Total Loss of Guidance | software | CCS System... | System is unable to navigate by itself in space | |
| 5 | F-5 | Total Loss of Control | mechanical | CCS System... | System is unable to control its orientation or correct its orbit | |
| 6 | F-6 | Total Loss of Thermal Regulation | mechanical | CCS System... | | |
| 7 | F-7 | Irreparable Structural Failure | mechanical | CCS System... | | |

Figure 31: FMEA Table View

Figure 31 shows what the FMEA Table looks like. It contains every category (column) that describes the creation of an FMEA Item.

Here we keep track of every FMEA category (column) that describes our FMEA under consideration. We will go a little bit in depth into what these columns try to describe.

Classification - What kind of failure is it, electrical, mechanical or software

Item - What component fails

Local Effect Of Failure - An element describing the effect that a **Failure Mode** has on the system element under consideration.

Final Effect Of Failure - An element describing the effect that **Failure Mode** has on an end user or environment.

Cause Of Failure - An element indicating the design weakness causing a **Failure Mode**.

Failure Mode - An element describing the specific manner in which a component, subsystem, system, process, etc.

The terms stated above are used to describe a possible risk, FMEA item, within the system. The table also captures the new requirements generated with the discovery and breakdown of the FMEA item under consideration.










| | |
|----|---|
| 10 |  Safety Requirements |
| 11 |  10 Comms Design Requirement |
| 12 |  15 Oxygen system shall have a backup oxygen supply standing by |
| 13 |  16 The system shall have the necessary backup communication systems |
| 14 |  17 The communication software system shall maintain a log with diagnostics |
| 15 |  18 The system shall have independent oxygen masks and oxygen tanks |
| 16 |  19 The system shall maintain a log of previous location data |
| 17 |  20 The communication system shall be radiation hardened with a safety factor of 1.5 |
| 18 |  9 Redundant Comms |

Figure 32: Some New Generated Requirements

Table 32 shows some of the new generated requirements within the FMEA Table. The blue badge symbol signifies that the requirement is a functional requirement. The yellow means safety and the pink/red signifies a hazard requirement.

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 the team 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 software Microsoft Teams.